

Version 8.2

Layer 7 Policy Authoring User Manual



1.800.681.9377
info@layer7tech.com
www.layer7tech.com



Copyright © 2014 CA. All rights reserved.

This documentation and any related computer software help programs (hereinafter referred to as the "Documentation") are for your informational purposes only and are subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW OR AS AGREED BY CA IN ITS APPLICABLE LICENSE AGREEMENT, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Contents

Chapter 1: Working with Service Policies	1
Policy Organization	2
Special Assertions	2
Hints and Tips	3
Policy Revisions	6
Configuring Policy Revisions	6
Creating a New Policy Revision	6
Working with Policy Revisions	6
Policy Properties	9
Organizing Services and Policies into Folders	12
Controlling Access Using Folders	12
Refining Access with Security Zones	13
Authentication in a Policy	14
Working with Aliases	15
Security Zones for Aliases	16
Working with Multiple Signatures	17
How to Permit Multiple Signatures	18
Configuring a Policy	18
Creating a Policy	21
Editing a Service Policy	22
Deleting a Policy	23
Disabling a Policy	24
Enabling a Policy	24
Validating a Policy	25
Instant Feedback Messages	25
Final Policy Validation Messages	26
Invalid Users or Groups	27
Invalid JMS Queue	28
Comparing Policies	28
Using the Policy Comparison Window	29
Viewing Assertion Differences	30
Working with Multiple Policy Tabs	32
Working with Internal Use Policies	33
wsdm-notifications	34
Audit Message Filter (AMF) Policy	34
Audit Viewer (AV) Policy	36
Working with Non-XML Messages	37
Assertions that Require XML	37
Assertions that Require SOAP	38
Example of a Branching Policy	38
Working with Comments	40
Adding a Comment	40
Editing a Comment	41
Deleting a Comment	42
Exporting/Importing a Policy	42
Exporting a Policy	42
Importing a Policy from a File	44
Importing a Policy via UDDI Registry	46

Import WS-Policy from URL in UDDI Registry Wizard	47
Searching the UDDI Registry	48
Resolve External Dependencies Wizard	51
Debugging a Policy	56
Working with the Service Debugger	56
Policy Debug Tracing	66
Working with the Debug Trace Policy	67
Managing Global Resources	72
Default Global Resources	73
Adding a New Global Resource	75
Editing a Global Resource	77
Deleting a Global Resource	77
Importing a Global Resource	77
Analyzing a Global Resource	84
Managing UDDI Registries	87
UDDI Registry Properties	89
Publish to UDDI Settings	92
Managing Meta Data	99
Chapter 2: Working with Policy Fragments	101
Types of Fragments	102
Adding a Policy Fragment to a Service Policy	103
Editing a Policy Fragment	104
Deleting a Policy	105
Working with Global Policy Fragments	106
Types of Global Policies	106
How Global Policies are Evaluated	107
How a Global Policy Relates to the Service Policy	108
Supported Assertions	108
Limitations to Global Policies	109
Chapter 3: Working with Policy Assertions	111
Adding an Assertion	112
Deleting an Assertion	119
Disabling an Assertion	119
Enabling an Assertion	121
Understanding Assertion Latency	122
How to Use the Assertion Latency Variables	122
When the Assertion Latency is Calculated	125
Working with Encapsulated Assertions	126
Encapsulated Assertions vs. Policy Fragments	126
Visibility of Context Variables	127
Understanding How Values are Passed to the Parent Policy	128
Making Encapsulated Assertions Available in a Role	130
Using Encapsulated Assertions	131
Managing Encapsulated Assertions	132
Encapsulated Assertion Configuration Properties	134
Policy Templates	142
Managing Kerberos Configuration	143
Authenticating a Client via Kerberos	145
Changing the WSS Assertion Recipient	146

New WSS Recipient Wizard	151
Selecting a Target Identity	152
Selecting a Target Message	153
Selecting an XPath	154
Namespace Map	156
Editing the Namespace Map	157
Migrating Namespaces	158
Using the XML Editor	159
Chapter 4: Access Control Assertions	161
Authenticate Against Identity Provider Assertion	163
Authenticate Against Radius Server Assertion	164
Context Variables Created by This Assertion	164
Authenticate Against SiteMinder Assertion	167
Authenticate User or Group Assertion	170
Authenticating Against a Simple LDAP Identity Provider	171
Authorize via SiteMinder Assertion	173
Check Protected Resource Against SiteMinder Assertion	175
Exchange Credentials using WS-Trust Assertion	177
Extract Attributes from Certificate Assertion	180
Context Variables for Subject/Issuer DN	180
Context Variables for Extended Attributes	182
Extract Attributes for Authenticated User Assertion	185
Perform JDBC Query Assertion	187
Unsupported Functionality	188
Context Variables Created by This Assertion	189
SQL Query Tips	196
Caching Metadata	206
Query LDAP Assertion	209
Require Encrypted UsernameToken Profile Credentials Assertion	213
Require FTP Credentials Assertion	214
Require HTTP Basic Credentials Assertion	215
Require HTTP Cookie Assertion	215
Context Variables Created by This Assertion	216
Require NTLM Authentication Credentials Assertion	217
Context Variables Created by This Assertion	219
Creating a Computer Account for NTLM Authentication	221
Require Remote Domain Identity Assertion	226
Context Variables Created by This Assertion	227
Require SAML Token Profile Assertion	228
Context Variables Created by This Assertion	229
SAML Token Profile Wizard	231
Require SSH Credentials Assertion	237
Require SSL or TLS Transport Assertion	238
Require Windows Integrated Authentication Credentials Assertion	241
Require WS-Secure Conversation Assertion	242
Context Variable Created by This Assertion	243
Require WS-Security Kerberos Token Profile Credentials Assertion	243
Require WS-Security Password Digest Credentials Assertion	244
Require WS-Security Signature Credentials Assertion	246
Require WS-Security UsernameToken Profile Credentials Assertion	248

Require XPath Credentials Assertion	248
Retrieve Credentials from Context Variable Assertion	250
Retrieve Kerberos Authentication Credentials Assertion	251
Using the Protocol Transition Delegation Method	252
Using the Constrained Proxy Delegation Method	254
Kerberos Service Ticket/Session Caching	255
Retrieve SAML Browser Artifact Assertion	258
Use WS-Federation Credential Assertion	263
Chapter 5: Transport Layer Security Assertions	267
Require SSL or TLS Transport Assertion	267
Chapter 6: XML Security Assertions	271
Add or Remove WS-Security Assertion	273
Add Security Token Assertion	277
Configuring the Private Key for SAML Assertions	277
Applying WS-Security	278
Adding a WS-S UsernameToken	279
Adding a WS-SC SecurityContextToken	281
Adding a SAML Assertion	281
Adding a WS-S EncryptedKey	282
Add Timestamp Assertion	283
Build RST SOAP Request Assertion	285
Context Variables Created by This Assertion	285
Build RSTR SOAP Response Assertion	288
Context Variables Created by This Assertion	289
Build SAML Protocol Request Assertion	291
SAML Protocol Request Wizard	292
Build SAML Protocol Response Assertion	299
Configuring the [General] Tab	301
Configuring the [Issuer] Tab (SAML 2.0 only)	305
Configuring the [Advanced] tab	306
Cancel Security Context Assertion	306
Configure WS-Security Decoration Assertion	309
Applying WS-Security	309
Configuring the [General] Tab	311
Configuring the [Signing] Tab	312
Configuring the [Encryption] Tab	313
Configuring the [Advanced] Tab	315
Create SAML Token Assertion	315
Context Variables Created by This Assertion	316
SAML Token Creation Wizard	317
Create Security Context Token Assertion	328
Context Variable Created by This Assertion	328
Create XACML Request Assertion	330
Configuring the Subject Node	332
Configuring the Resource Node	333
Configuring the Action Node	333
Configuring the Environment Node	334
Configuring the Attribute Node	334
Configuring the Multiple Attributes Node	340

Configuring the Resource Content Node	344
Encrypt Element Assertion	346
Establish Outbound Secure Conversation Assertion	348
Context Variables Created by This Assertion	349
Evaluate SAML Protocol Response Assertion	353
Evaluate XACML Policy Assertion	356
Generate OAuth Signature Base String Assertion	360
Context Variables Created by This Assertion	360
Generate Security Hash Assertion	365
Look Up Certificate Assertion	367
Look Up Outbound Secure Conversation Session Assertion	370
Context Variables Created by This Assertion	370
(Non-SOAP) Check Results from XML Verification Assertion	372
(Non-SOAP) Decrypt XML Element Assertion	374
Context Variables Created by This Assertion	374
(Non-SOAP) Encrypt XML Element Assertion	376
(Non-SOAP) Sign XML Element Assertion	377
(Non-SOAP) Validate SAML Token Assertion	380
Step 1: Introduction	381
Step 2: SAML Version	381
Step 3: SAML Statement Type	382
Step 4: Authentication Methods	383
Step 5: Authorization Statement	384
Step 6: Attribute Statement	385
Step 7: Subject Confirmation	386
Step 8: Name Identifier	389
Step 9: Conditions	390
Step 10: Embedded Signature	391
(Non-SOAP) Verify XML Element Assertion	391
Context Variables Created by This Assertion	391
Process RSTR Response Assertion	395
Context Variables Created by This Assertion	395
Protect Against Message Replay Assertion	397
Require Encrypted Element Assertion	400
Require Signed Element Assertion	402
Context Variables Created by This Assertion	402
Require Timestamp Assertion	405
Sign Element Assertion	407
Use WS-Security 1.1 Assertion	409
Chapter 7: Message Validation/ Transformation Assertions	411
Character Encoding	412
Add or Remove XML Element(s) Assertion	414
Add WS-Addressing Assertion	416
Context Variables Created by This Assertion	416
Applying the WS-Addressing Elements	416
Signing the WS-Addressing Elements	417
Apply JSON Transformation Assertion	419
Apply XSL Transformation Assertion	424
Context Variables Created by This Assertion	425
Compress Messages to/from SecureSpan XVC Assertion	429

Customize Error Response Assertion	430
Decode MTOM Message Assertion	432
Encode/Decode Data Assertion	435
Encode to MTOM Format Assertion	437
Enforce WS-Security Policy Compliance Assertion	441
Enforce WS-I BSP Compliance Assertion	441
Enforce WS-I SAML Compliance Assertion	443
Evaluate JSON Path Expression Assertion	445
Context Variables Created by This Assertion	445
Evaluate Regular Expression Assertion	449
Context Variables Created by This Assertion	450
Evaluate Request XPath Assertion	458
Context Variables Created by This Assertion	458
Evaluate Response XPath Assertion	461
Context Variables Created by This Assertion	462
Evaluate WSDL Operation Assertion	465
Process SAML Attribute Query Request Assertion	466
Context Variables Created by This Assertion	466
Process SAML Authentication Request Assertion	472
Context Variables Created by This Assertion	472
Replace Tag Content Assertion	475
Require WS-Addressing Assertion	477
Context Variables Created by This Assertion	477
Set SAML Response Status Code Assertion	480
Translate HTTP Form to MIME Assertion	482
Translate MIME to HTTP Form Assertion	484
Validate Certificate Assertion	486
Context Variables Created by This Assertion	486
Validate HTML Form Data Assertion	488
Validate JSON Schema Assertion	490
Validate MTOM Message Assertion	493
Validate or Change Content Type Assertion	495
Validate SOAP Attachments Assertion	497
Validate XML Schema Assertion	499
 Chapter 8: Message Routing Assertions	 507
Configure Message Streaming Assertion	508
Copy Request Message to Response Assertion	510
Manage Cookie Assertion	512
Manage Transport Properties/Headers Assertion	515
Return Template Response to Requestor Assertion	518
Route via FTP(S) Assertion	520
FTP Cluster Properties for This Assertion	522
Configuring the [Connection] Tab	523
Configuring the [Authentication] Tab	526
Configuring the [Advanced] Tab	528
Route via HTTP(S) Assertion	529
Configuring the [Authentication] Tab	531
Configuring the [Headers] Tab	533
Configuring the [Connection] Tab	536
Configuring the [HTTP] Tab	538

Configuring the [Proxy] Tab	539
Configuring the [Other] Tab	539
Route via JMS Assertion	541
Context Variables Created by This Assertion	542
Configuring the [Target] Tab	544
Configuring the [Security] Tab	546
Configuring the [Request] Tab	548
Configuring the [Response] Tab	550
Route via MQ Native Assertion	551
Context Variables Created by This Assertion	552
Defined MQ Header Prefixes	553
Configuring the [Target] Tab	554
Configuring the [Request] Tab	556
Configuring the [Response] Tab	558
Route via Raw TCP Assertion	560
Route via SSH2 Assertion	563
Performing SFTP Partial Downloads/Uploads	563
Chapter 9: Service Availability Assertions	573
Apply Rate Limit Assertion	573
Apply Throughput Quota Assertion	578
Context Variables Created by This Assertion	579
Limit Availability to Time/Days Assertion	584
Look Up in Cache Assertion	585
Query Rate Limit Assertion	587
Context Variables Created by This Assertion	587
Query Throughput Quota Assertion	589
Context Variables Created by This Assertion	589
Resolve Service Assertion	590
Restrict Access to IP Address Range Assertion	592
Store to Cache Assertion	594
Chapter 10: Logging, Auditing, and Alerts Assertions	597
Message Auditing	597
System Audits	598
Administrative Audits	598
Policy Message Audits	598
Add Audit Detail Assertion	600
Audit Messages in Policy Assertion	602
Capture Identity of Requestor Assertion	604
Customize SOAP Fault Response Assertion	607
SOAP Faults	611
Send Email Alert Assertion	612
Send SNMP Trap Assertion	615
Chapter 11: Policy Logic Assertions	617
Add Comment to Policy Assertion	618
All Assertions Must Evaluate to True Assertion	619
At Least One Assertion Must Evaluate to True Assertion	619
Compare Expression Assertion	621
Continue Processing Assertion	625

Create Routing Strategy Assertion	626
Context Variables Created by This Assertion	626
Execute Routing Strategy Assertion	630
Context Variables Created by This Assertion	630
Export Variables from Fragment Assertion	632
When Used in a Global Policy Fragment	632
Generate UUID Assertion	634
Include Policy Fragment Assertion	635
Join Variable Assertion	636
Look Up Context Variable	637
Context Variables Created by This Assertion	638
Look Up Item by Index Position Assertion	640
Look Up Item by Value Assertion	641
Manipulate Multivalued Variable Assertion	642
Map Value Assertion	644
Process Routing Strategy Result Assertion	648
Run All Assertions Concurrently Assertion	651
Technical Issues to Consider	651
Configuring the Assertion	652
Run Assertions for Each Item Assertion	653
Context Variables Created by this Assertion	654
Set Context Variable Assertion	656
Split Variable Assertion	661
Stop Processing Assertion	664
Chapter 12: Threat Protection Assertions	665
Automatic Threat Protection	666
TCP/IP-Based Attacks	666
Coercive Parsing and XML Bomb	666
External Entity Attack	667
Schema Poisoning	667
WSDL Scanning	667
XML Routing Detours	668
Limit Message Size Assertion	668
Protect Against Code Injection Assertion	670
Protect Against Cross-Site Request Forgery Assertion	672
Context Variable Created by This Assertion	673
Protect Against Document Structure Threats Assertion	675
Protect Against JSON Document Structure Threats Assertion	678
Protect Against Message Replay Assertion	680
Protect Against SQL Attack Assertion	684
SQL Injections Detected	684
Scan Using ICAP-Enabled Antivirus Assertion	687
Context Variables Created by This Assertion	688
Scan Using Sophos Antivirus Assertion	692
Context Variables Created by This Assertion	692
Validate or Change Content Type Assertion	694
Validate JSON Schema Assertion	696
Validate OData Request Assertion	699
Retrieving the Service Metadata Document	700
Notes and Limitations	700

Context Variables Created by This Assertion	700
Validate XML Schema Assertion	703
Chapter 13: Internal Assertions	711
Collect WSDM Metrics Assertion	711
Convert Audit Record to XML Assertion	713
Handle UDDI Subscription Notification Assertion	714
Manage Gateway Assertion	715
Context Variables Created by Assertion	715
REST Manage Gateway Assertion	716
Context Variables Used by Assertion	717
Subscribe to WSDM Resource Assertion	717
Chapter 14: Custom Assertions	719
Access Resource Protected by JSAM Assertion	720
Context Variables Created by This Assertion	720
Access Resource Protected by Oracle Access Manager Assertion	722
Context Variables Created by This Assertion	722
Authenticate using Tivoli Access Manager Assertion	727
Usage Rules	727
Using the Assertion	728
Troubleshooting	730
Authenticate with SiteMinder R12 Protected Resource Assertion	730
Context Variables Created by This Assertion	731
Execute Salesforce Operation Assertion	734
Context Variables Created by This Assertion	735
Using the Assertion	737
Creating Objects	738
Updating Objects	740
Retrieving Objects	742
Retrieving Modified Objects	744
Retrieving Deleted Objects	745
Executing Queries	747
Searching Objects	748
Exporting/Importing Policies	749
Scan Using Symantec Antivirus Assertion	749
Index	751

List of Figures

Figure 1: Sample policy assertion tree in the policy development window	3
Figure 2: Policy with multiple identities	4
Figure 3: Policy Revisions dialog	7
Figure 4: Policy Properties	9
Figure 5: Mechanisms for gathering credentials	15
Figure 6: Enabling multiple signatures in a policy	18
Figure 7: Choosing a revision to activate when enabling a policy	24
Figure 8: The Policy Diff dialog	29
Figure 9: Policy Diff window - Properties tab	31
Figure 10: Policy Diff window - Raw XML tab	32
Figure 11: Default AMF policy	35
Figure 12: Default AV policy	36
Figure 13: Enter Comment dialog	40
Figure 14: Import WS-Policy from URL in UDDI Registry Wizard	47
Figure 15: Search UDDI dialog (Search UDDI Registry for WSDL example)	49
Figure 16: Selecting a wsdl:port	50
Figure 17: Sample unresolved dependency in Resolve External Dependencies Wizard	52
Figure 18: Example Service Debugger dialog, with a debug session in progress	59
Figure 19: Sample message variable	64
Figure 20: Debug Trace Policy on the interface	68
Figure 21: Debug trace default policy	69
Figure 22: Manage Global Resources dialog	74
Figure 23: Adding a Global Resource (XML Schema)	75
Figure 24: Resource Import Wizard	78
Figure 25: Analyze Global Resource dialog	85
Figure 26: Manage UDDI Registries dialog	87
Figure 27: UDDI Registries Properties dialog	89
Figure 28: Publish to UDDI Settings - [Service] tab	93
Figure 29: Publish to UDDI Settings - [WS-Policy] tab	96
Figure 30: Manage Meta Data dialog	99
Figure 31: A disabled assertion	120
Figure 32: Saving assertion latency by logging the message	123
Figure 33: Saving assertion latency by copying the value to another variable	124
Figure 34: Example of when assertion latency is calculated	125
Figure 35: Manage Encapsulated Assertion Configurations dialog (with sample assertions)	132
Figure 36: Encapsulated Assertion Configuration Properties dialog	136
Figure 37: Kerberos Configuration dialog	143
Figure 38: Configuring the HTTP(S) Routing Properties for Kerberos authentication	146
Figure 39: Change WSS Recipient dialog	150
Figure 40: New WSS Recipient Wizard	151
Figure 41: Selecting a message target	153
Figure 42: User interface for selecting an XPath	154
Figure 43: Edit Namespaces and Prefixes dialog	157
Figure 44: Migrate Namespaces dialog	159
Figure 45: Changing the identity provider used for authentication	164
Figure 46: Authenticate Against Radius Server Properties	166
Figure 47: Authenticate Against SiteMinder Properties	168
Figure 48: Search Identity Provider dialog	172
Figure 49: Sample policy for setting the SiteMinder Cookie	174
Figure 50: Authorize via SiteMinder Properties	174
Figure 51: SiteMinder Check Protected Resource Properties	176
Figure 52: WS-Trust Credential Exchange Properties	178

Figure 53: Certificate Attributes Properties	184
Figure 54: Identity Attributes Properties	185
Figure 55: User Attribute Mapping dialog	186
Figure 56: Multiple result sets example	190
Figure 57: JDBC Query Properties	192
Figure 58: LDAP Query Properties	210
Figure 59: Require Encrypted UsernameToken Profile Credentials Properties	214
Figure 60: HTTP Cookie Properties	217
Figure 61: NTLM Authentication Properties	220
Figure 62: Server Manager	221
Figure 63: Linux Properties - General tab	222
Figure 64: Configuring the [Delegation] tab for a new computer account	223
Figure 65: Add Services dialog	223
Figure 66: Select Users or Computers dialog	224
Figure 67: Add Services dialog containing available services	225
Figure 68: The netlogon service is displayed in the Delegation tab	226
Figure 69: Remote Domain Identity Properties	228
Figure 70: Example of SAML Token Profile Wizard in edit mode	230
Figure 71: SAML Token Profile Wizard	231
Figure 72: Require SSH Credentials Properties	238
Figure 73: SSL or TLS Transport Properties	239
Figure 74: Policy to handle both Kerberos and NTLM protocols	241
Figure 75: Require WS-Security Password Digest Credentials Properties	245
Figure 76: WS-Security Signature Properties	247
Figure 77: XPath Credentials Properties	249
Figure 78: Credentials from Context Variable Properties	251
Figure 79: Kerberos Authentication Credential Properties	256
Figure 80: SAML Browser Artifact Properties	259
Figure 81: Configure Authentication	261
Figure 82: WS-Federation Request Properties	264
Figure 83: SSL or TLS Transport Properties	268
Figure 84: WS-Security Properties	274
Figure 85: Security Token Properties - WS-S UsernameToken	279
Figure 86: Security Token Properties - WS-SC SecurityContextToken	281
Figure 87: Security Token Properties - SAML Assertion	282
Figure 88: Security Token Properties - WS-S EncryptedKey	282
Figure 89: Timestamp Properties	284
Figure 90: RST SOAP Request Builder Properties	286
Figure 91: RSTR SOAP Response Builder Properties	290
Figure 92: SAML Protocol Request Wizard	292
Figure 93: SAML Protocol Request Wizard - Step 8: Name Identifier (SAML 2.x version shown)	295
Figure 94: SAML Protocol Request Wizard - Step 9: Subject Confirmation	297
Figure 95: SAML Protocol Response Properties - [General] tab (SAML 2.0 shown)	301
Figure 96: Security Context Cancellation Properties	308
Figure 97: Configure WS-Security Decoration Properties - [General] tab	311
Figure 98: Configure WS-Security Decoration Properties - [Signing] tab	312
Figure 99: Configure WS-Security Decoration Properties - [Signing] tab	313
Figure 100: Configure WS-Security Decoration Properties - [Advanced] tab	315
Figure 101: SAML Token Creation Wizard	317
Figure 102: SAML Token Creation Wizard - Step 6: Attribute Statement (SAML 2.x version shown) ..	321
Figure 103: Edit SAML Attribute Properties dialog (SAML 2.x version shown)	322
Figure 104: SAML Token Creation Wizard - Step 8: Subject Confirmation	326
Figure 105: Security Context Token Creator Properties	329
Figure 106: XACML Request Properties	331

Figure 107: AttributeValue node	336
Figure 108: Multiple Attributes node (XACML 2.0 screen)	340
Figure 109: Resource Content node	345
Figure 110: Encrypt Request Element Properties dialog	347
Figure 111: Outbound Secure Conversation Properties	350
Figure 112: SAML Protocol Response Wizard	354
Figure 113: XACML Policy Properties	357
Figure 114: Generate OAuth Signature Base String Properties - Client mode	362
Figure 115: Generate OAuth Signature Base String Properties - Server mode	363
Figure 116: Generate Security Hash Properties	366
Figure 117: Certificate Lookup Properties	368
Figure 118: Outbound Secure Conversation Session Lookup Properties	371
Figure 119: (Non-SOAP) Check Results from XML Verification Properties	373
Figure 120: (Non-SOAP) XML Element Decryption Properties	375
Figure 121: (Non-SOAP) XML Element Encryption Properties	377
Figure 122: (Non-SOAP) XML Element Signature Properties	378
Figure 123: (Non-SOAP) Validate SAML Token Properties - Step 1	381
Figure 124: (Non-SOAP) Validate SAML Token Properties - Step 2	382
Figure 125: (Non-SOAP) Validate SAML Token Properties - Step 3	382
Figure 126: (Non-SOAP) Validate SAML Token Properties - Step 4	383
Figure 127: (Non-SOAP) Validate SAML Token Properties - Step 5	384
Figure 128: (Non-SOAP) Validate SAML Token Properties - Step 6	385
Figure 129: (Non-SOAP) Validate SAML Token Properties - Step 7	386
Figure 130: (Non-SOAP) Validate SAML Token Properties - Step 8	389
Figure 131: (Non-SOAP) Validate SAML Token Properties - Step 9	390
Figure 132: (Non-SOAP) Validate SAML Token Properties - Step 10	391
Figure 133: (Non-SOAP) XML Element Verification Properties	393
Figure 134: RSTR Response Processor Properties	396
Figure 135: Message Replay Protection Properties	399
Figure 136: Encrypted Element Properties	401
Figure 137: Signed Element Properties	404
Figure 138: Timestamp Properties	406
Figure 139: Sign Element Properties	408
Figure 140: Add or Remove XML Elements Properties	415
Figure 141: Add WS-Addressing Properties	417
Figure 142: JSON Transformation Properties	420
Figure 143: JSON Transformation Properties - [Test] tab	423
Figure 144: XSL Transformation Properties	426
Figure 145: Compression Properties	430
Figure 146: Error Response Properties	431
Figure 147: MTOM Decode Properties	434
Figure 148: Encode/Decode Properties	436
Figure 149: MTOM Encode Properties	439
Figure 150: WS-I BSP Compliance Properties	442
Figure 151: WS-I SAML Compliance Properties	444
Figure 152: Evaluate JSON Path Expression Properties - [Source and Destination] tab	447
Figure 153: Evaluate JSON Path Expression Properties - [Test] tab	449
Figure 154: Regular Expression Properties - [Source and Destination] tab	451
Figure 155: Evaluate Regular Expression assertion - [Test] tab	458
Figure 156: WSDL Operation Properties	466
Figure 157: SAML Attribute Query Request Properties	469
Figure 158: SAML Authentication Request Properties	474
Figure 159: Replace Tag Content Properties	476
Figure 160: WS-Addressing Properties	479

Figure 161: SAML Response Status Properties dialog	481
Figure 162: HTTP Form to MIME Translation Properties	483
Figure 163: Configure Field Information form	483
Figure 164: MIME to HTTP Form Translation Properties	485
Figure 165: Configure Field Name form	485
Figure 166: Validate Certificate Properties	487
Figure 167: HTML Form Data Properties	489
Figure 168: JSON Schema Validation Properties	491
Figure 169: MTOM Validate Properties	494
Figure 170: MTOM Validate Properties - Rule	495
Figure 171: Content Type Properties	496
Figure 172: SOAP Attachment Properties	498
Figure 173: XML Schema Validation Properties	501
Figure 174: Extract Schema from WSDL dialog	504
Figure 175: Confirming importing schema dependencies	505
Figure 176: Select Import Option dialog	506
Figure 177: Configure Message Streaming Properties	509
Figure 178: Request to Response Properties	511
Figure 179: Cookie Properties, with "Update" operation example	513
Figure 180: Transport Properties/Headers Properties	516
Figure 181: Template Response Properties	519
Figure 182: FTP(S) Routing Properties - [Connection] tab	523
Figure 183: FTP(S) Routing Properties - [Authentication] tab	526
Figure 184: FTP(S) Routing Properties - [Advanced] tab	528
Figure 185: JMS Routing Properties - [Target] tab	544
Figure 186: JMS Routing Properties - [Security] tab	546
Figure 187: JMS Routing Properties - [Request] tab	548
Figure 188: JMS Routing Properties - [Response] tab	550
Figure 189: MQ Native Routing Properties - [Target] tab	554
Figure 190: MQ Native Routing Properties - [Request] tab	556
Figure 191: MQ Native Routing Properties - [Response] tab	558
Figure 192: Raw TCP Routing Properties	561
Figure 193: Sample policy for SFTP partial downloads	563
Figure 194: SSH2 Routing Properties - [Connection] tab	565
Figure 195: SSH2 Routing Properties - [Authentication] tab	569
Figure 196: SSH2 Routing Properties - [Advanced] tab	570
Figure 197: Rate Limit Properties	574
Figure 198: Throughput Quota Properties	580
Figure 199: Time/Day Availability Properties	584
Figure 200: Cache Lookup Properties	586
Figure 201: Rate Limit Query Properties	588
Figure 202: Throughput Quota Query Properties	590
Figure 203: Resolve Service Properties	591
Figure 204: IP Address Range Properties	593
Figure 205: Cache Storage Properties	595
Figure 206: Audit message path	599
Figure 207: Audit Detail Properties	601
Figure 208: Audit Properties	603
Figure 209: Viewing message context mappings in the Gateway Audit Events window	605
Figure 210: Requestor Identity Properties	606
Figure 211: Fault Response Properties	609
Figure 212: Email Alert Properties	613
Figure 213: SNMP Trap Properties	615
Figure 214: Comment Properties	618

Figure 215: Compare Expression Properties	622
Figure 216: Create Routing Strategy Properties	627
Figure 217: Execute Routing Strategy Properties	631
Figure 218: Export Variables from Fragment Properties	633
Figure 219: Generate UUID Properties	635
Figure 220: Selecting a policy fragment to include	636
Figure 221: Join Variable Properties	637
Figure 222: Look Up Context Variable Properties	639
Figure 223: Look Up Item by Index Position Properties	640
Figure 224: Look Up Item by Value Properties	642
Figure 225: Manipulate Multivalued Variable Properties	643
Figure 226: Map Value Properties	647
Figure 227: Process Routing Strategy Result Properties	649
Figure 228: Sample Policy Fragment for Run Assertions for Each Item assertion	653
Figure 229: Run Assertions for Each Item Properties	655
Figure 230: Context Variable Properties - data type "Message"	657
Figure 231: Context Variable Properties - data type "Date/Time"	658
Figure 232: Split Variable Properties	662
Figure 233: Message Size Limit Properties	669
Figure 234: Code Injection Protection Properties	671
Figure 235: CSRF Protection Properties	674
Figure 236: Document Structure Threat Protection Properties	676
Figure 237: JSON Document Structure Threat Protection Properties	679
Figure 238: Message Replay Protection Properties	683
Figure 239: SQL Attack Protection Properties	686
Figure 240: ICAP Antivirus Scanner Properties	689
Figure 241: Sophos Antivirus Properties	693
Figure 242: Content Type Properties	695
Figure 243: JSON Schema Validation Properties	697
Figure 244: OData Validate Request Properties	702
Figure 245: XML Schema Validation Properties	705
Figure 246: Extract Schema from WSDL dialog	708
Figure 247: Confirming importing schema dependencies	710
Figure 248: Select Import Option dialog	710
Figure 249: Gateway Management Properties	716
Figure 250: WSDM Subscription Properties	718
Figure 251: Access Resource Protected by JSAM Properties	721
Figure 252: Oracle Access Manager Protected Resource Properties	724
Figure 253: Tivoli Access Manager Authentication Properties	729
Figure 254: SiteMinder R12 Custom Assertion Properties	732
Figure 255: Execute Salesforce Operation Wizard: Configure Connection	738
Figure 256: Execute Salesforce Operation Wizard: Configure Action - Create Objects	739
Figure 257: Execute Salesforce Operation Wizard: Configure Action - Update Objects	741
Figure 258: Execute Salesforce Operation Wizard: Configure Action - Retrieve Objects	743
Figure 259: Execute Salesforce Operation Wizard: Configure Action - Retrieve Modified Objects	744
Figure 260: Execute Salesforce Operation Wizard: Configure Action - Retrieve Deleted Objects	746
Figure 261: Execute Salesforce Operation Wizard: Configure Action - Execute Query	747
Figure 262: Execute Salesforce Operation Wizard: Configure Action - Search Objects	748

List of Tables

Table 1: Policy revision tasks	7
Table 2: Policy Properties settings	10
Table 3: Organizing assertions in the policy development window	20
Table 4: Instant feedback message types	25
Table 5: Troubleshooting final policy validation errors	26
Table 6: Resolving identity errors	27
Table 7: Sample branching policy for XML	39
Table 8: Options for importing a file	44
Table 9: Using the Import WS-Policy from URL in UDDI	47
Table 10: Configuring the Search UDDI settings	49
Table 11: Resolve External Dependencies Wizard settings	52
Table 12: Context variables for debug trace policy	71
Table 13: Global Resource settings	75
Table 14: Analyze Global Resource dialog	85
Table 15: Manage UDDI Registries columns	87
Table 16: Manage UDDI Registries tasks	88
Table 17: UDDI Registries settings	89
Table 18: Publish to UDDI Settings - [Service] tab	94
Table 19: [Service] tab - Publishing Status	96
Table 20: Publish to UDDI Settings - [WS-Policy] tab	97
Table 21: keyedReference settings	99
Table 22: Policy fragment tasks	102
Table 23: Policy Manager assertions	112
Table 24: Encapsulated assertions vs. Policy fragments	127
Table 25: Encapsulated assertion tasks	131
Table 26: Encapsulated assertion configuration columns	132
Table 27: Manage Encapsulated Assertions tasks	133
Table 28: Encapsulated assertions: Argument Properties	139
Table 29: Encapsulated assertions: Argument Properties	140
Table 30: Encapsulated assertions: Result Properties	141
Table 31: Editing policy templates	142
Table 32: Kerberos Configuration settings	143
Table 33: WSS Assertion Recipient tasks	150
Table 34: Using the New WSS Recipient Wizard	151
Table 35: Namespace map actions	157
Table 36: XML Editor options	159
Table 37: XML Editor keyboard shortcuts	160
Table 38: Context variables created by Authenticate Against Radius Server assertion	164
Table 39: Radius reason codes	165
Table 40: Authenticate Against Radius Server settings	166
Table 41: Authenticate Against SiteMinder settings	169
Table 42: Search Identity Provider settings	172
Table 43: Authorize via SiteMinder settings	174
Table 44: SiteMinder Check Protected Resource settings	176
Table 45: WS-Trust Credential Exchange settings	179
Table 46: Context variables for Subject/Issuer DN in an X.509 certificate	180
Table 47: Context variables for extended attributes in an X.509 certificate	182
Table 48: Identity Attributes settings	186
Table 49: User Attribute Mapping settings	187
Table 50: Perform JDBC Query context variables	189

Table 51: JDBC Query settings	193
Table 52: Default output variables from a function	199
Table 53: Messages for procedures or functions with no parameters	199
Table 54: Supported data types	205
Table 55: Boolean value support	205
Table 56: LDAP Query settings	210
Table 57: Inbound context variables created by Require NTLM Authentication assertion	219
Table 58: NTLM Authentication settings	220
Table 59: Context variables created by Require Remote Domain Identity assertion	227
Table 60: SAML Token Profile Wizard tabs in edit mode	230
Table 61: Using the SAML Token Profile Wizard	231
Table 62: SSL or TLS Transport settings	239
Table 63: WS-Security Signature settings	245
Table 64: WS-Security Signature settings	247
Table 65: XPath Credentials settings	249
Table 66: Kerberos Authentication Credentials settings	256
Table 67: SAML Browser Artifact settings	259
Table 68: Authentication methods	260
Table 69: Authentication methods	261
Table 70: WS-Federation Request settings	264
Table 71: SSL or TLS Transport settings	268
Table 72: WS-Security Properties settings	275
Table 73: Configured private key for various SAML Assertion types	277
Table 74: Adding a WS-S UsernameToken	279
Table 75: Timestamp settings	284
Table 76: Context variables created by Build RST SOAP Request assertion	285
Table 77: RST SOAP Request Builder settings	286
Table 78: Context variables created by Build RSTR SOAP Response Assertion	289
Table 79: Using the SAML Protocol Request Wizard	293
Table 80: SAML Protocol Response Properties - [General] tab	301
Table 81: SAML Protocol Response Properties - [Issuer] tab - SAML 2.0 only	305
Table 82: Security Context Cancellation settings	308
Table 83: WS-Security decorations in Configure WS-Security Decoration assertion	309
Table 84: Context variables created by Create SAML Token assertion	316
Table 85: Using the SAML Token Creation Wizard	318
Table 86: SAML Attribute Properties settings	322
Table 87: Filter options in the Attribute StatementTable 3	325
Table 88: Security Context Token Creator settings	329
Table 89: AttributeValue node settings	336
Table 90: Effects of multivalued variables in AttributeValues	337
Table 91: Multiple Attribute node settings	340
Table 92: Resource Content node settings	345
Table 93: Outbound secure conversation session attributes	349
Table 94: Outbound Secure Conversation settings	350
Table 95: SAML Protocol Response Wizard settings	354
Table 96: XACML Policy settings	357
Table 97: Context variables created by Generate OAuth Signature Base String assertion	360
Table 98: Generate OAuth Signature Base String Properties settings	363
Table 99: Generate Security Hash settings	366
Table 100: Certificate Lookup settings	369
Table 101: Outbound secure conversation session attributes	370
Table 102: Context variables created by (Non-SOAP) Decrypt XML Element assertion	374
Table 103: (Non-SOAP) XML Element Signature settings	379
Table 104: Context variables created by (Non-SOAP) Verify XML Element assertion	392

Table 105: (Non-SOAP) XML Element Verification settings	393
Table 106: Context variables created by Process RSTR Response assertion	395
Table 107: Message Replay Protection settings	399
Table 108: Context variables created by Require Signed Element assertion	402
Table 109: Timestamp settings	406
Table 110: Add or Remove XML Elements settings	415
Table 111: Add WS-Addressing settings	418
Table 112: JSON Transformation Settings	420
Table 113: Transformation stylesheet locations	427
Table 114: Error Response settings	431
Table 115: MTOM Decode Settings	434
Table 116: Encode/Decode Data settings	436
Table 117: MTOM Encode Settings	439
Table 118: WS-I BSP Compliance Properties settings	443
Table 119: WS-I SAML Compliance Properties settings	444
Table 120: Context variables created by Evaluate JSON Path Expression assertion	446
Table 121: Evaluate JSON Path Expression Properties - basic settings	447
Table 122: Evaluate JSON Path Expression Properties - [Source and Destination] tab	447
Table 123: Regular Expression Properties - basic settings	451
Table 124: Regular Expression Properties - Source & Destination settings	453
Table 125: Context variables created by Evaluate Request XPath assertion	459
Table 126: Context variables created by Evaluate Response XPath assertion	462
Table 127: Context variables created by Process SAML Attribute Query Request assertion	467
Table 128: SAML Attribute Query Request settings	470
Table 129: Context variables created by Process Authentication Request assertion	473
Table 130: SAML Authentication Request settings	475
Table 131: Replace Tag Content settings	476
Table 132: Context variables created by Require WS-Addressing assertion	477
Table 133: WS-Addressing settings	479
Table 134: SAML Response Status settings	482
Table 135: HTTP Form to MIME Translation settings	483
Table 136: Configure Field Information	484
Table 137: MIME to HTTP Form Translation actions	485
Table 138: Context variables created by the Validate Certificate assertion	486
Table 139: Validate Certificate settings	487
Table 140: HTML Form Data settings	489
Table 141: Configuring the JSON schema based on location	492
Table 142: MTOM Validate Settings	494
Table 143: Rate Limit settings	496
Table 144: SOAP Attachment Properties settings	498
Table 145: Configuring the schema based on location	502
Table 146: Extract Schema from WSDL settings	504
Table 147: Configure Message Streaming settings	509
Table 148: Request to Response Properties settings	511
Table 149: Cookie tasks	513
Table 150: Transport Properties/Headers tasks	516
Table 151: Template Response settings	519
Table 152: FTP Cluster Properties that only affect the Route via FTP(S) assertion	522
Table 153: FTP(S) connection settings	524
Table 154: FTP(S) command settings	524
Table 155: WSS header handling	528
Table 156: Route via HTTP(S): Authentication methods	531
Table 157: Scenarios for header rules	535
Table 158: Retrieving IP addresses during HTTP routing	536

Table 159: Failover Strategies during HTTP routing	536
Table 160: WSS Header Handling during HTTP routing	539
Table 161: Assertion Outcome during HTTP routing	541
Table 162: Context variables created by the Route via JMS assertion	542
Table 163: Dynamic properties for template outbound destinations	545
Table 164: Service Authentication during JMS routing	546
Table 165: WSS Header Handling during JMS routing	547
Table 166: Defining the JMS properties for forwarding	549
Table 167: Context variables created by the Route via MQ Native assertion	552
Table 168: Defined MQ Headers Prefixes	553
Table 169: WSS Header Handling during MQ Native routing	555
Table 170: MQ Native Route Properties - [Request]	557
Table 171: MQ Native Route Properties - [Response]	559
Table 172: Raw TCP Routing settings	561
Table 173: Assertions in SFTP partial downloads fragment	563
Table 174: SSH2 Routing Settings [Connection] tab	565
Table 175: Command Types for the Command Selection drop down list	567
Table 176: SSH2 Routing Settings [Authentication] tab	569
Table 177: SSH2 Routing Settings [Advanced] tab	570
Table 178: Rate Limit settings	574
Table 179: Context variables created by Apply Throughput Quota assertion	579
Table 180: Throughput Quota settings	580
Table 181: Time/Day Availability settings	585
Table 182: Cache Lookup settings	586
Table 183: Context variables created by Query Rate Limit assertion	587
Table 184: Rate Limit Query settings	588
Table 185: Context variables created by Query Throughput Quota assertion	589
Table 186: Throughput Quota Query settings	590
Table 187: IP Address Range settings	593
Table 188: Cache Storage settings	595
Table 189: Interaction between cluster properties and auditing assertions	600
Table 190: Audit Detail settings	601
Table 191: Audit settings	603
Table 192: Requestor Identity settings	606
Table 193: Fault Response settings	609
Table 194: Email Alert settings	614
Table 195: SNMP Trap settings	616
Table 196: Compare Expression assertion: Data type	623
Table 197: Compare Expression assertion: Handling multivalued context variables	624
Table 198: Compare Expression assertion: Comparison rules	624
Table 199: Context variables created by the Create Routing Strategy assertion	626
Table 200: Configuring the Route List	628
Table 201: Configuring route properties	629
Table 202: Context variables created by the Execute Routing Strategy assertion	630
Table 203: Execute Routing Strategy assertion settings	631
Table 204: Generate UUID settings	635
Table 205: Join Variable settings	637
Table 206: Context variables created by the Look Up Context Variable assertion	638
Table 207: Look Up Context Variable settings	639
Table 208: Manipulate Multivalued Variable settings	643
Table 209: Map Value settings	647
Table 210: Process Routing Strategy Result Properties	649
Table 211: Feedback information for the current route	650
Table 212: Explanation of Policy Fragment for Run Assertions for Each Item assertion	653

Table 213: Context variables created by Run Assertions for Each Item assertion	655
Table 214: Run Assertions for Each Item settings	656
Table 215: Context Variable settings	658
Table 216: Split Variable settings	662
Table 217: Request Size Limit settings	669
Table 218: Code Injection Protection settings	671
Table 219: CSRF Protection settings	674
Table 220: Document Structure Threat Protection settings	676
Table 221: JSON Document Structure Threat Protection settings	679
Table 222: Message Replay Protection settings	683
Table 223: SQL Attack Protection - response to injections	684
Table 224: SQL Attack Protection settings	687
Table 225: Context variables created by the ICAP-Enabled Antivirus assertion	688
Table 226: Configuring the ICAP Antivirus server list	689
Table 227: Configure the ICAP antivirus service parameters	690
Table 228: Failover Strategies for ICAP antivirus	691
Table 229: Context variables created by the Sophos Antivirus assertion	692
Table 230: Sophos Antivirus settings	693
Table 231: Rate Limit settings	695
Table 232: Configuring the JSON schema based on location	698
Table 233: Context variables created by Validate OData Request assertion	701
Table 234: Odata Validate Settings	702
Table 235: Configuring the schema based on location	707
Table 236: Extract Schema from WSDL settings	708
Table 237: QosMetrics properties supported in Gateway	712
Table 238: Context variables created by Manage Gateway assertion	715
Table 239: Context variables used by REST Manage Gateway assertion	717
Table 240: Context variables created by Access Resource Protected by JSAM assertion	720
Table 241: Access Resource Protected by JSAM settings	721
Table 242: Context variables created by the Access Resources Protected by OAM assertion	722
Table 243: Oracle Access Manager Protected Resource settings	724
Table 244: Tivoli Access Manager assertion settings	729
Table 245: Tivoli Access Manager errors	730
Table 246: SiteMinder R12 Custom Assertion settings	732
Table 247: SiteMinder R12 errors	733
Table 248: Context variables created by the Execute Salesforce Operation assertion	735
Table 249: Configure Action - Create Objects	739
Table 250: Configure Action - Update Objects	741
Table 251: Configure Action - Retrieve Objects	743
Table 252: Configure Action - Retrieve Modified Objects	744
Table 253: Configure Action - Retrieve Deleted Objects	746
Table 254: Configure Action - Execute Query	747
Table 255: Configure Action - Search Objects	749

Chapter 1:

Working with Service Policies

The core function of the Policy Manager is its ability to centrally define, provision, monitor, and audit security and integration policies for web services and XML applications.

After a service is published (using either the Publish SOAP Web Service Wizard, Create WSDL Wizard, Publish Web API Wizard, or Publish REST Service Proxy Wizard), it appears in the Services and Policies list and an initial policy is created in the policy development window (this is called the "service policy"). If the WSDL document of a published web service contains at least one HTTP(S) binding URL, then the initial policy will include a [Route via HTTP\(S\)](#) assertion preconfigured to point to the HTTP(S) binding URL. Before a service is adequately protected, you will need to configure additional policy assertions.

In addition to the service policy, you can also create these other types of policies:

- **Global policy fragments:** These are special policies that can be configured to run automatically at certain points within any service policy, without needing to be explicitly added to the service policy. For more information, see "Working with Global Policy Fragments" on page 106.
- **Included policy fragments:** These are "boilerplate" policies that can be inserted into any service policy. They are also known as "policy fragments". For more information, see "Chapter 2: Working with Policy Fragments" on page 101.
- **Internal use policies:** These are policies associated with internal services. For more information, see Working with Internal Services in the *Layer 7 Policy Manager User Manual*.
- **Policy-Backed Identity Provider Policy Fragment:** Similar to included policy fragments, except they are intended for use as an authentication policy for Policy-Backed Identity Providers. For more information, see Policy-Backed Identity Providers in the *Layer 7 Policy Manager User Manual*.

A policy defines restrictions for the consumption of a published Gateway-protected service. To learn about the interface areas related to services and policies, see Interfaces in the *Layer 7 Policy Manager User Manual*.

Note: In this chapter, the term *identity* includes both users and groups; *user* can represent an individual human or machine; *service* includes both web services and XML applications.

Policy Organization

In the Policy Manager, a policy includes assertions that determine the authentication method, identity credentials, transport method, and routing method for the web service or XML application. The specific types of assertions, their relative location, and the other assertions determine the properties and validity of a policy.

You construct a policy by moving [assertions](#) and [policy fragments](#) into a meaningful tree structure in the policy development window. During processing, the Gateway scans each policy assertion from top to bottom, assigning a 'succeed' or 'fail' outcome to each.

The following is the message processing model for a typical policy:

1. Service request arrives.
2. Request is run through the WS-Security processor:
 - Encrypted sections are decrypted and WS-Security Signatures are verified. The sign and/or encrypt order is chosen by the sender
 - Default security header can be optionally removed before routing.
3. Request is run through the policy assertions:
 - Routing assertion sends a request to the service server
 - Remainder of policy assertions are applied to the service response.
4. Response is run through the WS-Security decorator:
 - Default security header is created
 - Signatures specified by the policy are applied
 - Encryption specified by the policy is performed.
5. Response is sent back to the client.

Special Assertions

There are two special assertions that can help you refine the policy logic:

- ["At least one assertion must evaluate to true" folder](#)

Each child assertion placed in this folder is processed until an assertion succeeds. At this point, processing of the folder stops and the "At least one" folder is assigned a successful outcome. However if all assertions in the folder fail, then the "At least one" assertion is assigned a failure outcome.

- "All assertions must evaluate to true" folder

Each child assertion placed in this folder is processed until an assertion fails. At this point, processing of the folder stops and the "All assertions" folder is assigned a failure outcome. However if all assertions in the folder succeed, then the "All assertions" folder is assigned a successful outcome.

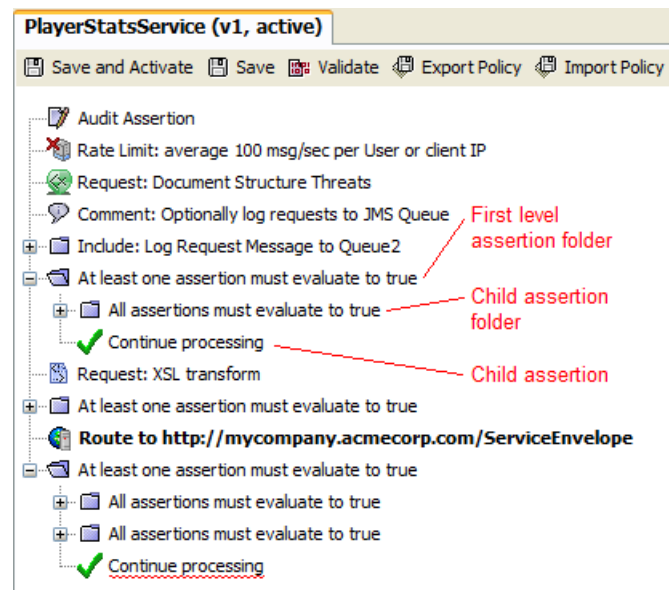


Figure 1: Sample policy assertion tree in the policy development window

All first level assertions in a policy (including first level "At least" and "All assertions" folders) must succeed in order for the overall policy to succeed. When the policy succeeds, the service requestor receives a response message. If the policy fails, the service requestor receives an error message.

Hints and Tips

- Turn on assertion line numbers to help you edit or troubleshoot a policy. You can jump to a specific line number ([**Edit**] > **Go to Assertion**) or use the search feature ([**Edit**] > **Find**) to quickly locate an assertion.
- If you do not want the failure of an "At least one assertion must evaluate to true" folder to fail the entire policy, then add a [Continue Processing](#) assertion into the assertion folder. This assertion will always evaluate to true, preventing the failure of a policy due to the failure of a non-essential or conditional assertion.

- Some policy assertions work together and require the presence of each other to succeed. For example, the [Authenticate User or Group](#) assertion requires an authentication assertion such as [Require HTTP Basic Credentials](#) to provide the credentials for validating the user's identity. Moreover, the authentication assertion must appear before the user or group. This example illustrates how the presence and order of assertions can affect the ultimate validity of a policy.
- An authentication assertion (such as [Require HTTP Basic Credentials](#)) can only provide credentials for a single user or group. The authentication assertion must appear before the identity provider assertion (Authenticate User or Group).
- It is best not to include more than one first-level identity assertion in a policy or within an "At least" or "All assertions" folder. If you must because the client expects the Gateway to authenticate more than one identity per request, the policy validator will display a warning, but you can still proceed.

To add more than one user or group in a policy, you should place each individual Authenticate User or Group assertion in a separate "[At least one assertion](#)" or "[All assertions](#)" folder with an authentication assertion (and other assertions, as required).

In Figure 2, the credentials for user "Bob" are authenticated by the Require HTTP Basic Credentials assertion, while "Sue" is authenticated by the [Require SSL or TLS Transport with Client Authentication](#) assertion.

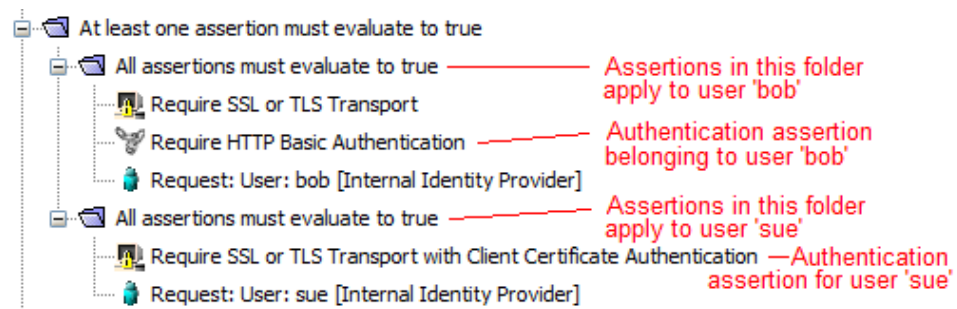


Figure 2: Policy with multiple identities

- It is best to put a [Stop Processing](#) assertion after any assertion whose sole purpose is to report on a prior error—for example, these assertions: [Audit Messages in Policy](#), [Send Email Alert](#), [Send SNMP Trap](#), or [Return Template Response to Requestor](#). Reason: These assertions always succeed, even though the intent is to halt the policy and send an HTTP challenge.

Example 1: You construct the following expecting the policy to halt if authentication fails, but in this case the routing will occur regardless:

```
Require HTTP Basic Credentials
At least one assertion must evaluate to true
  Request: Authenticate against Internal Identity Provider
```

```
Audit: "Authentication Failed!"
Route
```

Example 2: To correct Example 1 so that the policy operates as intended, add a Stop Processing assertion:

```
Require HTTP Basic Credentials
At least one assertion must evaluate to true
  Request: Authenticate against Internal Identity Provider
  All assertions must be true
    Audit: "Authentication Failed!"
    Stop Processing
Route
```

- In general, assertions should be placed *before* the routing assertion in a policy. This is to ensure that all assertion conditions are met before the request is routed to the protected web service or XML application.
- Exceptions to the above are the assertions designed to operate on the response; these should be placed *after* the routing assertion:
 - [Add Security Token](#) (with target set to 'Response')
 - [Add Timestamp](#) (with target set to 'Response')
 - [Encrypt Element](#) (with target set to 'Response')
 - [Evaluate Response XPath](#)
 - [Sign Element](#) (with target set to 'Response')
 - [Validate XML Schema](#)
 - [Apply XSL Transformation](#)
- Assertions where you can specify the target message to be acted upon will be prefixed with "Request:", "Response:", or "\${VARIABLE_NAME}" in the policy window. For example: [Request: Authenticate against XYZ](#) or [Response: Add signed Timestamp](#). For more information, see "Selecting a Target Message" on page 153.
- Pay attention to Policy Validation Messages window. It will display helpful messages as you [configure](#) or [validate](#) a policy.
- Use the Copy and Paste options on the Edit menu to help you organize the policy.
- Instead of deleting an assertion, consider [disabling](#) it instead. Disabling an assertion is useful during testing and troubleshooting. It has the same effect as deleting the assertion, but you can easily restore the assertion by re-enabling it.
- If you disable all assertions in a "All assertions..." folder, this folder will succeed. However if you disable all assertions within a "At least one..." folder, this folder will fail.

Policy Revisions

The Policy Manager can keep a revision history of changes made to a [policy](#) or [policy fragment](#). It can record when a change was made and who made it. A version number is assigned to each change. You can roll back to any version, making it the "active" policy.

Note: The policy revisions feature only tracks changes to the policy XML. It will not record changes to other objects such as users, groups, private keys, certificates, or JMS connections. It will also not include changes to other service or policy properties (for example, SOAP services intended, routing to URI or WSDL location).

Configuring Policy Revisions

By default, the Gateway is preconfigured to store 20 versions. You can change the number of versions stored by setting the `policyVersioning.maxRevisions` cluster property.

Note: A policy revision is "protected" once it is assigned a comment. This means it will never be overwritten and it does not count toward the stored revisions maximum. To remove this protection, simply delete the comment.

Creating a New Policy Revision

When policy revisions are enabled, the Policy Manager automatically creates a new revision each time you save a policy. If you wish to describe the policy, add a comment using the Policy Revisions dialog. Versions containing a comment are protected from being overwritten. Versions without a comment will be automatically overwritten when the revision limit is reached.

A new revision is created each time the policy is saved, even if no changes have been made.

Working with Policy Revisions

The title of the policy gives a concise indication of the revision in use and whether it is the active revision. Consider the following example:

Warehouse [/http] (v22/32, active)

This indicates that the policy has 32 revisions and revision 22, currently being edited, is the active revision.

The word "active" changes to "inactive" if the revision being edited is not the active revision.

➤ To view policy versions:

- In the Services and Policies list, right-click the service name or policy name and then select **Revision History**. The assertion properties are displayed.

Tip: To quickly open the active version of the policy, right-click the service name and select **Active Policy Assertions** or double-click the service name in the Services and Policies list. The word "active" will appear next to the service name and version number above the policy window to remind you that you are working with the active version.

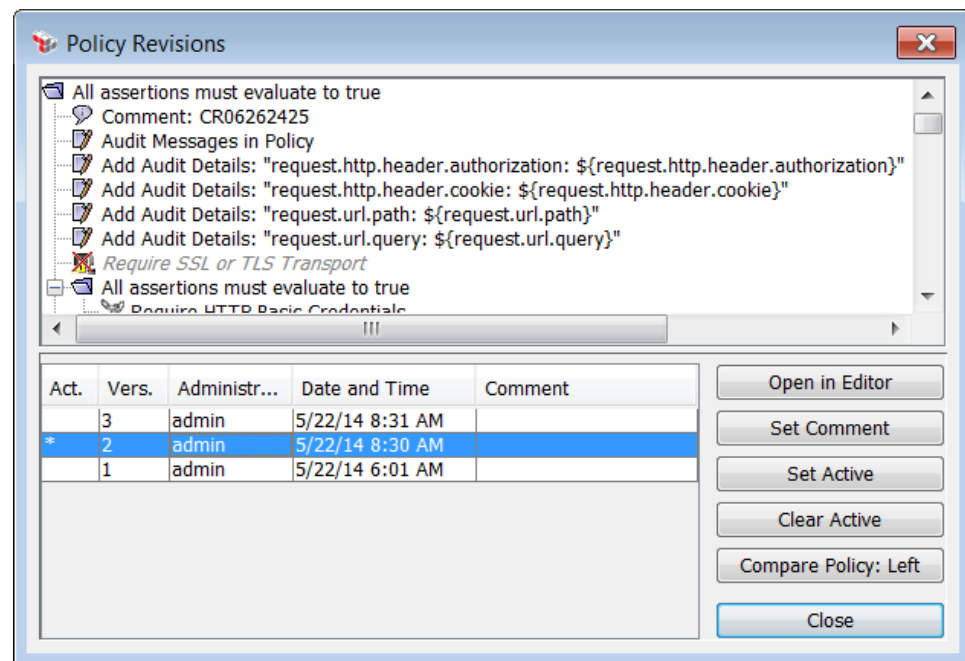




Figure 3: Policy Revisions dialog

The following table describes the dialog:

Table 1: Policy revision tasks

Element	Description
<i>Policy tree at top</i>	<p>The tree shows the policy revision currently selected. This helps you better visualize each revision.</p> <p>Tip: You can click the  and  icons next to a folder to collapse or expand the folder.</p>
Act.	Indicates the version that is currently active. It is possible to have a policy with no active versions; this disables the policy.
Vers.	<p>The version number, assigned by the Policy Manager. Version numbers may not be contiguous, but they will be unique.</p> <p>The number of versions stored is controlled by the</p>

Element	Description
	<p><i>policyVersioning.maxRevisions</i> cluster property. By default, 20 "uncommented" versions are stored.</p> <p>Tip: Once a comment is entered, the version is preserved permanently and does not count against the revision limit. For example, 30 versions may be displayed even though the maximum is 20, because 10 of those versions have comments entered. To allow a version to be discarded, simply delete the comment.</p>
Administrator	The person who was logged in at the time when the changes were saved.
Date and Time	When the version was created.
Comment	<p>A description of the version. To enter a comment, click [Set Comment]. Entering a comment is not mandatory but recommended for versions you may wish to roll back to in the future.</p> <p>Notes: (1) Revisions with comments will be protected against deletion; they will also be excluded from the maximum revisions cap. (2) Policy revisions for a policy fragment used in an encapsulated assertion may display an "Artifact Version" number. This number is automatically inserted when the encapsulated assertion configuration is imported into the Gateway. For information about the Artifact Version number, see "Encapsulated Assertion Configuration Properties" on page 134.</p>
Open in Editor	<p>Opens the currently selected version in a new tab within the policy development window for editing.</p> <p>Tip: The version number and its active status are displayed next to the service name in the tab for that version in the policy development window.</p>
Set Comment	Allows you to enter or remove a comment for the selected version.
Set Active	<p>Designates the currently selected version as the active version for the policy and loads the version into a new tab in the policy development window.</p> <p>For more information, see "Enabling a Policy" on page 24.</p> <p>Notes: (1) If the active version is currently open in the policy development window, it automatically becomes inactive. (2) Reverting to an older version of a policy may rely on objects (users, groups, WSDLs, private keys) that no longer exist. Be sure to validate the policy afterward.</p>
Clear Active	<p>Clears the active version and disables the policy. Click [Yes] to confirm when prompted. It is not necessary to select the active version before clicking [Clear Active].</p> <p>Note: If the active version is currently open in the policy development window, it automatically becomes inactive.</p>

Element	Description
	For more information, see "Disabling a Policy" on page 24.
Compare Policy: Left Right	<p>Loads the selected revision for policy comparison. The button label shows either "Left" or "Right" depending on whether this is the first or second revision or policy selected.</p> <p>Tip: Comparison begins immediately after the second policy is selected and may require a moment to complete. The results are displayed in a separate window.</p> <p>For more information, see "Comparing Policies" on page 28.</p>
Close	Closes the Policy Revisions dialog.

Policy Properties

A policy's properties are displayed when you [create a new policy](#). You can also view and edit the properties later.

Note: *Service policies* do not have properties visible on the interface. These policies are automatically created when a service is published and only one service policy may exist for a published service. For more information, see "Working with Service Policies" in the *Layer 7 Policy Manager User Manual*.

➤ To access the properties for a policy:

- Do either of the following:
 - Select **[Tasks] > Create Policy** from the Main Menu
 - Right-click a policy in the Services and Policies list and then select **Policy Properties**.








Figure 4: Policy Properties

2. Configure the properties as follows:

Table 2: Policy Properties settings

Setting	Description
Name	Enter a name for the policy. This name should readily identify the purpose of the policy (i.e., global, included, or internal). This name is displayed in the Services and Policies list and the policy assertions palette.
Policy GUID	This is the Globally Unique Identifier for the policy. It is assigned by the system and cannot be changed.
Policy ID	This is the entity ID for the policy. It is assigned by the system and cannot be changed.
Policy Type	<p>From the drop-down list, select the type of policy being created:</p> <ul style="list-style-type: none"> • Global Policy Fragment ( in the Services and Policies list) • Included Policy Fragment ( in the Services and Policies list) • Internal Use Policy ( in the Services and Policies list) • Policy-Backed Identity Provider Policy Fragment ( in the Services and Policies list) <p>For a description of each type, see "Creating a Policy" on page 21.</p> <p>Only users with the role of 'Administrator' can create a policy.</p> <p>Note: The Internal Use Policy option is available only when an internal service has been published.</p> <p>Tip: For a shortcut method to creating an Included Policy Fragment, see "Policy Fragment Shortcut" on page 102.</p>
Policy Tag	<p>This tag specifies the purpose of the policy and is used for these policy types:</p> <ul style="list-style-type: none"> • Internal Use Policy: The tag specifies which type of internal service can use the policy. Select one of the following: <ul style="list-style-type: none"> • wsdm-notification: The policy is eligible to be selected as a notification policy from the "Subscribe to WSDM Resource Assertion" on page 717. • audit-message-filter: The policy is an Audit Message Filter (AMF) policy. For more information on this type of policy, see "Working with Internal Use Policies" on page 33. • audit-viewer: The policy is an Audit Viewer (AV) policy. For more information on this type of policy, see "Working with Internal Use Policies" on page 33. • Global Policy Fragment: This tag indicates when the global

Setting	Description
	<p>policy fragment should be executed. Select one of the following:</p> <ul style="list-style-type: none"> • message-received: Global policy runs on receipt of a message before service resolution • pre-security: Global policy runs before (request) security undecoration • pre-service: Global policy runs before the service policy • post-service: Global policy runs after the service policy • post-security: Global policy runs after (response) security decoration • message-completed: Global policy runs when processing for a message completes (even on policy failure/exception, service not resolved, etc) <p>The policy tags will be evaluated in the order shown above. For more information about global policies, see "Working with Global Policy Fragments" on page 106.</p> <p>Tip: The policy tag is displayed next to the policy name in the Services & Policies list on the Policy Manager interface.</p>
Intended for SOAP services	<p>Indicates whether the policy can be used in SOAP-only policies or in both SOAP and non-SOAP policies:</p> <ul style="list-style-type: none"> • If the policy will contain assertions that require SOAP, select the Intended for SOAP services check box. The policy validator will issue a warning if anyone attempts to use the policy in a non-SOAP policy. Selecting this check box does <i>not</i> enforce the presence of SOAP-only assertions in the policy (in other words, the validator will not alert you if you have not added a SOAP-only assertion to the policy). For more information, see "Assertions that Require SOAP" under "Working with Non-XML Messages" on page 37. <p>Note: For SOAP-only assertions, an error occurs only when a non-SOAP <i>request</i> is received.</p> <ul style="list-style-type: none"> • If the policy is intended for use in both SOAP and non-SOAP policies, clear this check box. The policy validator will issue a warning if you attempt to add a SOAP-only assertion to the policy. <p>Note: The Intended for SOAP services check box only controls the validator warnings. It does <i>not</i> affect how the policy functions at policy runtime. Normal policy logic still applies.</p>
Security Zone (for "Included Policy Fragment" only)	<p>Optionally choose a security zone. To remove this entity from a security zone (security role permitting), choose "No security zone".</p> <p>For more information about security zones, see Understanding Security Zones in the <i>Layer 7 Policy Manager User Manual</i>.</p>

Setting	Description
	<p>Note: This control is hidden if either: (a) no security zones have been defined, or (b) you do not have Read access to any security zone (regardless of whether you have Read access to entities inside the zones).</p> <p>Tip: A policy may allow entities that are not members of its security zone to be added and edited, but validation will prevent unpermitted entities to be saved. For example, it is possible to paste policy XML for assertions that are not part a policy's security zone. While authoring, you are not prevented from editing these assertions. However these unpermitted assertions will be detected during policy validation when you attempt to save the policy.</p>

3. Click **[OK]**.

Organizing Services and Policies into Folders

In the Services and Policies list, you can create folders to help you organize your services and policies, and to control access to them. Only users with a role of *Administrator* or *Manage Web Services* can create and manipulate folders. For more information, see *Predefined Roles and Permissions* in the *Layer 7 Policy Manager User Manual*.

Controlling Access Using Folders

In addition to the organizational benefits provided by folders, controlling access to your services and policies is simplified too. Each time a folder is created, the Policy Manager automatically creates two corresponding folder roles:

- **Manage <folderName> Folder:** This role allows a user to create, read, update, and delete services or policies within the folder, including nested sub folders. If aliases are present in the folder, permission to read, update, or delete an alias is granted only if:
 - the user is assigned to a role that has access to the original entity, AND
 - the user has the "Manage Folder" role
- **View <folderName> Folder:** This role only allows a user to view entities within the folder, including the contents of nested sub folders. If aliases are present in the folder, the original entity may be modified but not deleted only if:
 - the user is assigned to a role that has access to the original entity, AND
 - the user has the "View Folder" role

Folder roles let you grant access to many policies to a user via a single role assignment. Once a folder role is assigned, the user is granted access to all services/policies in the folder, including services/policies contained in sub folders.

If a user has not been granted any folder roles, yet has permission to a service or policy nested within several sub folders, that user will be able to see all folder names between the root and the service's parent folder. However, all other folder content will not be visible.

For more information, see Managing Roles in the *Layer 7 Policy Manager User Manual*.

Refining Access with Security Zones

In addition to the two folder-based security roles mentioned above, you can also place folders into security zones to further refine access. Users with either the corresponding "Manage X Zone" or "View X Zone" roles will be able to view the items in the folders that are also in the X zone. Users *without* either of these roles but who have a "Manage X Folder" or "View X Folder" folder will be able to view the folder's content. User with none of these roles will not have access to the folder at all.

For more information, see Understanding Security Zones in the *Layer 7 Policy Manager User Manual*.

➤ To create a folder:

1. Right-click any folder or the root node and then select **Create New Folder**.
2. Enter a name for the new folder. The new folder is created as a subfolder within the chosen folder. You can create up to 8 levels of folders.

Tip: Ensure that all your folders have unique names, to avoid potential problems with roles and permissions.

3. Optionally choose a security zone. To remove this entity from a security zone (security role permitting), choose "**No security zone**". For more information about security zones, see Understanding Security Zones in the *Layer 7 Policy Manager User Manual*. **Note:** This control is hidden if either: (a) no security zones have been defined, or (b) you do not have Read access to any security zone (regardless of whether you have Read access to entities inside the zones).
4. Click [**OK**] when done.

➤ *To delete a folder:*

1. Ensure that no services or policies in the folder being deleted are still in use. You can delete a non-empty folder containing items that are no longer referenced elsewhere.
2. Right-click the folder and then select **Delete Folder**.

➤ *To change a folder's name or security zone:*

1. Right-click the folder and then select **Folder Properties**.
2. Modify the name, if necessary
3. Choose another security zone, if necessary.
4. Click **[OK]**.

Tip: Renaming a folder or changing its security zone automatically updates its associated folder role. For more information, see *Predefined Roles and Permissions in the Layer 7 Policy Manager User Manual*.

➤ *To move a service, policy, or folder:*

- Drag and drop the item from one folder to another.
- Or:
1. Right-click the item to move and then select **Cut ...**
 2. Right-click the destination folder and then select **Paste ...**

Tips: (1) You can move multiple items at once by holding down the **[Ctrl]** key to select the items before performing a drag and drop or cut and paste. (2) When moving services or policies between two folders (regardless of security zones), you must have Update permission on the entity (i.e., service or policy) and Update permission for the source and destination folders.

➤ *To search for a service or policy:*

- See "Quick Search" in Services and Policies in the *Layer 7 Policy Manager User Manual*.

Authentication in a Policy

How a user is authenticated in a service policy is a two step process:

- First, the credentials are collected.
- Next, the credentials are authenticated against an identity provider.

There are numerous mechanisms for gathering credentials, most of which are based upon industry standards. The chart in Figure 5 illustrates the different standards used.

Note: The mechanisms shown in red send passwords in the clear and should be sent over SSL. However, mechanisms shown in green are considered secure.

	Username & Password	X509 Certificate	Kerberos Token	SAML Token
Transport Layer	HTTP Basic Auth	Mutual SSL	Windows Integrated	
Message Layer	WSS Username Token Profile	WSS X509 Binary Security Token Profile	WSS Kerberos Token Profile	WSS SAML Token Profile
	Encrypted Username Token Profile			
	XPath Credentials			

Figure 5: Mechanisms for gathering credentials

Working with Aliases

An *alias* allows a service or [policy](#) to appear in more than one folder in the Services and Policies list. The alias is a linked copy of the original policy or service: all changes made to the alias are reflected in the original; changes to the original are automatically reflected in all aliases. Aliases help you organize your services and policies. User can modify a service alias or policy alias if they have the correct permissions.

Note the following if you choose to use aliases:

- An entity may have multiple aliases.
- The name of an alias is derived from the original and cannot be changed.
- Deleting an original removes all its aliases; deleting an alias does not affect the original or other aliases. Deleting an alias is possible only if you have delete privileges to the parent folder of the alias.
- Access to an alias depends on your access to the alias' parent [folder](#) and access to the original entity. Privileges will vary depending on whether your role for the parent folder is *View Folder* or *Manage Folder* (the 'Manage' role permits deletion of aliases, while 'View' role does not; both roles permit modifying the alias). Access to the original does not imply access to any of its aliases. For more information, see [Organizing Services and Policies into Folders](#) in the *Layer 7 Policy Manager User Manual*.

- Access to an alias also requires a role assignment that grants access to original service or policy. For more information, see Managing Roles in the *Layer 7 Policy Manager User Manual*.
- Folders cannot have aliases.

Note: Only users with a role of *Administrator* or *Manage Web Services* can create, delete, or view aliases. Other users will not see the aliases.

Security Zones for Aliases

If the source service or policy has been placed in a security zone, any aliases created will inherit that zone by default. However the original and alias may have independent security zones. This means security zone changes to the original will not affect the alias and vice-versa.

Aliases will always be visible provided you have access to the alias' owning policy/service and to the folder containing the alias. This holds true even after an entity type is removed from a security zone. For example, the Policy Alias entity type is removed from the "Test" zone. Three aliases have already been assigned to the "Test" zone and you have the Manage Test Zone role. After the removal, you can still see the three aliases, though you can no longer change their security zone since they are no longer zoned for your role.

To learn more about security zones, see Understanding Security Zones in the *Layer 7 Policy Manager User Manual*.

➤ *To create an alias:*

1. In the Services and Policies list, right-click a service/policy and select **Copy as Alias**.

Tip: You can select multiple services or policies by holding down the [Ctrl] key while selecting.

2. Create a destination folder for the alias, if necessary. For more information, see "Organizing Services and Policies into Folders" on page 12.
3. Right-click on the destination folder and then select **Paste as Alias**. The alias is added. The icon for the alias contains an 'a' and the word 'alias' is added to the policy name to remind you that this is not the original.

Note: You cannot create an alias in the same folder as the original.

➤ *To delete an alias:*

1. In the Services and Policies list, right-click the alias to remove and then select **Delete Service Alias** or **Delete Policy Alias**.
2. Click **[Yes]** to confirm. The alias is removed.

Tip: The ability to delete an alias depends on the user having the 'Manage Folder' role. If you can see an alias, it means that you have been assigned the role for the original service or policy.

➤ *To change the security zone for an alias:*

1. In the Services and Policies list, right-click the alias to change and then select **Security Zone**.
2. Choose the new security zone from the drop-down list and then click **[OK]**.

Working with Multiple Signatures

The Gateway can create or validate multiple signatures in a message with multiple identities involved.

Note: The Securespan XML VPN Client does not support multiple signatures in a message. Service consumption will always fail when there are multiple signatures in the response/request.

When multiple signatures are in use, there is more than one identity responsible for the contents of a message. A policy must be constructed in a way to indicate which identity is responsible for signing the various parts of a message. The signing identities may originate from different identity providers, for example:

At least one assertion must evaluate to true:
 User: Alice [Internal Identity Provider]
 User: Bob [Internal Identity Provider]
 Member of Group: Service Users [My Federated Provider]

In the example above, the policy is indicating that any of the identities ("Alice", "Bob", or "Service Users") are permitted as the signing identity.

There may be instances where it is not possible to distinguish between multiple signing identities, or when one of the identities does not correspond to an existing Group or Identity Provider. In this case, identity tagging can be used during authentication:

At least one assertion must evaluate to true:
 User: Alice [Internal Identity Provider] as "user"
 User: Bob [Internal Identity Provider] as "user"
 Member of Group: Service Users [My Federated Provider] as "user"

In the example above, the identity tag is "user". Here is another example:

`Authenticate against: My Federated Provider as "identity1"`
`Authenticate against: My Federated Provider as "identity2"`

Where "identity1" and "identity2" are the identity tags. For more information on using identity tags, see Identity Tags.

How to Permit Multiple Signatures

To permit multiple X.509 signatures in a policy, you must select the **Allow multiple signatures** check box in the "Require WS-Security Signature Credentials Assertion" on page 246:

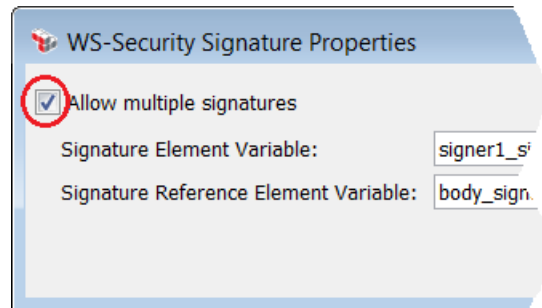


Figure 6: Enabling multiple signatures in a policy

You will also need to set the cluster property `wss.processor.allowMultipleTimestampSignatures` to "true".

Note: The Require WS-Security Signature Credentials assertion will fail if a message has multiple X.509 signatures but the Allow multiple signatures check box is not selected.

Configuring a Policy

In the Policy Manager, you construct a policy for a published service using these four general steps:

1. Select the service for the policy.
2. [Add](#) the assertions to the policy development window and configure as necessary. Refer to the documentation for specific assertions for configuration instructions. You can use [policy fragments](#) to help maintain consistency and enforce global rules across different services.
3. [Organize](#) the assertions into a logical tree-structure that conforms to the policy

and assertions rules..

4. [Validate](#) the policy.

A logical policy must be in place for each published service in the Policy Manager. In addition to manually constructing a policy, you can also:

- Import a policy from a [local file](#)
- Import a policy from a [UDDI registry](#)

Tip: When configuring a policy for the first time on a newly published service, you may want to disable the service while constructing the policy. This removes any possibility of unsecured and unauthorized access. A disabled service contains a red "X" through its icon.

➤ *To configure a policy:*

1. Access the policy development window for the target service by one of two methods:
 - Right-click the service name in the Services and Policies list and then select **Active Policy Assertions**
 - Double-click the service name in the Services and Policies list.

The policy development window appears.




2. [Add](#) assertions to the policy and configure if necessary. Some assertions require configuration immediately, while others have adequate defaults such that additional configuration may not be required. Some assertions do not require configuring at all.

When you publish a service using the Publish SOAP Web Service Wizard, Create WSDL Wizard, or Publish Web API Wizard, the Policy Manager automatically adds the service URL specified during the publication process as an [Route via HTTP\(S\)](#) assertion in the policy development window.

Note: Certain assertions in the Policy Manager expect an XML payload and will fail on non-XML messages. If you expect your published service to handle both XML and non-XML messages, then you should structure the policy to branch accordingly. For more information, see [Working with Non-XML Messages](#).

3. Organize the assertions into a logical structure as follows:

Table 3: Organizing assertions in the policy development window

Action	Description
Move an assertion up	<p>Do any of the following:</p> <ul style="list-style-type: none"> • Select the assertion, then click  on the Assertions Tool Bar • Right-click the assertion and then select Move Assertion Up • Drag and drop the assertion to its new position <p>Tip: You can move several assertions at once by using [Ctrl]-click to select them first.</p>
Move an assertion down	<p>Do any of the following:</p> <ul style="list-style-type: none"> • Select the assertion, then click  on the Assertions Tool Bar • Right-click the assertion and then select Move Assertion Down • Drag and drop the assertion to its new position <p>Tip: You can move several assertions at once by using [Ctrl]-click to select them first.</p>
Remove an assertion	<p>Do any of the following:</p> <ul style="list-style-type: none"> • Select the assertion, then click  on the Assertions Tool Bar. Click Yes to confirm. • Right-click the assertion and then select Delete Assertion
Add an "All" or "One or more" folder	<p>Organize the assertions into an "At least one assertion must evaluate to true" or "All assertions must evaluate to true" assertion folder in the policy development window. See "Policy Organization" for examples.</p>
Edit an assertion's properties	<p>This applies only if an assertion has editable properties. Do one of the following:</p> <ul style="list-style-type: none"> • Press [Enter] while a single assertion is selected in the policy development window. • Double-click an assertion in the policy development window. • Right-click the assertion in the policy development window and select "<assertion name> Properties".

Some additional tips to keep in mind:

- Use the feedback messages in the Policy Validation Messages window to help you construct your policy. See "Validating a Policy" on page 25 for more information.
- You can also use Copy and Paste in the Edit menu to organize the assertions.
- Instead of deleting an assertion, consider [disabling](#) it instead. Disabling an assertion is useful during testing and troubleshooting.

4. **Validate** the policy one more time. If no issues remain, you may now enable the service.

Creating a Policy

The *Create Policy* task is used to create these types of policies:

- **Global policy fragments:** These are policies that are always applied before or after every [service policy](#) in the system. Only Administrators can create global policies. For more information, see "Working with Global Policy Fragments" on page 106.
- **Included Policy Fragments:** These are fragments that group any number of [assertions](#) into a self-contained unit that can be dropped into any [service policy](#). For more information, see "Chapter 2: Working with Policy Fragments" on page 101.

Tip: For a shortcut method to creating an Included Policy Fragment, see "Policy Fragment Shortcut" on page 102.

- **Policy-Backed Identity Provider Policy Fragment:** These are fragments specifically intended for use with Policy-Backed Identity Providers. For more information, see Policy-Backed Identity Providers in the *Layer 7 Policy Manager User Manual*.
- **Internal use policies:** These are ready-made policies predefined in the CA API Gateway. These policies are designed to achieve a specific objective. For more information, see "Working with Internal Use Policies" on page 33.

There are several other types of policies that are not created via the Create Policy task:




- **Audit sink policy:** This is a special policy that is created when auditing to a policy is enabled. This policy may be edited, but it cannot be renamed nor deleted. For more information, see Managing Audit Sinks in the *Layer 7 Policy Manager User Manual*.
- **Debug trace policy:** This is a special trace policy to help you troubleshoot a service policy. For more information, see "Working with the Debug Trace Policy" on page 67.

Only users with the role of "Administrator" can create a policy.

➤ *To create a policy:*

1. Do either of the following:

- Select [Tasks] > **Create Policy** from the Main Menu
 - Right-click a [folder](#) within the Services and Policies list and then select **Create Policy**.
2. Complete the properties for the type of policy that you wish to create. For more information, see "Policy Properties" on page 9.
 3. Click [OK]. The new policy is created and loaded in the policy window for editing. If you currently have unsaved changes in the policy window, you are prompted to save before the new policy is loaded. New policies have the following default assertions:
 - **For included policy fragments:** An [Add Audit Detail](#) assertion that logs the creation of the new fragment.
 - **For internal use policies:** See [Working with Internal Use Policies](#) in the *Layer 7 Policy Manager User Manual* for details.

Tips: (1) The icon color in the Services and Policies list help you readily identify the type of policy:  = Global policy fragment;  = Included policy fragment;  = Internal policy. For global and internal policies, the [policy tag](#) is displayed next to the policy name. (2) If security zones have been deployed and you have been assigned a "Manage X Zone" role, the security zone 'X' must include the "[All assertions must...](#)" composite assertion as well as every assertion in the policy (or that will be added to the policy) before you can create or edit the policy.

Editing a Service Policy

➤ *To edit a service policy:*

1. Open the policy to edit for the target service using either of the following methods:
 - Right-click the service name in the Services and Policies list and then select **Active Policy Assertions**.
 - Double-click the service name in the Services and Policies list.

The [active version of the policy](#) is open for editing in a [new tab](#). **Tip:** To edit an inactive version, right-click the service name and select **Revision History** instead.

2. In the policy development window:

- Reorganize the assertions as necessary (see step 3 in [Configuring a Policy](#)).
 - Modify assertions as necessary.
 - Add or remove [policy fragments](#) as necessary.
3. [Validate](#) the policy. This is important because changing assertions within a policy may affect the validity of the policy.

Deleting a Policy

There are several ways to delete a [policy](#) in the Policy Manager:

- For included policies (i.e., policy fragments), you can delete it directly from the service policy. This removes the policy and all its assertions from that one service policy only.
- For all policy types, you can delete it from Policy Manager. This removes it from the Services and Policies list and makes it unavailable for use in any service policy.

Deletions may take up to 15 seconds to take effect. **Tip:** Consider [disabling a policy](#) instead if you think you may need it again in the future.

Note: You cannot delete a *service policy* unless you first delete its associated published service. For more information, see *Deleting a Published Service* in the *Layer 7 Policy Manager User Manual*.

➤ *To delete an included policy from a service policy:*

- Delete the "**Include:** <fragment name>" assertion from the policy. For more information, see "Deleting an Assertion" on page 119. You cannot delete individual assertions within the fragment; you must delete the entire fragment. **Tip:** To remove individual assertions within a policy fragment, you should [edit](#) the fragment instead.

➤ *To delete a policy from the Policy Manager:*

1. Right-click the policy icon in the Services and Policies list and then select **Delete**.
2. Click **Yes** to confirm.

Notes: (1) You cannot delete a policy that is still in use in any service. (2) You can delete only one policy at a time. If more than one policy icon is selected in the Services and Policies list, only the first will be deleted.

Disabling a Policy

You can temporarily disable a policy to prevent it from executing. A disabled policy will behave as follows depending on the policy type:

- A disabled non-global policy fragment (for example, [service policy](#) or [policy fragment](#)) will always fail.
- A disabled [global policy fragment](#) will not be used—it is as though the global fragment was deleted.

A disabled policy can be [re-enabled](#) at any time.

➤ *To disable a policy in the Policy Manager:*

1. Right-click the policy icon in the Services and Policies list and then select **Revision History**. The [Policy Revisions](#) dialog appears.
2. Click [**Clear Active**] to revoke the active revision. It is not necessary to select the active revision first.
3. Click [**OK**] to confirm that you wish to disable the policy.

The policy is now disabled: the Policy Revisions dialog shows "Policy disabled - no active version" and "(inactive)" is displayed next to the policy name above the policy development window.

Enabling a Policy

A disabled policy is one that has had its active revision revoked. You can enable a policy by selecting an active revision or by starting with a new revision.

➤ *To enable a policy, do any of the following:*

- Double-click the policy name in the Services and Policies list and then select a revision to activate:

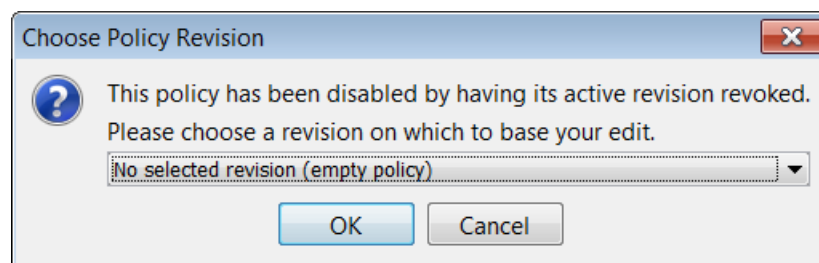


Figure 7: Choosing a revision to activate when enabling a policy

Tip: Instead of choosing a revision, you can select **"Start from an empty policy"** to create a new policy revision and set it as the active revision.

- Right-click the policy name in the Services and Policies list and then select **Active Policy Assertions** from the Choose Starting Revision dialog as shown above.
- Right-click the policy name in the Services and Policies list and then select **Revision History**. Select a revision to activate and then click **[Set Active]**. For more information, see "Policy Revisions" on page 6.

Validating a Policy

The Policy Manager provides two types of policy validation:

- Instant feedback messages in the Policy Validation Messages window when you [configure](#) or [edit](#) a policy
- Final policy validation when **[Save]** or **[Validate]** is clicked on the Policy Tool Bar. A final policy validation is more thorough than an instant feedback message, as it queries the server.

Ensure that the Policy Validation Message window is visible and that policy validation feedback has not been disabled in the Preferences. [Disabled](#) assertions in a policy are ignored during validation.

Note: A service with an invalid policy is still active, enabled, and accessible. Only a disabled service is inactive and inaccessible. See *Service Properties* in the *Layer 7 Policy Manager User Manual* for information on disabling a service.

Instant Feedback Messages

The Policy Validation Messages window displays valuable feedback messages during policy configuration or editing. The window displays the following types of messages:

Table 4: Instant feedback message types

Message Type	Description
Confirmation	Confirmation messages confirm that an assertion has been properly added and configured.
Warning	Warning messages warn that an assertion: <ul style="list-style-type: none"> • Has been duplicated • Requires another assertion to be valid • Has been configured improperly

Message Type	Description
Error	Error messages warn that an assertion: <ul style="list-style-type: none"> Has not been configured Is located improperly within the policy Is in conflict with another assertion

Note: The instant feedback messages are not designed to detect all possible errors. For example, it will not report on non-existent identities in an imported policy. Always perform a final policy validation and carefully review any messages.

Final Policy Validation Messages

The Policy Manager performs additional validation checks when you click **[Save]** or **[Validate]** on the Policy Tool Bar, or select **[File] > Validate** from the Main Menu.

- If the policy contains no errors, the message "Policy validated ok" is displayed. You may now enable or [export](#) the policy.
- If error messages appear, refer to the following table to troubleshoot.

Table 5: Troubleshooting final policy validation errors

Message Type	Suggested Solution
Assertion organization errors	Assertions must be placed in a logical order, and must be valid at the time of validation. Common policy errors include: <ul style="list-style-type: none"> Assertions out of order Assertion dependencies not established User/group IP referenced in the policy are no longer valid Reorganize the assertions so that they conform to the organization and rules described in "Policy Organization" on page 2.
Assertion configuration errors	Double check the settings of each assertion in the policy. Refer to the documentation for each assertion for more details.
Assertion permission errors	The validation message "Permission is denied for this assertion. The policy cannot be saved." indicates that the policy contains assertions that are not permitted for the policy. This could be caused by the assertions belonging to a different security zone from the policy or the user does not have sufficient permissions to save the assertions.
Identity errors	See " Invalid Users or Groups " below for more details. An example of such an error: "The corresponding identity cannot be found. Please remove the assertion from the policy."
JMS warning messages	See " Invalid JMS Queue " below for more details. An example of such an error: "The assertion might not work as

Message Type	Suggested Solution
	<i>configured. There is no protected service JMS queue defined. "</i>
Save errors	The message "Error saving service and policy" appears. This error can occur if multiple Policy Managers modify the same policy simultaneously, resulting in conflicts. If this happens, dismiss the error message, refresh the service policy, modify the policy further if necessary, then try saving again.
Namespace errors	<p>A message similar to the following appears: "Assertion: <name> Warning: This assertion contains an XPath that uses the SOAP 1.1 envelope namespace URI, but the service is configured as using only SOAP 1.2. The XPath will always fail."</p> <p>To correct this, click the Fix It link next to the message. This will allow you to update the namespaces for all your XPath assertions. For more information, see "Migrating Namespaces" on page 158.</p>

Continue validating the policy until no errors remain. At this point, you may enable or [export](#) the policy. If you require assistance troubleshooting the policy, contact CA Technical Support.

Invalid Users or Groups

You will receive an identity error during policy validation if either of the following occurs:

- An Internal Identity Provider user in the policy has an expired account
- The user or group in the policy has been deleted from the LDAP Identity Provider, Federated Identity Provider, or Internal Identity Provider.

To resolve these errors, refer to the following table:

Table 6: Resolving identity errors

Resolution	Steps
Identity no longer required	<ul style="list-style-type: none"> • Delete the assertion.
Need to keep the user or group	<ol style="list-style-type: none"> 1. Delete the assertion. 2. Set a new account expiration date for the user (see Internal Identity Provider Users and Group) <p>OR:</p> <p>Re-add the user or group to the appropriate Internal Identity Provider Users and Groups, Federated Identity Provider Users and Groups, or LDAP Identity Provider. Note: Since LDAP Identity Provider users and groups are defined outside of the Policy Manager, use the appropriate external management</p>

Resolution	Steps
	<p>program to re-add the missing LDAP Identity Provider user or group.</p> <ol style="list-style-type: none"> 3. Use the Authenticate User or Group assertion to re-add the user or group into the policy. 4. Click [Validate] or [Save] on the Policy Tool Bar to perform the final validation check.

When constructing a new policy, you will not be able to add a user or group that is not in the target identity provider.

Invalid JMS Queue

When JMS routing is used in a new or existing policy, the validation process checks the outbound queue attached to the [Route via JMS](#) assertion. An error message will appear in the Policy Validation Messages window if the queue is unspecified or invalid. Try the following steps if errors occur:

1. Enter or re-enter the outbound JMS queue.
2. Test the outbound JMS queue.
3. Edit the JMS Routing assertion, if necessary, for the new or revised queue.

Comparing Policies

The policy compare tool in the Policy Manager lets you compare any two policies. The summary appears in a separate dual-pane window that shows which assertions were added/removed or changed between the policies. Use this feature to compare any two policy versions or two completely different policies. The comparison results are shown in different colors, allowing you to see differences at a glance.

You can view assertion differences two ways: a high level summary listing the property changes, or a low level view showing the raw XML differences.

➤ *To compare policies:*

1. Choose the first policy to be compared, using any of the following methods. This will be added to the left pane of the results windows:
 - *If the policy is already open in the editor:* Right-click the policy name in the tab title and then select **Compare Policy: Left**.

- *If the policy is not yet open:* Right-click the policy in the services and policies list and then select **Compare Policy: Left**. Desktop client users can also select **File > Compare Policy: Left**.
 - *If you are currently browsing policy revisions:* Select the revision in the [Policy Revision](#) dialog and then click **Compare Policy: Left**.
2. Repeat this to choose the second policy (label now reads **Compare Policy: Right**). Allow a moment for the policy comparison to complete. Note that complex, dissimilar policies will take longer to complete. The Policy Comparison window is displayed when the comparison is complete.

Tip: If you decide not to proceed with a comparison or if you decide you want a different left pane policy, simply choose the same policy and then close the resulting comparison window.

Using the Policy Comparison Window

The Policy Comparison window displays your left and right policies and uses color coding to illustrate the differences.

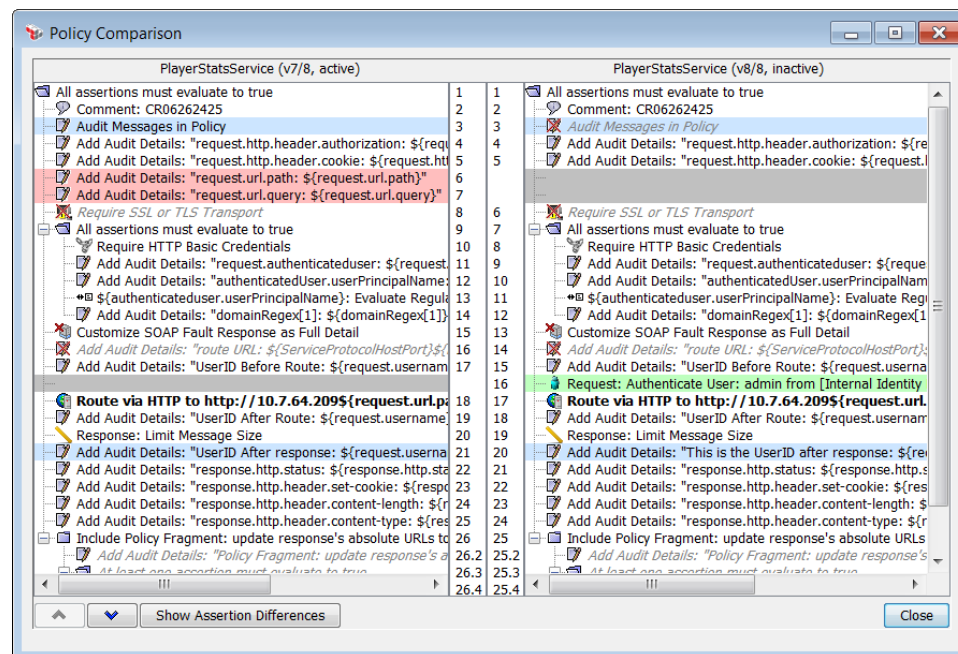


Figure 8: The Policy Diff dialog

The following color coding scheme is used:


- **Red:** Assertions in the left policy that are not present in the right policy. This may indicate assertions added to the left or deleted from the right.

- **Green:** Assertions in the right policy that are not present in the left policy. This may indicate assertions added to the right or deleted from the left.
- **Gray:** Shows where the assertions are missing as compared to the other pane.
- **Blue:** Matching assertions; assertions with the same names but their properties differ.

Assertions with no color highlighting are the same in both panes.

Tips and Hints

The following are some tips for using the Policy Comparison window:

Use the  and  buttons to jump to the next/previous difference.

- For matching assertions that differ (highlighted in blue), you can view the differences in greater detail (see "[Viewing Assertion Differences](#)" below).
- Maximize the window to see your policies more easily.
- Scrolling is synchronized between the two panes.
- Use the line numbers to help you reference assertions.
- The policy name, revision number, and active status is displayed above each pane.
Note: If the policy name is too long to display (more than half the width of the result window), it is truncated. However the full name will be visible in the tooltip that appears when you point at the policy name.
- Copy selected assertions from either pane into any open policy by using the standard Copy and Paste commands.

Viewing Assertion Differences

For assertions that are highlighted in blue (meaning assertions with the same name which exist in both policies, but their configurations differ), you can view the differences using any of these methods:

- Select the assertion (from either pane) and then click **[Show Assertion Differences]**. This opens another two-tab pane at the bottom of the window.
- Double-click the assertion (from either pane). This displays the same information as above, but in a separate Assertion Comparison window.
- Right-click the assertion and then select **Compare Assertions**. As above, this opens a separate Assertion Comparison window.

Assertion Properties

The [Properties] tab displays a high level summary of the differences between the two assertions:

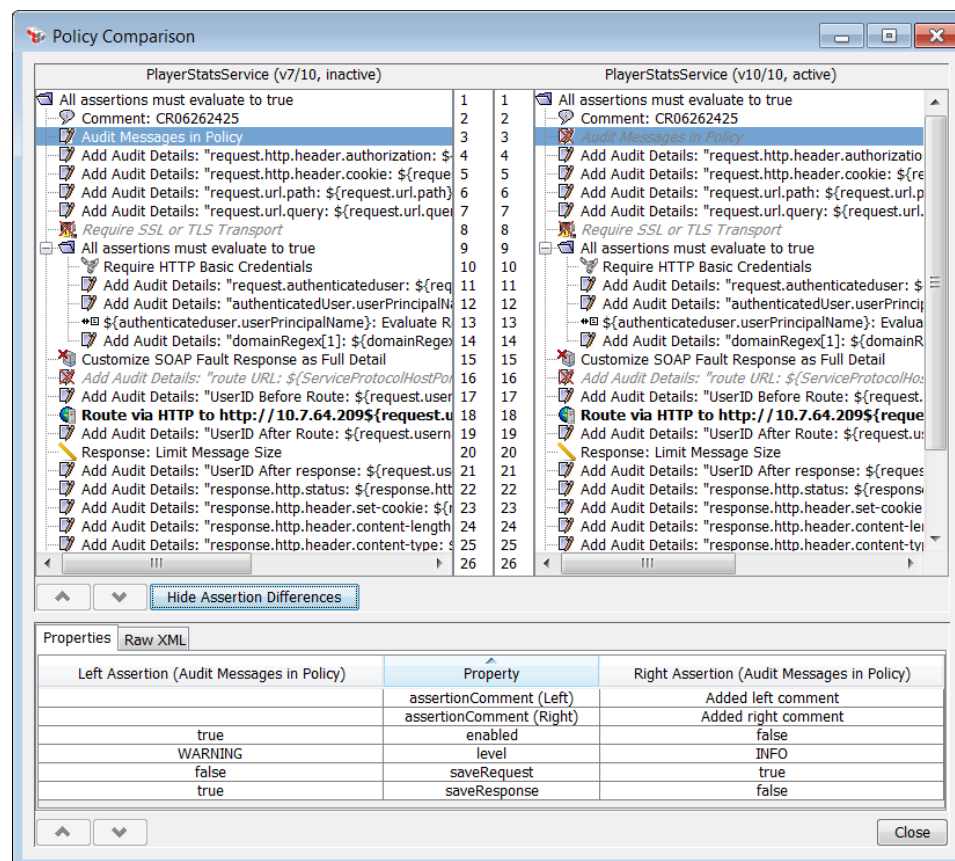


Figure 9: Policy Diff window - Properties tab

The middle column lists the properties of the assertion, while the left and right columns show the values of the properties in each assertion. **Note:** The properties listed use internal system nomenclature, which may differ from the labels found in the interface.

The [Properties] tab is useful to see the differences "at a glance", however it is not possible to display all differences due to the complexity of certain object types. The [Raw XML] tab will show complete assertion differences.

Raw XML Properties

The [Raw XML] tab shows the low level XML code for each assertion, allowing you to see precisely where the differences occur.

Tip: The Raw XML view is designed for advanced users familiar with interpreting XML code. For a more easily interpreted high level summary of the differences, use the [Properties] tab instead.

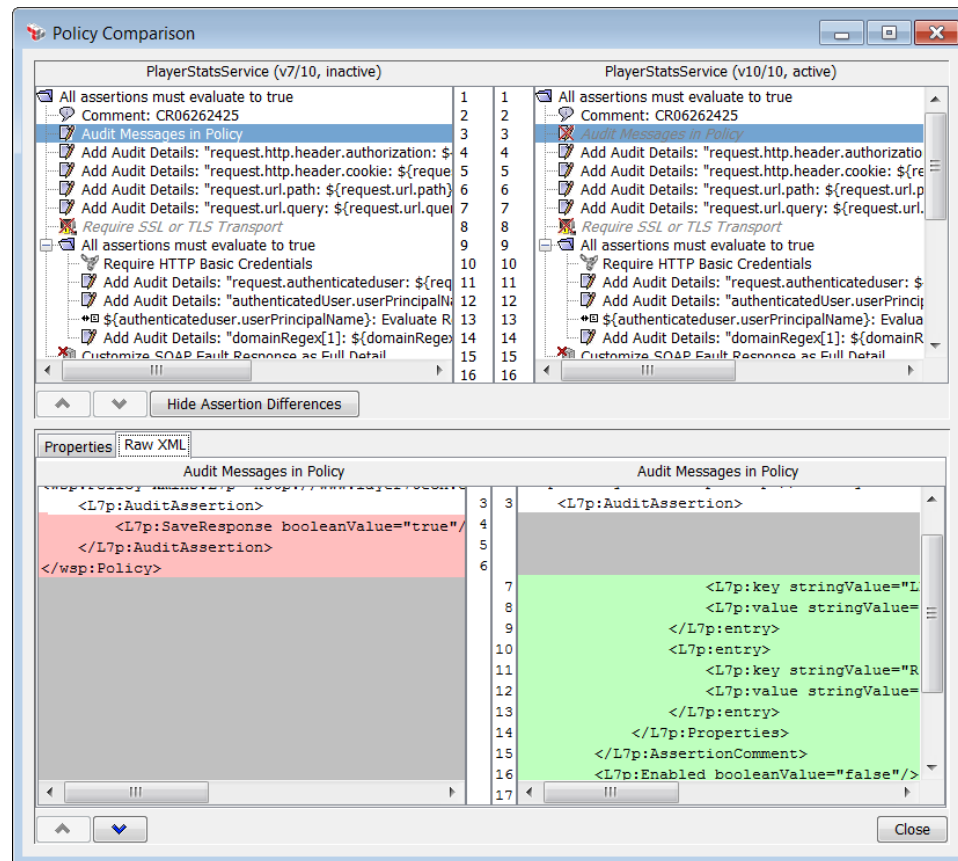


Figure 10: Policy Diff window - Raw XML tab

The [Raw XML] tab has its own  and  buttons at the bottom to jump to the next/previous difference. The same color coding is used as in Figure 8.

Working with Multiple Policy Tabs

When you open a [policy version](#) for editing, it is displayed in its own tab within the policy development window. Once you have reached the maximum number of tabs (set in the Preferences dialog), the Policy Manager automatically closes the oldest tab for which there are no unsaved changes. If there are no unsaved tabs, you are prompted to manually close some tabs.

To close a tab, do either of the following:

- Click the 'x' to the right of the tab title.
- Right-click the tab and select one of the close tab commands. These close commands mimic those used in Web browsers. Note that the Reopen Closed Tab command will not work if the policy revision from the closed tab no longer exists.

The following are additional hints and tips relating to policy tabs:

- You can choose how tabs should be displayed once you reach the browser's width: either wrap onto multiple rows (default) or maintain a single row that requires scrolling. This is set in the Preferences dialog.
- You can define how many tabs can be open at once in the policy editor workspace (maximum 100). When the maximum is reached, the Policy Manager will automatically close the least recently-used tab with no unsaved changes. This is set in the Preferences dialog.
- The Policy Manager remembers the open tabs when you disconnect (or when timeout occurs) and will reopen them for you the next time.
- The Policy Manager automatically updates all open tabs if the service name is changed or when a policy's active status is changed.
- When a service is deleted, all tabs related to that service are closed, regardless of whether there are unsaved changes.
- An asterisk (*) appears next to the names of the tabs with unsaved changes.
- Closing the last tab will result in a blank panel. Click **[Home]** in the main tool bar to return to the home screen.
- Tab titles too long to display are truncated on the interface, but will appear in full in a tooltip when you point at the tab.
- Each tab maintains its own settings for: (1) settings from the policy tool bar: Show Comments, Show Assertion Numbers, (2) all settings for the policy search bar, and (3) position of the dividing line between the policy development window and the Policy Validation Messages pane.

Working with Internal Use Policies

An internal use policy is a special preconfigured policy that is designed to achieve a specific outcome. These policies are prepackaged in every Gateway.

To use an internal use policy, choose "Internal Use Policy" as the policy type when creating a new policy, then choose the policy to use from the list of policy tags. For more information, see "Creating a Policy" on page 21.

The following internal use policies are currently available on the CA API Gateway:

wsdm-notifications
Audit Message Filter
Audit Viewer

wsdm-notifications

The wsdm-notifications policy is evaluated for each WSDM notification message. This message is related to a subscription added via the "WSDM Subscription Service" internal service. In this policy, the request is initialized to the notification message.

This policy adds a [Route via HTTP\(S\)](#) assertion that routes to `$(esmNotificationUrl)`, which refers to the value of the

"<wsnt:Subscribe><wsnt:ConsumerReference><wsa:Address>" tag of the Subscribe method in *EsmSubscriptionManagementServiceBinding*.

Audit Message Filter (AMF) Policy

The AMF policy is designed to remove sensitive data from messages and to protect data prior to auditing. This policy is intended to be used in conjunction with the [AV \(Audit Viewer\) policy](#).

After the service policy and any [global policy fragment](#) completes, the AMF policy will be executed for each request and/or the response that will be audited. This allows the policy author to (for example) remove sensitive data from the message or apply any necessary encryption or signature to the message.

Note: The request message that is run through the AMF policy may have undergone a security undecoration process by the Gateway. As a result, it may not be the same as the request first received by the Gateway.

A message is audited under the following conditions:

1. The policy contains the [Audit Messages in Policy](#) assertion, configured with a sufficiently high level.
2. An assertion fails, causing the target message to be audited. (This assumes the *audit.hinting* cluster property is set to its default value of "true".)

If the AMF policy completes successfully, the value in the request or response message is passed onto the Gateway's auditing subsystem (either the internal database and/or an audit log sink, if one has been configured). If the AMF policy fails (that is, one of its assertions returns any assertion status code other than '0'), then the message is not sent to the auditing subsystem. Instead, an audit detail is added to the audit record stating that the AMF policy failed for the relevant message—request or response.

Note: The Audit Message Filter policy only runs for *policy message audits*. To learn more about this type of audit and about Gateway auditing in general, see [Message Auditing](#) in the *Layer 7 Policy Authoring User Manual*.

Keep in mind the following when using the AMF policy:

- Only one AMF policy can be created per Gateway cluster.
- This policy cannot access context variables created by any other service policy or [global policy fragment](#).
- Auditing within an AMF policy is disabled.
- The output of the AMF policy must be text/xml for it to work with the AV policy.
- The AMF policy may use the audit viewer subject certificate as the recipient certificate for the [\(Non-SOAP\) Encrypt XML](#) assertion. If you create an AMF policy after designating an AV key, then the default AMF policy will encrypt for the AV key. However, there is currently no warning if the recipient certificate is something other than the designated AV key.

Understanding the Default AMF Policy

The following default policy is created when you add an AMF policy (assuming that all the assertions are licensed):

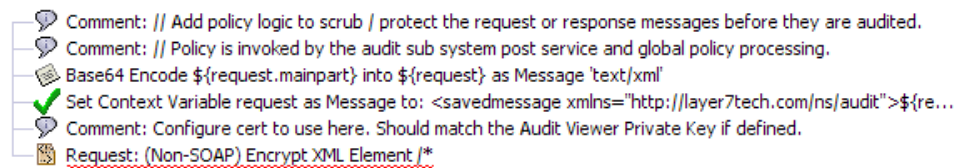


Figure 11: Default AMF policy

The default policy contains these assertions to help you get started:

- [Encode/Decode Data Assertion](#): This assertion encodes the request into Base64 format, which offers the greatest flexibility in handling the various message types.
- [Set Context Variable Assertion](#): This assertion creates an XML Message variable using an arbitrary Layer 7 schema containing the Base64 data.
- [\(Non-SOAP\) Encrypt XML Element Assertion](#): This assertion encrypts the XML message into the request.

Use the default assertions as a starting point to help you create your own AMF policy. For more information, see [Creating a Policy in the Layer 7 Policy Authoring User Manual](#).

Tip: Develop your AMF policy as policy fragments. This makes it easier for testing and troubleshooting. For more information, see [Working with Policy Fragments](#) in the *Layer 7 Policy Authoring User Manual*.

Audit Viewer (AV) Policy

The AV policy can be invoked when viewing audits (for audit messages or audit details) in the Gateway Audit Events window. The AV policy is intended to reverse the actions of the AMF policy. Using security roles, you can restrict the AV policy only to individuals who have a business need to view data protected by the AMF policy. The AV policy uses a special "audit viewer" private key to enforce this restricted access.

For information on viewing audits, see Gateway Audit Events in the *Layer 7 Policy Manager User Manual*. For information on the audit viewer private key, see Private Key Properties in the *Layer 7 Policy Manager User Manual*.

The AV policy takes messages (requests or responses) that were encrypted by the AMF policy and displays them in decrypted form in the Gateway Audit Events window.

Keep in mind the following when using the AV policy:

- Only one AV policy can be created per Gateway cluster.
- The AV policy assumes that the audit message or detail to be processed is in XML format (Content-Type 'text/xml'). If it is not, then the AV policy cannot process it.
- This policy cannot access context variables created by any other service policy or [global policy fragment](#).
- Avoid using the [Run All Assertions Concurrently](#) assertion in the AV policy.

Understanding the Default AV Policy

The following default policy is created when you add an AV policy (assuming that all the assertions are licensed).

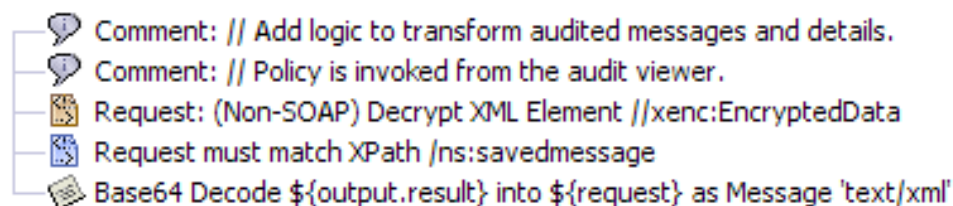


Figure 12: Default AV policy

The default policy contains these assertions:

- [\(Non-SOAP\) Decrypt XML Element Assertion](#): This assertion decrypts the XML that was encrypted by the AMF policy.
- [Evaluate Request XPath Assertion](#): This assertion uses an XPath query to extract the Base64 data from the request.
- [Encode/Decode Data Assertion](#): This assertion decodes the results back to text/xml format.

The default assertions are designed to reverse the effects of the default AMF policy. Use them as a starting point to help you create your own AV policy. For more information, see [Creating a Policy in the Layer 7 Policy Authoring User Manual](#).

Tip: Develop your AV policy as policy fragments. This makes it easier for testing and troubleshooting. Policy fragments can also be used to protect data audited via the [Add Audit Detail Assertion](#). For more information, see [Working with Policy Fragments](#) in the *Layer 7 Policy Authoring User Manual*.

Working with Non-XML Messages

Certain [assertions](#) expect an XML payload and will fail if such a Content-Type is not detected at runtime. If your published service is expected to process both XML and non-XML messages, or process messages without payloads (such as GET or DELETE requests), then you should structure your policy to separate XML processing from non-XML processing. This will prevent inconsequential assertion failures from affecting the outcome of your policy.

For example, the [Protect Against Document Structure Threats](#) assertion is designed to detect XML threats and will fail when processing a non-XML message. However, failing in this manner has little consequence, since non-XML messages cannot contain an XML threat.

Assertions that Require XML

The following assertions require that a message be XML:

- [\(Non-SOAP\) Check Results from XML Verification](#)
- [\(Non-SOAP\) Decrypt XML Element](#)
- [\(Non-SOAP\) Encrypt XML Element](#)
- [\(Non-SOAP\) Sign XML Element](#)
- [\(Non-SOAP\) Verify XML Element](#)
- [Add or Remove XML Elements\(s\)](#)
- [Apply XSL Transformation](#)
- [Document Structure Threats](#)
- [Evaluate Request XPath](#)

Evaluate Response XPath
Require XPath Credentials
Validate XML Schema

Assertions that Require SOAP

The following assertions not only require that the request be in XML but that it be SOAP:

Add WS-Addressing
Add or Remove WS-Security
Add Security Token
Add Timestamp
Configure WS-Security Decoration
Encrypt Element
Encode to MTOM Format
Enforce WS-Security Policy Compliance
Enforce WS-I BSP Compliance
Enforce WS-I SAML Compliance
Evaluate WSDL Operation
Exchange Credentials using WS-Trust
Process SAML Authentication Request
Process RSTR Response
Require Encrypted Element
Require Encrypted UsernameToken Profile Credentials
Require SAML Token Profile
Require Signed Element
Require Timestamp
Require WS-Addressing
Require WS-Secure Conversation
Require WS-Security Kerberos Token Profile Credentials
Require WS-Security Signature Credentials
Require WS-Security UsernameToken Profile Credentials
Sign Element
Use WS-Federation Credential
Validate SOAP Attachments

Example of a Branching Policy

The following sample illustrates how you might implement branching in your policy, to separate the processing of XML requests from non-XML requests:

- 1 At Least One Assertion Must Evaluate to True
- 2 All Assertions Must Evaluate to True

3 Compare Expression: Proceed if \${request.http.header.content-type} contains "text/xml"

4 All Assertions Must Evaluate to True

5 <portion of policy relating to XML messages>

6 All Assertions Must Evaluate to True

7 Compare Expression: Proceed if \${request.http.header.content-type} does not contain "text/xml"

8 All Assertions Must Evaluate to True

9 <portion of policy relating to non-XML messages or empty payloads>

10 <portion of policy common to both>

The following table explains the above sample:

Table 7: Sample branching policy for XML

Line	Description
1	The "At least" branching ensures that either line 2 (message is XML) or line 6 (message is non-XML) is executed.
2	This "All assertions" folder groups the portion of the policy to be processed if the request is XML.
3	Tests whether the message is XML by examining the Content-Type header. If true, processing continues with line 4. If false, then the Compare Expression assertion fails and the "All assertions" in line 2 fails. Processing then continues with line 6.
4	This "All assertions" folder groups the assertions to be processed for XML requests.
5	This portion of the policy contains the assertions listed under "Assertions that Require XML" or "Assertions that Require SOAP" above.
6	This "All assertions" folder groups all the assertions to be processed if the request is non-XML or if line 4 fails.
7	Tests whether the message is non-XML by examining the Content-Type header. If true, processing continues with line 8. If false, then the Compare Expression assertion fails and the "All assertions" in line 6 fails. Processing then continues with line 10.
8	This "All assertions" folder groups the assertions to be processed for non-XML requests.
9	This portion of the policy contains the assertions <u>not</u> listed under "Assertions that Require XML" or "Assertions that Require SOAP" above.
10	List policy logic common to both XML and non-XML messages here.

Working with Comments

It may be useful to annotate your policy with comments. This will make it easier for others to understand the policy logic or to assist you during policy troubleshooting.

Adding a Comment	40
Editing a Comment	41
Deleting a Comment	42

Adding a Comment

There are two different ways to add a comment to your policy:

- Use the [Add Comment to Policy](#) assertion. This assertion can be placed anywhere in the policy and is intended for comments not specific to any assertion (for example, to document the policy logic). You can add as many of these assertions as necessary.
- Append a comment directly to an item in the policy development window, such as an assertion, folder, or policy fragment. This method ensures that the comment remains with the item even after [repositioning](#), copying/pasting, or [exporting/importing](#) the item.

Tip: If your comments are not visible in the policy window, click [**Show Comments**] in the Policy Tool Bar.

➤ *To add a comment to an assertion or folder:*

1. In the policy window, right-click the assertion or folder and then select **Add Comment**. The Enter Comment dialog is displayed:

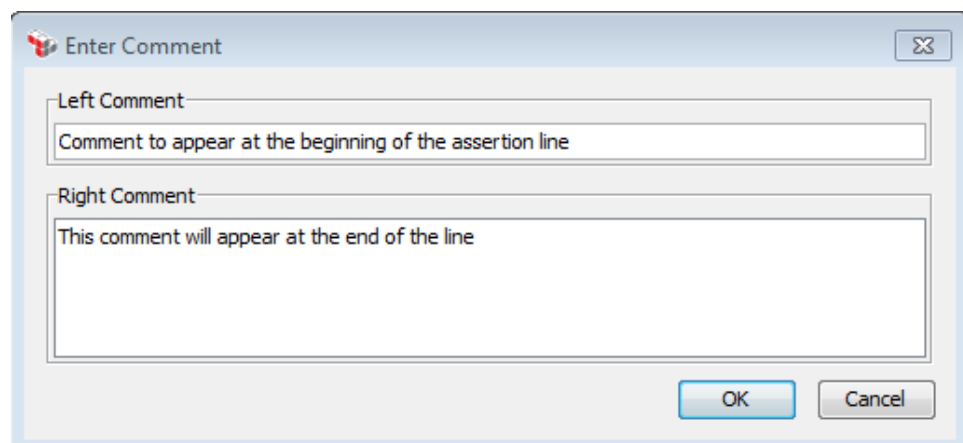


Figure 13: Enter Comment dialog

2. Type your comments in the "Left" and "Right" boxes:

- **Left Comment** will appear before the item, left-aligned in the policy window. The maximum for this comment is 100 characters.
- **Right Comment** will appear after the item, right-aligned in the policy window. The maximum for this comment is 4000 characters.

Here are some tips when entering comments:

- By default, only the first 30 characters of the left comment and first 100 characters of the right comment are displayed in the policy development window. These can be changed in the Preferences.
- Comments will be displayed in a tooltip when you hover the mouse pointer over the assertion in the policy window. This is useful to read long comments that have been truncated.
Exception: Comments are not visible if the tooltip is displaying a warning that should be resolved. For example, you might see this warning message in a tooltip: *"The policy may be invalid due to warnings. The assertion might now work as configured."*
- You can prefix the comment with any separator character but you are not required to do so. The comments will be displayed in a different font color in the policy window to make them stand out.

3. Click **[OK]**. The comment is displayed next to the assertion or folder in the policy window, unless comments have been hidden.

Editing a Comment

➤ *To edit a comment in the policy window:*

- If the comment appears as **"Comment: <comment text>"**, then double-click it to edit the comment. For more information, see "Add Comment to Policy Assertion" on page 618.
- If the comment is in light grey text aligned to the left or right of an assertion or folder:
 - a. Right-click the assertion or folder and select **Edit Comment**.
 - b. Modify the comment as required. For more information, see "Adding a Comment" on page 40.

Deleting a Comment

➤ To delete a comment in the policy window:

- If the comment appears as "**Comment:** <comment text>", then right-click it and select **Delete Assertion**. The entire "Comment:" line is removed. For more information, see "Deleting an Assertion" on page 119.
- If the comment is in light grey text aligned to the left or right of an assertion or folder, then right-click it and select **Delete Comment**. The comment text is removed after confirmation.

IMPORTANT: If both left and right comments exist, they will both be removed. If you wish to remove only one of the comments, [edit the comment](#) instead and remove the desired comment.

Exporting/Importing a Policy

The following topics describe how to export or import a policy, as well as to resolve any conflicts that may arise during importing.

Exporting a Policy	42
Importing a Policy from a File	44
Importing a Policy via UDDI Registry	46
Import WS-Policy from URL in UDDI Registry Wizard	47
Searching the UDDI Registry	48
Resolve External Dependencies Wizard	51

Exporting a Policy

The Policy Manager allows you to export a policy to a file. Use this feature to share policies internally or externally, or to save copies of policies for record-keeping purposes. Exported policies that are saved locally can be renamed, deleted, or edited by replacement.

It is recommended that you only export valid policies. Validation confirms the proper configuration and organization of a policy. See "Validating a Policy" on page 25 for more information. An exported policy may contain [disabled](#) assertions. If security zones have been defined, you must have Read permissions to the policy in order to export.

The portable policy XML file generated during export includes references to:

- The identity providers belonging to the users and groups in the policy
- The JMS routing endpoints, or destinations, if included in the policy, and
- Any [custom assertions](#), if present in the policy.

When exporting a policy using the browser client version of the Policy Manager, the Java applet must be running in the trusted mode. For more information, see Policy Manager Browser Client in the *Layer 7 Policy Manager User Manual*.

➤ *To export a policy:*

1. Open the policy to be exported using either of the following methods:
 - Right-click the service name in the Services and Policies list and then select **Active Policy Assertions**. Or,
 - Double-click the service name in the Services and Policies list.
2. Click **[Export Policy]** on the Policy Tool Bar or select The Export Policy dialog appears.
3. Do one of the following:
 - **If using the standard client:** Either use the default directory offered or navigate to another location. If you use the default location ("..\\l7tech\\policy.templates"), the exported policy will appear under the Policy Templates section of the [Assertions] tab.

Note: Only exported policies saved as a template in the default directory will appear under the Policy Templates section of the [Assertions] tab. Policies saved to any other location will not appear in the Policy Templates section. Policies can be [imported](#) regardless of their saved locations.

- **If using the browser client:** Navigate to a folder of your choice to save the template.

Note: Be sure to note the location of the saved templates for later import. This is because there is no Policy Templates section in the [Assertions] tab for the browser client version.

4. Enter a descriptive name for the exported policy and then click **[Save]**. The policy is exported as a portable XML file which encapsulates all of the assertions and back-end settings. The policy appears under **Policy Templates** in the [Assertions] tab (standard client version only).

Tip: You can open the exported XML file in a text editor to view policy details. Be sure you have a backup before you make any modifications to the file.

Importing a Policy from a File

The Policy Manager allows you to import a policy into the policy development window from a file. This ensures policy consistency and saves configuration time. Importing a policy is particularly useful for sharing policies with external departments, partners, and others who have separate Gateway installations.

The imported XML file encapsulates all of the originating policy information. Any [disabled](#) assertions in the imported policy will remain disabled after import.

When importing a policy using the browser client version of the Policy Manager, the Java applet must be running in the trusted mode. For more information, see Policy Manager Browser Client in the *Layer 7 Policy Manager User Manual*.

Importing encapsulated assertions: Do not use the policy import feature if you want to import an encapsulated assertion. To correctly import an encapsulated assertion, use the Import button in the Manage Encapsulated Assertions Configuration dialog. For more information, see "Working with Encapsulated Assertions" on page 126 and "Managing Encapsulated Assertions" on page 132.

Note the following security zone considerations:


- If the policy being imported belongs to a security zone, you must have a security role that permits updating of the policy.
- If the policy being imported contains assertions that have been placed in a security zone, you must have a security role that has Read permissions for those assertions, otherwise you will not be able to save the imported policy.

➤ *To import a policy from a file:*

1. Make sure the service that is receiving the imported policy is open. If not, double-click the service in the Services and Policies list, or right-click the service name and select **Active Policy Assertions**. The policy development window appears.
2. Import your policy using either of the following methods:

Table 8: Options for importing a file

Method	Description
Import from "Policy Templates"	Use this method if the policy you want is visible under the Policy Templates section of the [Assertions] tab. This method is not available in the browser client version of Policy Manager.

Method	Description
<i>(Standard client only)</i>	<p>Do one of the following:</p> <ul style="list-style-type: none"> • Drag and drop the policy you want from the Policy Templates section into the policy development window. • Select the template in the Policy Templates section and click the  (Add Assertion) button in the Assertions Tool Bar.
Import from any file <i>(Standard and Browser client version)</i>	<p>Use this method to import version 3.0 or later XML policy files. Note: In the browser client, importing is possible only when the Java applet is running in the trusted mode.</p> <ol style="list-style-type: none"> 1. Click [Import Policy] on the Policy Tool Bar. The default folder for storing saved templates appears. 2. Select the template to import. If the policy was not stored in the default location, navigate to the correct folder first. 3. Click [Open].

Note: When you import a policy, the target policy in the policy development window is completely replaced by the elements in the incoming policy template. These include [policy assertions](#), [policy fragments](#), identity providers, JMS destination references, and any [custom assertion](#), if present.

3. Before importing the policy into the policy development window, the Policy Manager automatically attempts to resolve the back-end requirements of the imported policy against the back-end configuration of the target policy's Gateway.
 - If the automatic reconciliation is successful, the imported policy will appear in the policy development window.
 - If the automatic reconciliation is not successful, the [Resolve External Dependencies Wizard](#) appears. Use this wizard to instruct the Policy Manager how to handle each unresolved element.

The wizard appears if the imported policy contains references to elements that are not present on the target system. This will typically happen if the policy came from another system or if the policy refers to an element (for example, a user or group) that had been deleted since the policy was originally exported.

The Policy Manager compares the object-level property values of the imported identity provider with each identity provider configured in the target Gateway. A difference in even one value will cause a reconciliation failure

4. The routing assertion(s) and other assertions in the policy development window are specific to the service that originated the policy. Edit the assertions for the

target service as required:

- See [Message Routing Assertions](#) to re-configure the replaced routing assertion with service-specific information
- See [Policy Assertions Overview](#) to re-configure other assertions as required.

Ensure that policy [edits](#) conform to the policy and assertion rules outlined in "Policy Organization" on page 2.

5. Finish the import procedure by doing the following:

- Proceed to [Validating a Policy](#) to perform a final validation check on the policy. When the policy passes the validation process, enable the service, if necessary
- If the imported policy contains a [Validate XML Schema](#) assertion that includes an import statement, then you will need to resolve the external reference(s) using the [Manage Global Resources](#) task.
- *(Optional)* [Export](#) the validated policy as a new policy template, or use it to replace an existing template. This provides a backup of your policy for safekeeping.

Importing a Policy via UDDI Registry

Importing a policy via the UDDI registry is similar to [importing a policy from a file](#), except that the source is not a policy XML file on a hard disk, but rather XML resolved from an HTTP URL [published](#) in a UDDI registry.

Note: Importing a policy via the UDDI registry is supported in the browser client version of the Policy Manager only when the Java applet is running in the trusted mode. For more information, see Policy ManagerBrowser Client in the *Layer 7 Policy Manager User Manual*.

➤ *To import a policy via the UDDI Registry:*

1. Make sure the service that is receiving the imported policy is open. If not, double-click the service in the [Services] tab, or right-click the service name and select **Policy Assertions**. The policy development window appears.
2. Click **[Import from UDDI]** on the Policy Tool Bar. The [Import WS-Policy from URL in UDDI Registry](#) wizard appears.
3. Complete the wizard to import the policy.

Import WS-Policy from URL in UDDI Registry Wizard

The *Import WS-Policy from URL in UDDI Registry Wizard* extracts a WS-Policy document from a URL in the UDDI registry. This wizard starts when you attempt to [import](#) a policy via the UDDI registry.

Note: Importing a policy via the UDDI registry is supported in the browser client version of the Policy Manager only when the Java applet is running in the trusted mode. For more information, see *Policy ManagerBrowser Client* in the *Layer 7 Policy Manager User Manual*.

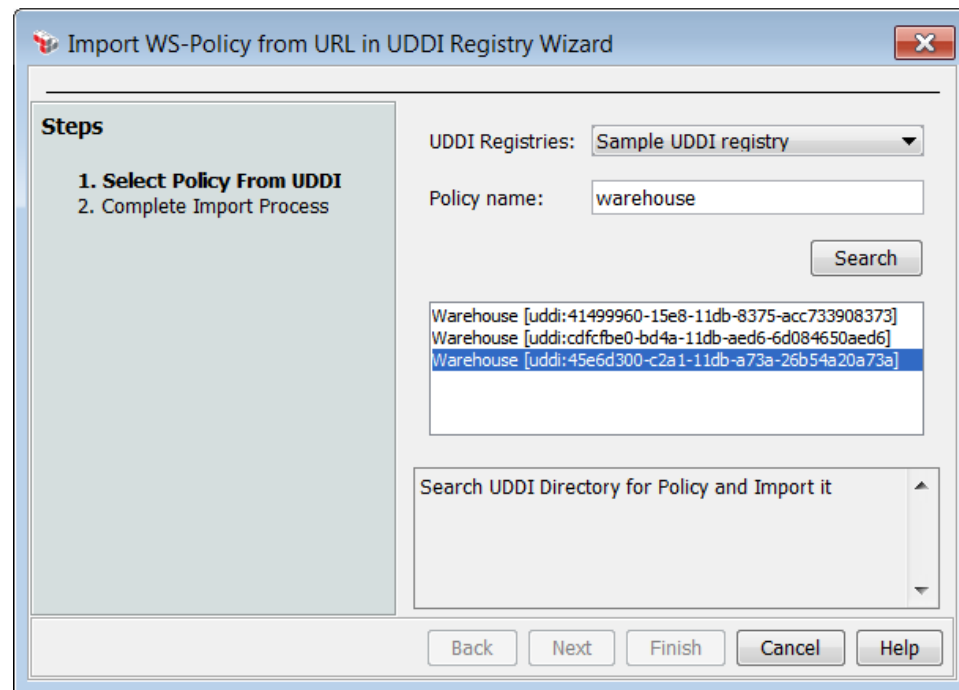


Figure 14: Import WS-Policy from URL in UDDI Registry Wizard

For more information about wizards, see "Wizard" under Interfaces in the *Layer 7 Policy Manager User Manual*.

Table 9: Using the Import WS-Policy from URL in UDDI

Wizard Step	Description
Step 1: Select Policy from UDDI	<p>Select the policy from the UDDI registry to be imported:</p> <ol style="list-style-type: none"> 1. Select the registry to use from the UDDI Registries drop-down list. These registries were defined using the Manage UDDI Registries task. Only enabled registries are displayed. 2. Optionally type a few characters of the policy name in the Policy name field. This will help narrow down the search if there are many policies stored in the UDDI registry. Leave this field blank to

Wizard Step	Description
	<p>see all the policies.</p> <ol style="list-style-type: none"> Click [Search]. The policies containing the string entered above are displayed. If no string was entered, all policies are listed. Select the policy you want and then click [Next]. <p>Note: You will see an error if the selected policy does not resolve to a policy document. If this happens, try another policy. If no policy yields a policy document, then you cannot import a policy from the specified UDDI registry.</p>
Step 2: Complete Import Process	<p>Review the details for the selected policy to ensure that you have selected the correct one. Click [Finish] to close the wizard and import the policy into the policy development window.</p>

Searching the UDDI Registry

The Search UDDI dialog allows you to search a UDDI registry while performing the following tasks:

- When publishing a SOAP web service, you can enter either the URL or file path to the WSDL document. If a UDDI registry has been configured, you can also search the UDDI to retrieve the appropriate URL.
- When [publishing a business service](#), you can search a UDDI registry for a specific business entity.

Note: Before you can search a UDDI registry, ensure that the UDDI registry product is correctly installed and at least one UDDI registry has been configured in the Gateway. For more information, see "Managing UDDI Registries" on page 87.

➤ *To search a UDDI registry:*

- Do one of the following:
 - Click **[UDDI]** button in Step 1 of the Publish SOAP Web Service Wizard in the *Layer 7 Policy Manager User Manual*. **Note:** If the **[UDDI]** button is not visible, then no UDDI registries were configured. For more information, see "Managing UDDI Registries" on page 87.
 - Click the **[Select]** button on the **[UDDI]** tab of the Service Properties dialog.

The Search UDDI dialog is displayed. Figure 15 shows the dialog that is displayed when searching for a WSDL URL from a UDDI registry.

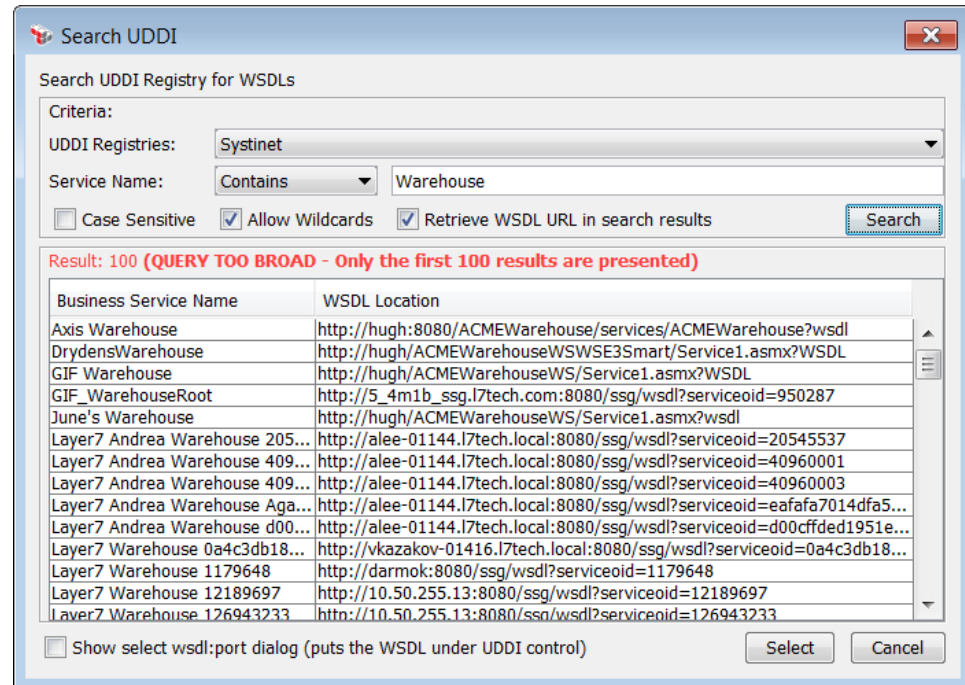


Figure 15: Search UDDI dialog (Search UDDI Registry for WSDL example)

2. Configure the search as follows:

Table 10: Configuring the Search UDDI settings

Setting	Description
UDDI Registries	From the drop-down list, select the UDDI registry to be searched. Only registries that are enabled are shown.
Service Name	To refine your search, you can optionally specify that the Service Name Contains or Equals the string of characters that you specify. ('Contains' encloses the search string within '%' characters.) Leave the Service Name search field blank to retrieve all available services.
Case Sensitive	Select this check box to make the search case sensitive.
Allow Wildcards	Select this check box to use the percent symbol (%) wildcard to match any number of characters, or the underscore symbol (_) to match any single character. Wildcards can be used with both 'Contains' and 'Equals'. When used with 'Contains', they work in addition to the '%' already added to the start and end of the entered search text.
Retrieve WSDL URL in search results (only for Service)	Select this check box to retrieve the WSDL URL for each service in the search results. Clearing this check box will improve the search performance as only a single UDDI query needs to be made.

Setting	Description
<i>searches)</i>	Tip: Many UDDI queries are required to assemble the list of service names and WSDL locations. By omitting the WSDL URL, fewer searches are required and the results are returned more quickly. When you select a service, the WSDL location is then resolved for that service.
[Search]	Click this button to begin the search. The search will run until it returns results or is cancelled by the user.
Results	The number of items found is displayed. The table shows the results of the search. The WSDL Locations column will be blank if the Retrieve WSDL URL in search results check box is not selected.
Show select wsdl:port dialog <i>(only for Service searches)</i>	Select this check box if you need to select a specific port to use from the WSDL. See "Selecting a wsdl:port" below for more information.

- Click **[Select]** after selecting the Business Service or Business Entity. The Search UDDI dialog closes and your selected service is added to the previous dialog.

Selecting a wsdl:port

When the **Show select wsdl:port** check box is selected in Figure 15, the "Select wsdl:port" dialog is displayed after you select a row and click **[Select]**. This dialog allows you to select a specific wsdl:port to use.

Tip: Why select a wsdl:port? You will do this if you want to place the published service's WSDL to be under the control of the UDDI registry. Once the WSDL is under UDDI control, its possible to enable monitoring. Monitoring cannot be enabled unless the Gateway knows which specific wsdl:port (bindingTemplate) in the UDDI Business Service to monitor.

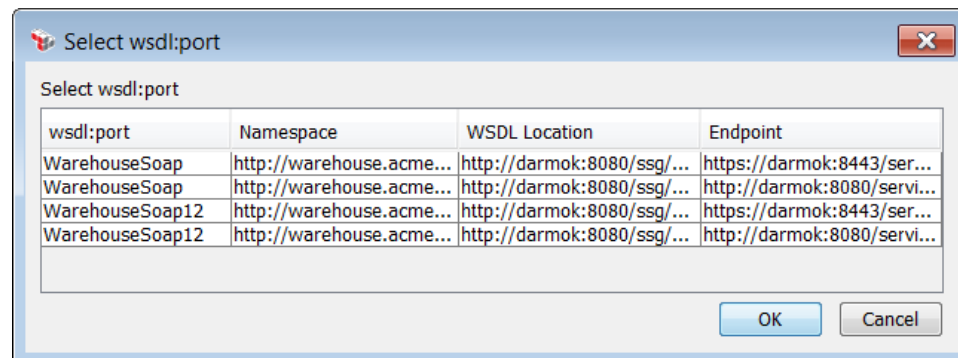


Figure 16: Selecting a wsdl:port

Technical note

When a Gateway publishes a WSDL to a UDDI Registry, it may publish each wsdl:port twice, if the Gateway cluster defines an HTTPS and an HTTP endpoint. When this happens, the search results will show more than a single wsdl:port and possibly namespace with the same values. In this case the endpoint values should differ by protocol. For WSDL's published to UDDI which do not originate from the Gateway, the wsdl:port and namespace columns uniquely identify each wsdl:port from a WSDL.

Resolve External Dependencies Wizard

The *Resolve External Dependencies Wizard* lets you manually reconcile the following from an imported policy:

[Custom assertions](#)

Identity providers

JDBC connections

JMS routing endpoints

[Policy fragments](#)

Private keys

SiteMinder configurations

Stored passwords

Trusted certificates

[XML schemas](#)

This wizard appears when Policy Manager is unable to automatically reconcile these elements during import. For more information, see "Importing a Policy from a File" on page 44.

For more information about wizards, see "Wizards" under Interfaces in the *Layer 7 Policy Manager User Manual*.

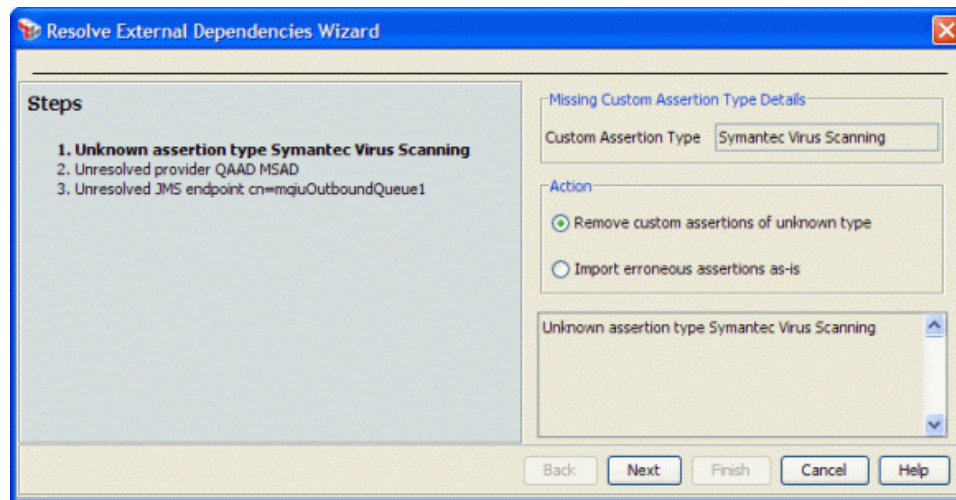


Figure 17: Sample unresolved dependency in Resolve External Dependencies Wizard

The steps that appear in this wizard depend on the elements that require manual reconciliation; the steps shown in Figure 17 are just an example. Table 11 describes all the external dependencies that can be resolved by this wizard.

Note that you can click **[Cancel]** at any time to cancel the importing of the policy.

Table 11: Resolve External Dependencies Wizard settings

Dependency	Description
Unknown custom assertion	<p>The incoming policy contains one or more custom assertions that are not configured in the target policy is listed. Choose a solution:</p> <ul style="list-style-type: none"> Remove the unknown assertion from the import. Ignore the error and import the assertions as-is.
Unresolved global resource	<p>The incoming policy contains assertions that reference an unresolved global resource. The missing global resource details are displayed. Choose a solution:</p> <ul style="list-style-type: none"> Remove the assertions from the policy that refer to the missing global resource. Ignore the error and import the assertions as-is. Click [Add Global Resource] to manually add the missing

Dependency	Description
	global resource to the Policy Manager. Complete the Edit Global Resource dialog. For more information, see "Adding a New Global Resource" on page 75.
Unknown identity provider	<p>The incoming policy contains assertions referring an identity provider that is unknown in the target policy. Details of the identity provider are displayed. Choose a solution:</p> <ul style="list-style-type: none"> • Select a local identity provider to substitute for the imported identity provider. This option is available only when there is another identity provider of the same type to choose. • Remove the assertions from the policy that refer to the missing identity provider • Ignore the error and import the assertions as-is • Click [Create a new Identity Provider] to configure a new identity provider. Use the details displayed to assist you, if necessary. Complete the Federated Identity Provider Wizard or LDAP Identity Provider Wizard that appears. • Exit the wizard and edit an existing identity provider so that its properties match the imported identity provider. Repeat the import process and then choose the first option in the wizard ("Change assertions to use this identity provider"). <p>Tip: When creating or editing an identity provider, consult the properties values of the imported identity provider found in the imported policy XML file. Open the file in a text editor and note the values in the "<exp:References>" parameter. Policies displayed under Policy Templates can be found in the ".\I7tech\policy.templates" directory.</p>
Unresolved JDBC connections	<p>The incoming policy contains a JDBC connection that cannot be resolved in the target policy. The name of the missing JDBC connection is displayed. Choose a solution:</p> <ul style="list-style-type: none"> • Select another connection from the drop-down list. • Remove the assertions from the policy that refer to the missing connection. • Ignore the error and import the assertions as-is. • Click [Manage JDBC Connections] to create a new JDBC connection. Complete the Manage JDBC Connections dialog. For more information, see Managing JDBC Connections in the <i>Layer 7 Policy Manager User Manual</i>.
Unresolved JMS routing endpoint	<p>The incoming policy contains a Route via JMS assertion or JMS endpoints that cannot be resolved in the target policy. Details of the missing JMS endpoints are displayed. Choose a solution:</p> <ul style="list-style-type: none"> • Change the assertions to use another JMS endpoint selected from the drop-down list. This option is available only when there is another JMS endpoint to choose.

Dependency	Description
	<ul style="list-style-type: none"> Remove the assertions from the policy that refer to the missing endpoint. Ignore the error and import the assertions as-is. Click [Manage JMS Destinations] to create a new JMS endpoint. Complete the Manage JMS Destinations dialog. For more information, see Managing JMS Destinations in the <i>Layer 7 Policy Manager User Manual</i>. Exit the wizard and edit existing inbound/outbound queues to match the configuration of the imported queue references. Repeat the import process and then choose the first option in the wizard ("Change assertions to use this endpoint"). <p>Tip: When creating or editing a JMS destination, consult the properties values of the imported queues found in the imported policy XML file. Open the file in a text editor and note the values in the "<exp:References>" parameter. Policies displayed under Policy Templates can be found in the ".\7tech\policy.templates" directory.</p>
Unresolved policy fragments	<p>The incoming policy contains a policy fragment that has the same name as an existing fragment, but has a different GUID. You are prompted to enter a new name for the incoming policy fragment.</p> <p>Note: If the incoming policy has the same GUID as an existing fragment but different contents, the Resolve External Dependencies Wizard does not appear. Instead, you are notified that the existing fragment will be used instead of the incoming fragment.</p>
Unresolved private keys	<p>The incoming policy contain a private key that is not present in the target policy. Details of the missing key are displayed. Choose a solution:</p> <ul style="list-style-type: none"> Use default private key: Select this option to use the default SSL key for the target policy. Use custom private key: Select this option to choose another private key from the drop-down list. You can click [Manage Private Keys] to import or create new private keys or to view details for any key. For more information, see Managing Private Keys in the <i>Layer 7 Policy Manager User Manual</i>. Remove all assertions from the incoming policy that refer to the missing private key. Import the erroneous assertions without changes. You will need to correct this error later to prevent policy validation errors.
Unresolved SiteMinder Configuration	<p>The incoming policy contains assertions referencing a SiteMinder configuration that does not exist in the target policy. The following missing SiteMinder details are displayed:</p> <p style="padding-left: 40px;"><i>Configuration Name</i> <i>Hostname</i></p> <p>Choose a solution:</p>

Dependency	Description
	<ul style="list-style-type: none"> • Change the assertion to reference another SiteMinder configuration that does exist in the policy. • Remove the assertions that refer to the missing configuration. • Ignore the error and import the assertions as-is. • Click [Create SiteMinder Configuration] to create a new SiteMinder configuration.
Unresolved stored password	<p>The incoming policy contains one or more assertions that refer to unknown stored (secure) passwords. The details for the unknown stored passwords are displayed. Choose an action:</p> <ul style="list-style-type: none"> • Change the incoming assertions to use another stored password instead. Either choose a stored password from the drop-down list. If the password you require is not listed, click [Create Stored Passwords] to define one now. For more information, see <i>Managing Stored Passwords in the Layer 7 Policy Manager User Manual</i>. • Remove all assertions from the incoming policy that refer to the missing stored password. • Import the erroneous assertions without changes. You will need to correct this error later to prevent policy validation errors.
Unresolved trusted certificate	<p>The incoming policy contains one or more assertions that refer to unknown trusted certificates. The details for the unknown certificate are displayed. Choose an action:</p> <ul style="list-style-type: none"> • Change the incoming assertions to use another trusted certificate instead. Either select a certificate listed and then click [Select a Certificate] or click [Create a new certificate] to create a new trusted certificate. For more information see <i>Adding a New Certificate in the Layer 7 Policy Manager User Manual</i>. • Remove all assertions from the incoming policy that refer to the missing trusted certificate. • Import the erroneous assertions without changes. You will need to correct this error later to prevent policy validation errors.
Unresolved XML schemas	<p>The incoming policy contains assertions that refer to an unresolved external schema. Information about the missing external schema is shown. Choose an action:</p> <ul style="list-style-type: none"> • Change the assertion to use another schema from the drop-down list. This option is not available if no other suitable schemas are available. <p>Note: If a target namespace is listed, then only schemas from that namespace are shown, otherwise all namespaces are shown. Ensure that the schema you choose is an appropriate schema to use as a replacement, as the wizard does not check</p>

Dependency	Description
	<p>for appropriateness.</p> <ul style="list-style-type: none"> Remove the Validate XML Schema assertions that reference the missing schema. Ignore the error and import the assertions as-is. The wizard indicates whether the reference is currently valid or invalid (depending on schemas added using [Add External Schema]). If the reference is invalid, you will need to correct this error later to prevent policy validation errors. Click [Add External Schema] to add a new schema to the Gateway. For more information, see "Adding a New Global Resource" on page 75.

When the wizard is finished, the imported policy will appear in the policy development window.

Note: If you chose "Import assertion as-is" in any wizard step, then the imported policy will contain validation errors. These errors must be corrected before the policy is used. For more information, see "Validating a Policy" on page 25.

Debugging a Policy

The following topics describe how to use the debugging features in the Policy Manager to perform advanced policy troubleshooting.

Working with the Service Debugger	56
Policy Debug Tracing	66
Working with the Debug Trace Policy	67

Working with the Service Debugger

The Policy Manager has a built-in debugger that can help you troubleshoot your policies. This debugger behaves much like the debuggers available within programming environments, allowing you to:

- Add or remove breakpoints
- Step through a policy and view its path
- Step into or over composite assertions
- View values within context variables
- Pause and resume debugging

Tip: The Service Debugger is different from the [Debug Trace Policy](#) that can also be used to help you troubleshoot your policies in the Policy Manager. For more information, see "Working with the Debug Trace Policy" on page 67.

Keep in mind the following before you debug a policy:

- Only the main service policy and [global policy fragments](#) can be debugged. [Aliases](#) and [all other policy types](#) cannot be debugged.
- [Included policy fragments](#) cannot be debugged on their own, but they can be debugged once inserted into a service policy or global policy fragment (using the "Step Into" function of the debugger).
- Only the active and saved [version](#) of a policy can be debugged. To debug another policy or version, close and reopen the debugger.
- There can be only one active debugger session per policy per Gateway node. In a clustered environment, you can have one active debugger session per policy per node.
- [Encapsulated assertions](#) cannot be stepped into.
- Once the debugger is started, the next message that arrives at the service endpoint is sent to the debugger, regardless of port number. In a high traffic Gateway, consider creating a copy of the policy and start the debugger in the copied policy.
- If the active version of a policy changes after a debugger is started, be sure to close and reopen the debugger to re-synchronize the debugger with the correct policy.
- Be aware that the debugger is attached to a service on a *specific* node. This means that if the debugger is started on only some (or one) node in a clustered environment, the load balancer may route a message to a node without the debugger attached. To prevent this from happening, start the debugger on all nodes for the service.

Security Roles

In order to use the Service Debugger, you must have debugger permission to the policy being debugged. The following predefined roles have this permission:

- **Administrator:** Allows you to launch the debugger for all policies.
- **Manage Webservices:** Allows you to launch the debugger for all policies.
- **Manage [name] Service:** Allows you to launch the debugger for the named service only.

Note: Debugger access to any global policy fragments also require a separate "Manage [name] Policy" role. If the service policy contains an included fragment, you require Read permission to that fragment in order to view or step into that fragment.

- **Manage [name] Policy:** Allows you to launch the debugger for the named [global policy fragment](#) only. If the service policy contains an included fragment, you require Read permission to that fragment in order to view or step into that fragment. **Note:** For all other [fragment types](#), this role has no impact on the Service Debugger.

For more information about the predefined roles, see Predefined Roles and Permissions on page 1.

Notes: (1) The "Service Debugger" option will not be visible for unsupported roles. (2) It is currently not possible to add debugger permissions to custom roles.

Running the Service Debugger

➤ *To debug a policy:*

1. Right-click a policy in the services and policies list and then select **Service Debugger**. The active version of the policy is loaded into the Service Debugger dialog.

Notes: (1) Only the main service policy and [global policy fragments](#) can be debugged. All other policy types cannot be debugged and will not display the "Service Debugger" option. (2) You must have [debugger permission](#) for the given policy in order to see the debugger option.

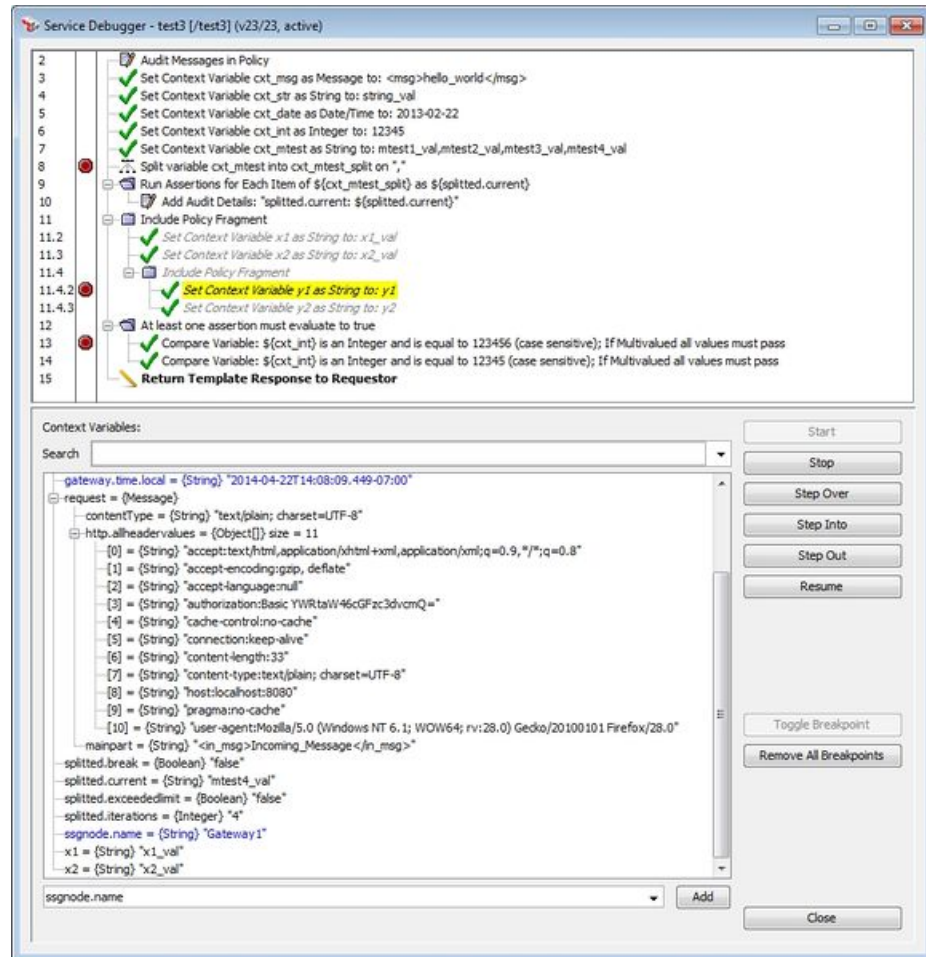


Figure 18: Example Service Debugger dialog, with a debug session in progress

The top pane displays the policy that was active when the Service Debugger was open. Once debugging begins, the bottom pane shows the context variables in use by the policy, along with their values at the particular point in the policy. For more information, see ["Using the Context Variables Tree"](#). **Tip:** You may resize the Service Debugger dialog and alter the size of each pane by dragging the split bar separating the two panes.

2. Add one or more breakpoints to the policy. This allows you to temporarily pause policy processing in order to examine the results. For more information, see ["Using Breakpoints"](#).
3. Optionally add more context variables that you wish to inspect to the variables tree in the bottom pane. For more information, see ["Using the Context Variables Tree"](#).

4. Click **[Start]** (shortcut key: *[F1]*) to start the debugging monitor. The Service Debugger will wait for the next message destined for the service endpoint. When a message arrives, it is sent to the debugger and will be processed by the policy until it reaches the first breakpoint, at which point processing is suspended. If no breakpoints are defined, then the message runs to the end of the policy.
5. When processing pauses at a breakpoint, you can choose to use one of the stepping options to manually step through the assertions or click **[Resume]** to resume processing until the end of the policy or the next breakpoint is reached, whichever comes first. For more information, see ["Stepping Through the Policy"](#).
6. When the policy finishes executing, a message is displayed in the status area at the bottom of the dialog.
 - If the policy executed successfully, the message will read "Policy completed successfully".
 - If the policy did not complete successfully, the message describes the failure and the line number where it occurred; for example: "Policy completed with error. Assertion Falsified: assertion number 27".

At this point, you can choose to do any of the following:

- Click **[Start]** to restart debugging monitoring again.
- Modify your [breakpoints](#) or [list of context variables](#) before restarting monitoring.
- Click **[Close]** to dismiss the Service Debugger dialog and then alter the policy before debugging again.

You can click **[Stop]** (shortcut: *Shift+[F1]*) at any time to stop the debugging.

Using Breakpoints

Breakpoints allow you to suspend processing of a message at a particular point in the policy. This allows you to examine the values of [context variables used in the policy](#) or to [manually step through the policy](#).

Notes: (1) Breakpoints are discarded when you close the Service Debugger dialog. (2) Breakpoints cannot be added to the [Add Comment to Policy](#) assertion or for any [disabled](#) assertion in the policy. (3) Breakpoints added to assertions within the [Run All Assertions Concurrently](#) assertion will be ignored during debugging. (4) Breakpoints inside of composite assertions can affect how the "stepping" controls work. See ["Stepping Through the Policy"](#) for more details.

You can set a breakpoint in any number of ways:

- Click in the white space between the line number and assertion name in the top pane of Figure 18.
- Right-click an assertion and select **Toggle Breakpoint**.
- Select an assertion and then click **[Toggle Breakpoint]**.

You can clear a breakpoint in any number of ways:

- Click the breakpoint icon next to the assertion name.
- Right-click an assertion and select **Toggle Breakpoint**.
- Select an assertion and then click **[Toggle Breakpoint]**.
- Click **[Remove All Breakpoints]** to delete all breakpoints at once. This is useful if you wish to replace all existing breakpoints with new ones. **Tip:** Breakpoints are not saved when the debugger is closed, so it is not necessary to clear the breakpoints before returning to your policy.

When processing is paused at a breakpoint, the assertion is highlighted in yellow to indicate the progress of the message. Choose one of the "stepping" options below to continue.

Stepping Through the Policy

When processing is paused at a breakpoint, you can use one of the stepping options in Figure 18 to manually step through the policy.

Note: If breakpoints exist inside a composite assertion or included policy, they take precedence over the stepping hierarchy. See examples below for more details.

Step Over

Click **[Step Over]** to step to the next non-disabled assertion *at the same level*. For composite assertions, [Step Over] will move to the composite assertion parent, then to the next equal-level assertion immediately after the composite assertion. *Shortcut key: [F3]*

For example, consider the following:

```

Assertion 1    <-- breakpoint here
Assertion 2 (composite assertion or policy
fragment)
    Child A
    Child B
    Child C
Assertion 3
  
```

This is how [Step Over] will behave:

- *Assertion 1 --> Assertion 2 --> Assertion 3*
- If also breakpoint at Child A: *Assertion 1 --> Assertion 2 --> Child assertion A --> Assertion 3*
- If also breakpoints at Childs A & C: *Assertion 1 --> Assertion 2 --> Child A --> Child C --> Assertion 3*

Step Into

Click **[Step Into]** to step to the next non-disabled assertion, regardless of hierarchy.

Shortcut key: [F2]

Note: It is not possible to step into the [Run All Assertions Concurrently](#) assertion or an [encapsulated assertion](#). Clicking [Step Into] in this case will have the same effect as [Step Over].

Step Out

Click **[Step Out]** if you wish to exit processing a composite assertion. This will select the next parent-level assertion. *Shortcut key: Shift+[F3]*

For example, consider the following:

```

Assertion 1
  Assertion 2 (composite assertion or policy
fragment)
    Child A
    Child B <-- breakpoint here
    Child C
    Assertion 2a (composite assertion or
policy fragment)
        Child D
        Child E
    Assertion 3

```

This is how [Step Out] will behave:

- From *Child B --> Assertion 3*
- If also breakpoint at Child D: *Child B --> Child D --> Assertion 3*
- If also breakpoints at Child D & E: *Child B --> Child D --> Child E --> Assertion 3*

Note: Stepping out within a [Run Assertions for Each Item](#) assertion will step out of the *current loop only*, not the entire assertion. If there are more iterations remaining in this assertion, the debugger will enter the loop again and then stop at the next breakpoint within that assertion (if one exists).

Resume

Click [**Resume**] to continue processing based on the policy logic. Processing continues until the next breakpoint or the end of the policy, whichever comes first. (*shortcut key: [F4]*)

Stop

Click [**Stop**] to stop debugging, but leave the Service Debugger dialog open. Policy processing continues in the background until the end of the policy is reached.

Using the Context Variables Tree

The context variables tree (bottom half of Figure 18) displays the context variables that have been set as of a particular breakpoint. It offers an easy way to "peek" into a variable in real time during processing. The context variables are listed in the following format:

name = {dataType} "value"

Where:

- **name** is the name of the context variable
- **dataType** is the Java class name of the context variable (for example: String, Integer, Message, ArrayList)
- **value** is the value of the variable at that particular breakpoint

Example:

```
error.status = {String} "403"
```

Tips: (1) The context variables are listed in alphabetical order. (2) You can copy any line in the context variables tree by selecting it and pressing Ctrl-[C]. Note that child nodes need to be specifically selected (use the Shift or Ctrl keys to select). (3) The context variables are retained when the debugger stops, but are cleared when the debugger is started again.

Message Variables

For context variables of type Message, child nodes display the values for the attributes of the message; for example:

```
request = {Message}
  contentType = {String} "text/xml;charset=UTF-8"
  http.allheadervalues = {Object[]} size = 7
    [0] = {String} "accept-encoding:gzip,deflate"
    [1] = {String} "connection:Keep-Alive"
    [2] = {String} "content-length:298"
    [3] = {String} "content-type:text/xml;charset=UTF-8"
    [4] = {String} "host:docsg1.l7tech.com:8080"
    [5] = {String} "soapaction:""
    [6] = {String} "user-agent:Apache-HttpClient/4.1.1 (java 1.5)"
  mainpart = {String} "<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/
```

Figure 19: Sample message variable

Interpreting the example above:

- The context variable is a request message, where "request" is the root node of the variable. All information about this variable are given in the child nodes.
- The three context variables for this message are:

```
request.contentType
request.http.allheadervalues
request.mainpart
```

Tip: These three variables are automatically shown in the context variables tree as you step through the policy.

- The variable `request.http.allheadervalues` is an array containing seven values. The child nodes list of the value of all the headers in the message.

List Variables

For context variables of type List or Array, child nodes display the value of each index. For an example, see "http.allheadervalues" in Figure 19.

Context Variable Exceptions

Context variables set by the [Evaluate Request XPath](#) and [Evaluate Response XPath](#) assertions will not show in the context variables tree unless these context variables are used later in the policy. To make these variables visible, add an assertion to the policy that uses these variables (for example, the [Export Variables from Fragment](#) assertion).

Context variables set by the [Require WS-Addressing](#) assertion will show in the context variables tree only if a prefix is defined in the [WS-Addressing Properties](#) (a prefix is optional for this assertion). Exception: The `${<prefix>.elements}` variable is displayed in the context variables tree only if it is used later in the policy.

Searching for a Context Variable

To quickly locate a context variable, type the first few characters of its name in the Search box to display all matching variables. You can jump to a variable by clicking on the displayed name or by selecting it using the Up/Down arrow keys followed by the [Enter] key.

The following are some search tips:

- The search is not case sensitive.
- The search will match the typed text anywhere within the variable name.
- Only variable names are matched; variable values are not included in the search.
- For nested variables, only the child portion of the name is matched. For example, in Figure 19 the full name for the "contentType" variable is actually "request.contentType". However you need to search for "contentType", not "request.contentType".

Manually Adding Context Variables

In addition to the context variables set in the policy, you can also manually add other context variables to the variable tree. This allows you to view other built-in variables that are not displayed by default in the tree or to examine any custom context variables.

Context variables may be added while the debugger is running or stopped.

Note: Manually added variables are not saved in the context variable tree when the debugger is closed.

➤ *To manually add another built-in context variable:*

1. Choose the variable from the drop-down list below the context variable tree. For a description of the variables, see Context Variables.
2. Click **[Add]**. The variable is added to the tree and is shown in blue to indicate a manually added variable. The variable will have an empty value initially, until the debugger resumes.

➤ *To specify a custom context variable:*

1. Type in the name of the custom variable in the drop-down list box, with or without the "\${ }" wrapper characters. The on-screen validator will check the syntax of the variable.

2. Click **[Add]**. The variable is added to the tree and is shown in blue to indicate a manually added variable. The variable will have an empty value initially, until the debugger resumes.

➤ *To remove a manually added variable from the tree:*

- Right-click the variable and then select **Delete**.

Variables that are displayed by default (that is, those not manually added) cannot be removed.

Policy Debug Tracing

The Policy Manager has a special "trace policy" that can be invoked to help you diagnose and troubleshoot problems in a service policy, for both SOAP and non-SOAP services. When enabled, this trace policy executes after each assertion has completed within the service being debugged. The following is a partial list of the information that is passed to the trace policy for the assertion that just finished executing:

- service entity ID
- service ordinal
- policy entity ID
- policy ordinal
- assertion status

For a complete list of information available to the trace policy, see "Context Variables in Debug Trace Policy" on page 71.

IMPORTANT: Enable policy debug tracing only for troubleshooting purposes. Do not enable it for production use. As the trace policy is run for *each* assertion in the policy, performance is significantly degraded.

Debug Tracing with an Audit Sink

For a comprehensive debugging solution, you can configure an audit sink to be run in addition to a debug trace policy. This will help troubleshoot issues such as the service policy terminating unexpectedly with a serious policy exception. When a policy terminates abnormally, debug tracing also stops. The addition of an audit sink lets you take some action after the termination.

If an audit sink is configured, it will be invoked after a request has finished processing. However, be aware that the audit sink policy cannot access any of the [context variables created by the debug trace policy](#).

For more information on using audit sinks, see *Managing the Audit Sink* and *Working with the Audit Sink Policy*, both in the *in the Layer 7 Policy Manager User Manual*.

➤ *To enable policy debug tracing:*

1. Open the properties for the service being debugged.
2. In the [General] tab, select the **Enable policy debug tracing** check box.
3. Click **[OK]**. You are asked whether you wish to edit the debug trace policy.
 - Click **[Yes]** to open the trace policy for editing. You will have a chance to save any currently open policy as a new [revision](#).
 - Click **[No]** to continue working in the current policy. You can edit the trace policy later by opening [Internal Debug Trace Policy] from the Services and Policies list. **Tip:** There is a default trace policy that you can use right away without further configuration.

Once tracing is enabled, the trace policy is run every time an assertion completes in the service policy. The performance impact depends on the complexity of the trace policy, but it will likely be significant.

Tip: To allow debug tracing to access the assertions within the underlying policy fragment ("backing policy") of an encapsulated assertion, you must select the "Allow debug tracing into backing policy" check box in the "Encapsulated Assertion Configuration Properties" on page 134.

To learn more about the trace policy, see "Working with the Debug Trace Policy" on page 67.

Working with the Debug Trace Policy

A special trace policy is available to help you troubleshoot a service policy. This trace policy is enabled by selecting the **Enable policy debug tracing** check box in the [General] tab of the Service Properties in the *Layer 7 Policy Manager User Manual*. Any existing trace policy is used, otherwise one is created.

Tip: You can also use the [Service Debugger](#) to help you troubleshoot your policies. For more information, see "Working with the Service Debugger" on page 56.

When the tracing policy is enabled, it appears in the Services and Policies list on the Policy Manager interface with the name "[Internal Debug Trace Policy]", which is fixed and cannot be changed.



Figure 20: Debug Trace Policy on the interface

When enabled, the trace policy is executed once for each assertion that completes in the target policy.

The following characteristics are unique to the debug trace policy:

- There is a single trace policy shared by all published services that have tracing enabled.
- The debug trace policy is edited like a normal policy, but cannot be deleted while it is in use (i.e., enabled in the Service Properties of any published service).
- The debug trace policy can access a large number of debug-specific context variables that exist only while the trace policy is active. See "[Context Variables for the Debug Trace Policy](#)" below for details.
- The debug trace policy uses the same audit context as the policy being traced. For example, audit detail messages added during tracing will be combined with the detail messages from the target policy.
- The properties for a debug trace policy cannot be modified.
- The debug policy can optionally trace into the underlying policy fragment of an encapsulated assertion. For details, see "Encapsulated Assertion Configuration Properties" on page 134.

Aside from the above exceptions, the debug trace policy is [configured](#) and [edited](#) in similar fashion to an ordinary policy. Multiple policy [revisions](#) may be created and you may [export](#) or [import](#) the debug trace policy.

Deleting the Debug Trace Policy

When the debug trace policy is no longer required, you can delete it by right-clicking it in the Services and Policies list and selecting **Delete Policy**. Note that you cannot delete the trace policy if debug tracing is still enabled on any policy.

Tip: If you delete the debug trace policy, it will be recreated the next time policy debug tracing is enabled. However note that this will be an entirely new trace policy—it will not have access to any [policy revision history](#) from the previously deleted trace policy. Do not delete the trace policy if you wish to keep its revision history.

Understanding the Debug Trace Policy

When the debug trace policy is enabled for the first time, it is created with the simple default policy (line wraps under "Audit Details" have been added for clarity):

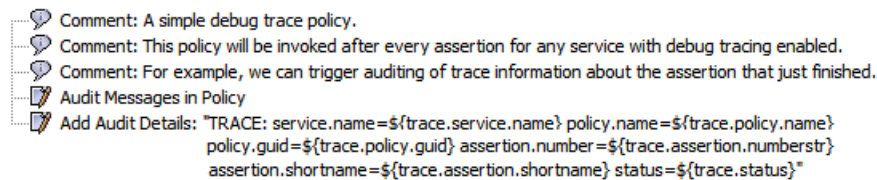


Figure 21: Debug trace default policy

The default trace policy can be used immediately, without modification. It will do the following:

1. Enables auditing with the [Audit Messages in Policy](#) assertion.
2. Adds the following details to the audit record via the [Add Audit Detail](#) assertion:

- name of the service
- name of the policy
- GUID of the policy
- number of the assertion within the policy
- name of the assertion
- status returned by the assertion

These details are retrieved from the corresponding [debug trace context variables](#), described below. You may [edit](#) the default trace policy as necessary.

More Complex Example

The following is a more complex trace policy that will collect trace information for an entire request as a batch, then email it to someone, sending no more than one email per traced request:

```

Set Context Variable: ${trace.out} = "${trace.out}
TRACE: service.oid=${trace.service.oid}
assertion.number=${trace.assertion.numberstr}
policy.guid=${trace.policy.guid}
assertion.shortname=${trace.assertion.shortname}
status=${trace.status}\n"
  
```

At Least One Assertion Must Evaluate to True

All Assertions Must Evaluate to True

Compare Expression: `${trace.final} == "true"`

Send Email Alert: `bob@acmecorp.com: subject=Debug trace for policy
body="${trace.out}"`

Continue Processing

In this more complex example, a new line beginning with "TRACE:" is appended to the `${trace.out}` context variable each time the trace policy is invoked for a request. When the debug trace is complete (`${trace.final}` returns "true"), the contents of the `${trace.out}` variable is emailed to `bob@acmecorp.com`.

Tip: Though the trace policy can be as complex and full featured as any normal service policy, it is highly recommended to keep it as short and basic as possible. Remember, the full trace policy will be executed *each time* an assertion completes in the target policy.

Saving Trace Information to a File

➤ To save the trace information to a log file:

1. Using the Manage Log Sinks task, create a new log sink with the following properties:

Name: `trace`

Description: `Save trace information to a file`

Type: `File`

Severity Threshold: `All`

Selected Categories: `Gateway Log, Audits` (hold down [Ctrl] key to select both)

2. Using the Manage Cluster-Wide Properties task, add the cluster property `log.levels` with the following line appended to the value:

`com.l7tech.server.trace.TracePolicyEvaluator.level = FINER`

3. Configure your trace policy to accumulate any desired trace information in the context variable `${trace.out}`. For example, the policy sample under "[More Complex Example](#)" above is a good example.
4. When service consumption is complete, you can find the trace log file in this directory:

`/opt/SecureSpan/Gateway/node/default/var/logs`

Security Permissions

In order to edit a Debug Trace Policy, you must have one or more roles that grant permission for:

- Managing services
- Managing policies
- Managing cluster-wide properties

These can be either predefined roles or custom roles with the appropriate permissions.

Context Variables in Debug Trace Policy

The following context variables contain values only when used in a debug trace policy, or within a policy fragment that is [included](#) in a debug trace policy. If called from any other policy, these variables will not exist and will be interpolated as blank (unless the *template.strictmode* cluster property is enforced, in which case the calling assertion will fail).

Table 12: Context variables for debug trace policy

Variable	Description
trace.service.oid	The internal object identifier of the published service with the policy currently being traced.
trace.service.name	The name of the published service with the policy currently being traced.
trace.policy.guid	The GUID of the policy containing the assertion that just executed.
trace.policy.name	The name of the policy containing the assertion that just executed.
trace.policy.version	The policy version number that is active in the policy containing the assertion that just executed.
trace.assertion.number	This is a multivalued variable that contains the full path to the traced assertion, with each position in the path as a separate value. For example, for "3.2.17 Compare Expression", this variable will contain the values "3", "2", "17".
trace.assertion.numberStr	Similar to <i>trace.assertion.number</i> above, except the full path is recorded as an assertion number; for example, "3.2.17".
trace.assertion.ordinal	The ordinal of the assertion within its policy fragment. Using the "Compare Expression" example under <i>trace.assertion.number</i> above, this will be "17".
trace.assertion.shortName	The short name of the assertion; for example "Continue Processing".

Variable	Description
trace.assertion.xml	The raw XML code for the assertion; useful for deeper inspection. IMPORTANT: Use this variable carefully, as it will further impact system performance during debug tracing.
trace.status	The assertion status code returned by the assertion that just finished. A status of "0" means the assertion succeeded. Any other status means the assertion failed.
trace.status.message	The text from the assertion status code message; for example, "Authentication Failed".
trace.request	The original request message from the policy being traced.
trace.response	The original response message from the policy being traced. Note: Take care not to modify the original request or response within the trace policy to avoid affecting the behaviour of the policy being traced. Even strictly read-only operations like XPath or schema validation may affect the exact behaviour of the original policy in subtle ways by changing how and when the XML is parsed or the data is read.
trace.var. <variableName>	Returns the contents of the <code>_\${variableName}</code> context variable from the policy being traced. The <code>_\${variableName}</code> variable can be any context variable that has been set in the policy up to that point.
trace.final	This variable is set to "true" for the final trace invocation, after the last assertion has finished for this request.
trace.out	This is a special utility variable that is empty initially. It is normally used to accumulate trace information during debug tracing.

Managing Global Resources

The *Manage Global Resources* task is used to manage resources that apply globally, such as XML schema or DTD (Document Type Definition) resources. A global resource can be referenced by an import statement in one or more [Validate XML Schema](#) assertions in a policy, or from another global resource. During runtime, the Gateway resolves a schema referenced by an import, include or redefine statement as part of the validation process. The referenced schema, or global schema, must exist in the Manage Global Resources table—and hence in the Gateway—in order for validation to proceed.

Tip: Schema dependencies (i.e., import targets) do not need to be in the Manage Global Resources table when monitoring a URL for a schema to validate ("Monitor URL for latest value" option in the Validate XML Schema assertion).

In order for the Gateway to be able to resolve external schemas when running the [Validate XML Schema](#) assertion, the import statement in the assertion schema must contain a "schemaLocation" attribute value that matches the global schema's "System ID" value.

Default Global Resources

The following global resources are present by default:

- **SOAP 1.1 and 1.2 XML Schemas:**

<http://schemas.xmlsoap.org/soap/envelope/> (SOAP 1.1)

<http://www.w3.org/2003/05/soap-envelope/> (SOAP 1.2)

<http://www.w3.org/2001/xml.xsd> (XML namespace)

- **DTDs:**

<http://www.w3.org/2001/XMLSchema.dtd> (XML Schema)

<http://www.w3.org/2001/datatypes.dtd> (XML Schema Datatypes)

You can edit and delete these resources as with other resources.

Tip: By default, an XML schema may not reference a DTD. If you wish to override this behaviour, set the *schema.allowDoctype* cluster property to "true".

➤ *To manage global resources:*

1. In the Policy Manager, select [Tasks] > **Manage Global Resources** from the Main Menu (on the browser client, from the **Manage** menu). The Manage Global Resources dialog appears.

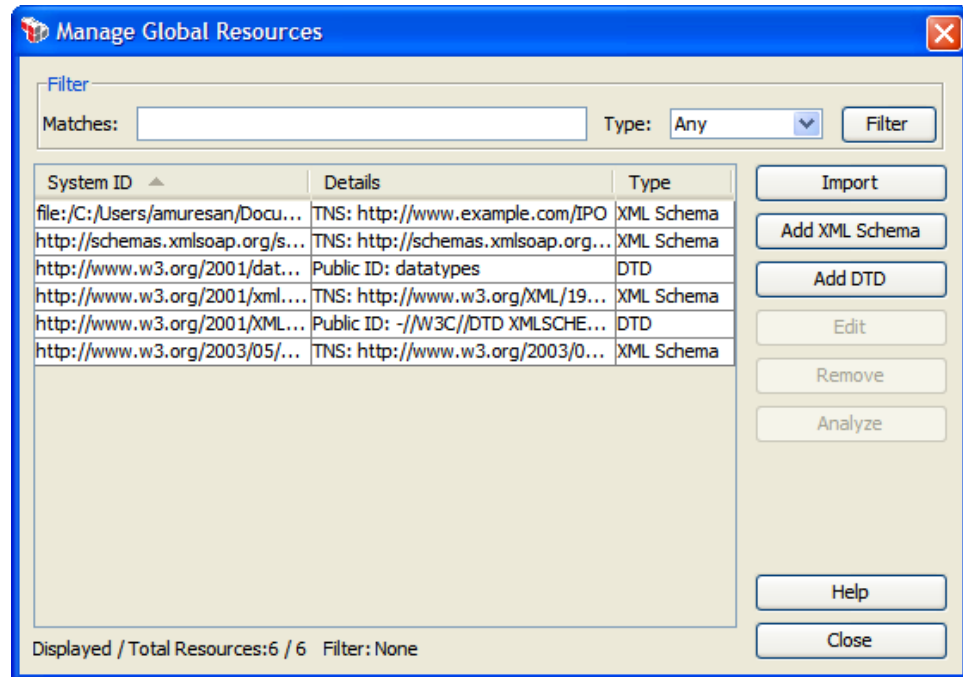


Figure 22: Manage Global Resources dialog

2. Optionally filter the list of resources displayed:
 - a. In the **Matches** field, enter the filter string. You can use a regular expression for more precise matching. **Tip:** Some characters used in a URI may need to be escaped to be used.
 - b. Select the **Type** of resources to be matched: **XML Schema**, **DTD**, or **Any**.
 - c. Click [**Filter**]. The list is filtered to display only the matching resources. **Tip:** The status message at the bottom of the dialog summarizes any filtering in effect.

To return the list to an unfiltered state, clear the **Matches** field, select type **Any**, then click [**Filter**].

3. Choose an action to perform:

To...	See
Import a global resource	"Importing a Global Resource" on page 77
Add a XML schema	"Adding a New Global Resource" on page 75
Add a DTD	"Adding a New Global Resource" on page 75
Edit a global resource	"Editing a Global Resource" on page 77

To...	See
Remove a global resource	"Deleting a Global Resource" on page 77
Analyze a global resource	"Analyzing a Global Resource" on page 84

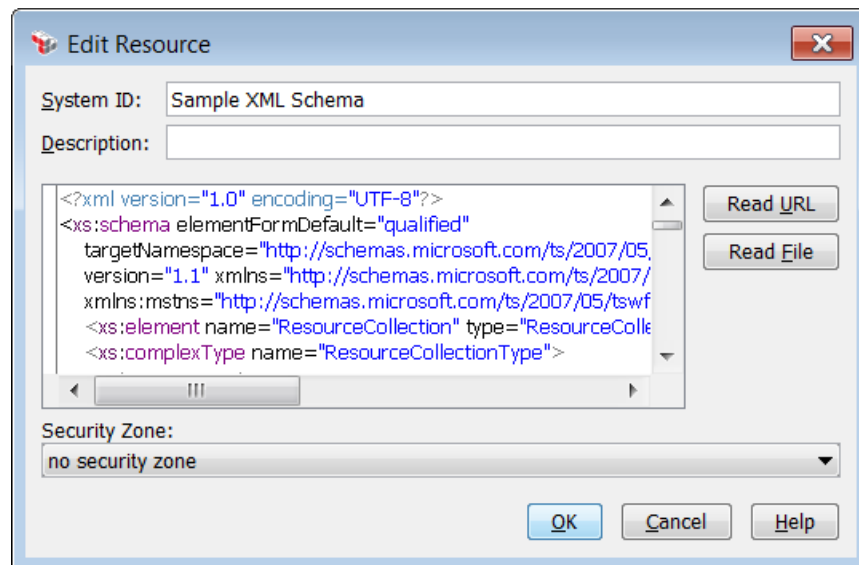
- Click **[Close]** when done.

Adding a New Global Resource

You can manually add a [global resource](#) to the Gateway at any time.

➤ To add a new global resource:

- In the Policy Manager, select **[Tasks] > Manage Global Resources** from the Main Menu. The [Manage Global Resources](#) dialog appears.
- Click **[Add XML Schema]** or **[Add DTD]**, depending on the type of resource to be added. The Edit Global Resource dialog appears.



Edit Resource

System ID: Sample XML Schema

Description:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified"
  targetNamespace="http://schemas.microsoft.com/ts/2007/05"
  version="1.1" xmlns="http://schemas.microsoft.com/ts/2007/05"
  xmlns:mstns="http://schemas.microsoft.com/ts/2007/05/tswf"
  <xs:element name="ResourceCollection" type="ResourceColle
  <xs:complexType name="ResourceCollectionType">
```

Read URL

Read File

Security Zone: no security zone

OK Cancel Help

Figure 23: Adding a Global Resource (XML Schema)

- Complete the dialog as follows:

Table 13: Global Resource settings

Setting	Description
System ID	Enter the URI that indicates the location of the resource, maximum 4096 characters. Tip: For backwards compatibility, you may enter a relative URI. However, a warning will appear.

Setting	Description
	<p>For global XML schemas, this value relates to the "schemaLocation" attribute in the schema import, include or redefine statement that references the global schema. For example, schema A contains an import statement that references global schema B. The value in the "schemaLocation" attribute in schema A is <i>http://example.org/account.xsd</i>:</p> <pre><s:import namespace="http://www.acme.com/schemas/account" schemaLocation="http://example.org/account.xsd" /></pre> <p>And the System ID of schema A is <i>http://example.org/main.xsd</i></p> <p>In order to connect schema A and global schema B in the Gateway, the System ID of global schema B must be <i>http://example.org/account.xsd</i>.</p> <p>Note: The System ID field is case sensitive. Entering a value with the incorrect case will cause connection problems.</p>
Public ID (DTD only)	<p>For DTD resources, enter the public identifier for the resource. The valid characters are [a-zA-Z0-9-'()+,./:=?;!*\$@\$_%], with a maximum 4096 characters. Any white space will be replaced with a single space character (#x20), and leading and trailing spaces will be removed.</p>
Description	<p>Optionally enter a "friendly" description of the global resource, with a maximum 255 characters.</p>
[code window]	<p>In the code window, type or paste the XML schema or DTD content. The XML Editor is available in the code window to help you search, parse, format, or comment as required.</p>
[Read URL]	<p>Use this to retrieve an XML schema or DTD resource from a URL.</p>
[Read File]	<p>Use this to retrieve an XML schema or DTD resource from a file.</p>
Security Zone	<p>Optionally choose a security zone. To remove this entity from a security zone (security role permitting), choose "No security zone".</p> <p>For more information about security zones, see Understanding Security Zones in the <i>Layer 7 Policy Manager User Manual</i>.</p> <p>Note: This control is hidden if either: (a) no security zones have been defined, or (b) you do not have Read access to any security zone (regardless of whether you have Read access to entities inside the zones).</p>

- Click **[OK]** when done. The new resource is added to the Manage Global Resources table.

Editing a Global Resource

➤ *To edit an existing global resource:*

1. In the Policy Manager, select **[Tasks] > Manage Global Resources** from the Main Menu. The [Manage Global Resources](#) dialog appears.
2. In the table, select the global resource to edit and then click **[Edit]**. The Edit Global Resource dialog appears.
3. Modify the resource details as necessary. See "Adding a New Global Resource" on page 75 for information about each field.

Deleting a Global Resource

You can delete a global resource using the [Manage Global Resources](#) task.

➤ *To delete an existing global resource:*

1. Select **[Tasks] > Manage Global Resources** from the Main Menu. The [Manage Global Resources](#) dialog appears.
2. In the table, select the global resource(s) to delete. You can delete multiple resources by holding down the [Ctrl] key while selecting.
3. Click **[Remove]** and then click **[OK]** to confirm. The resource(s) are removed from the Manage Global Resources table.

Note: You are warned if the resource being deleted is used in policies or is registered for hardware use. You can acknowledge the warning and continue with the deletion. In this case, you should adjust the affected policies or edit the `schema.hardwareTargetNamespaces` cluster property afterward. Note that no warning is given if the schema being deleted is referenced from another schema that is unused. To detect these references, use the Analyze feature in the Manage Global Resources task. For more information, see "Analyzing a Global Resource" on page 84.

Importing a Global Resource

You can import resources and dependencies into the Manage Global Resources table using the import wizard. You can add resources individually or in bulk.

➤ *To import a global resource:*

1. In the Policy Manager, select **[Tasks] > Manage Global Resources** from the Main Menu. The [Manage Global Resources](#) dialog appears.
2. Click **[Import]**. The Resource Import Wizard appears.

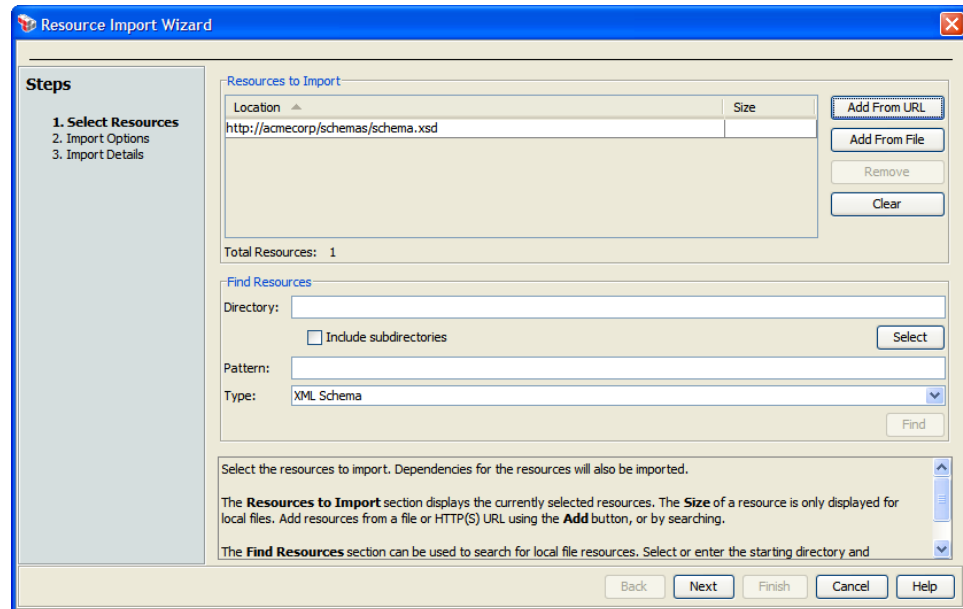


Figure 24: Resource Import Wizard

3. Complete the wizard as described below

Step 1: Select Resources

In this step, specify the resources to import. You can either add them directly from a URL or file, or search within a specified directory.

➤ To add a resource directly, do either of the following:

- Click **[Add from URL]** and enter the URL. **Tip:** To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click **[HTTP Options]** to open the Manage HTTP Options dialog.
- Click **[Add from File]** and browse to the file containing the resource to add.

➤ To remove resources from the list, do either of the following:

- Select a resource to remove and then click **[Remove]**. The resource is removed from the list.
- Click **[Clear]**. This removes *all* the resources from the list.

➤ *To add resources by searching within a directory:*

1. Enter the path in the **Directory** field or click **[Select]** to browse for the directory.
2. Select **[Include subdirectories]** to include the subdirectories of the specified directory in the search. Otherwise, only the specified directory itself is searched.
3. Optionally specify a **Pattern** to match. If you do not specify a pattern, all files with the following extensions are located: ***.xsd** or ***.dtd**.
4. Select the **Type** of resource to match using the drop-down list: **XML Schema** or **DTD**.
5. Click **[Find]**. The matching resources are added to the "**Resources to Import**" table.
6. Review the resources in the "**Resources to Import**" table carefully to ensure that the correct resources have been identified. To make corrections:
 - Use either of the **[Add...]** buttons to manually specify the resource to import. (See "[To add a resource directly](#)" from above.)
 - Use **[Remove]** to remove a single resource from the list.
 - Use **[Clear]** to clear all files from the list to start over again.

Step 2: Import Options

In this step, indicate how the wizard should respond to any of the following issues during resource importing:

- If a resource dependency's target namespace matches multiple existing XML Schemas:
 - **Ask:** Allows you to choose a resolution each time this issue occurs. This setting is the default.
 - **Manually select an existing XML Schema:** Lets you select an existing XML Schema to use; does not import the XML Schema belonging to the dependency.
 - **Import the XML Schema:** Always import the XML Schema associated with the dependency. This will create a new resource with the same target namespace.
 - **Don't import the XML Schema:** Don't import the dependency or any resources that depend on it. This represents a failure case as the main schema will not be imported.
- If a resource dependency's public identifier matches multiple existing resources:

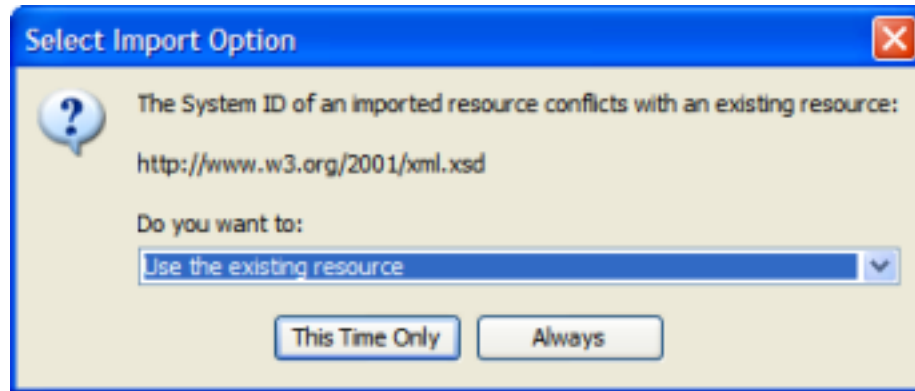
- **Ask:** Allows you to choose a resolution each time this issue occurs. This setting is the default.
- **Manually select an existing resource:** Lets you select the existing resource to use for the dependency.
- **Import the duplicate resource:** Always import the resource associated with the dependency. This will create a new resource with the same target namespace.
- **Don't import the duplicate resource:** Don't import the dependency or any resources that depend on it.
- If an imported resource's system identifier conflicts with an existing system identifier:
 - **Ask:** Allows you to choose a resolution each time this issue occurs. This setting is the default.
 - **Use the existing resource:** The existing resource is used; a new resource is not created for this dependency.
 - **Update the existing resource:** Replace the existing system identifier with the one from the imported resource.
 - **Don't import the conflicting resource:** Exclude the conflicting resource from the import. Resources that depend on the resource will not be imported. This is a failure case for the import.
- If an imported XML Schema's target namespace matches an existing value:
 - **Ask:** Allows you to choose a resolution each time this issue occurs. This setting is the default.
 - **Use the existing XML Schema:** Use the existing XML Schema; the contents of the imported dependency are ignored.
 - **Update the existing XML Schema:** Update the existing XML Schema with the import. This will preserve the current URI and use the content of the imported schema.
 - **Replace the existing XML Schema:** Replace the existing XML Schema with the one from the import. This will use the URI and content from the import. Any existing XML Schemas that reference the dependency should be updated to use the new URI.
 - **Import the XML Schema:** Import the incoming XML Schema as is. This will use the URI from the import. This will create a new resource.
 - **Don't import the matching resource:** Exclude the resource with the matching target namespace from the import. Resources that depend on the resource will not be imported.

- If an imported resource's public identifier duplicates an existing value:
 - **Ask:** Allows you to choose a resolution each time this issue occurs. This setting is the default.
 - **Use the existing resource:** Use the existing resource; the contents of the imported dependency are ignored.
 - **Update the existing resource:** Update the resource with the contents from the import. This will preserve the current URI.
 - **Replace the existing resource:** Replace the existing resource with the one from the import. This will use the URI and content from the import. Any existing resources that reference the dependency should be updated to use the new URI.
 - **Import the duplicate resource:** Always import the resource, using the import URI. This will result in duplicate resources.
 - **Don't import the duplicate resource:** Exclude the duplicate resource from the import. Resources that depend on the resource will not be imported.
- If a resource dependency cannot be found, or is invalid:
 - **Ask:** Allows you to choose a resolution each time this issue occurs. This setting is the default.
 - **Manually resolve or fix the dependency:** Displays a dialog that lets you manually resolve or update the dependency.
 - **Don't import the invalid resource:** Exclude the invalid resource from the import. Also exclude all other resources that depend on this resource.

When '**Ask**' is specified in any of the settings, you will be prompted to respond to any issues that need attention.

Example 1:

The following dialog is displayed when there is a conflicting System ID:

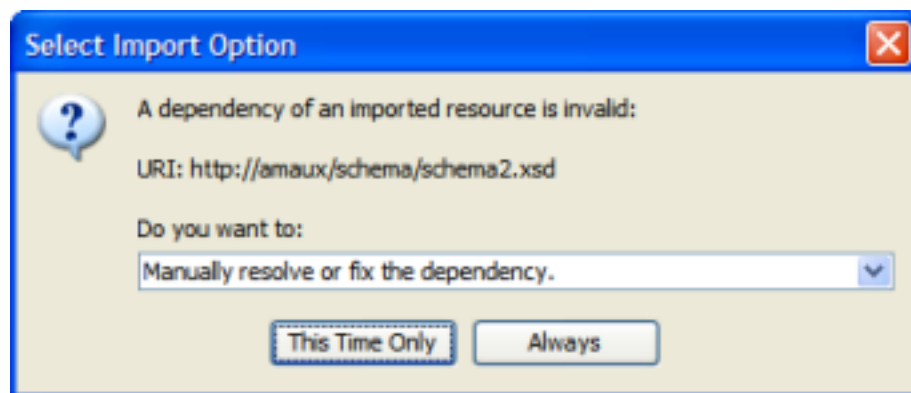


Select a resolution from the drop-down list, then specify whether:

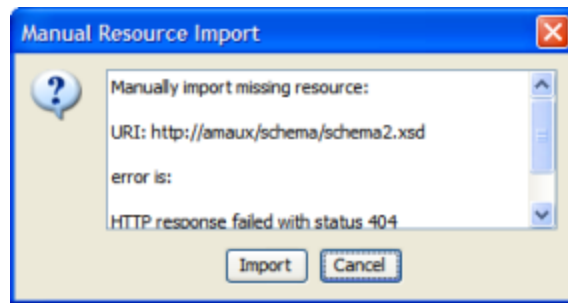
- **[This Time Only]:** Use the selected action only for this occurrence of the conflict. When another similar conflict occurs, you will be asked again how to resolve it.
- **[Always]:** Use the selected action for all the conflicts of this type. You will not be prompted for a resolution if another similar conflict occurs during this import. This is the same as configuring the choice in the wizard step.

Example 2:

The following dialog is displayed when a dependency of an imported resource is invalid:



Select a resolution from the drop-down list, as in the previous example. If you choose to manually resolve, you will be given a chance to fix the dependency in the following dialog:



Click [**Import**] to select the dependency from a URL or a file.

Step 3: Import Details

This step summarizes what actions will be taken when finishing the import. If any resource could not be created due to an error, the details will be shown here.

Import Summary

The table lists all the resources involved in the import:

- **System ID:** The URI of the resource being imported.
- **Details:** The target namespace (for XML Schemas) or the public identifier of the resource (for DTDs).
- **Type:** Whether the resource is a **XML Schema** or **DTD**.
- **Status:** The resource import status.
- **Action:** What will be done with the resource:
 - *Ignore:* The resource will not be imported.
 - *Update:* The resource will update an existing global resource.
 - *Create:* A new global resource will be created for the resource.
- **[View]:** Use this to view the contents of a resource.
- **[Remove]:** Use this to remove a resource from the import list. Any resource that depends on this resource will also be removed.
- **[Update]:** Use this to update the system identifiers of any matching resources before importing, if this is possible. This will alter all resources being imported for consistency; it does not act on any resource that may be selected in the import list. Enter the current and updated System Identifier prefixes as prompted. **Note:** The updated system identifiers must be absolute URIs.

You cannot update system identifiers if:

- The updated system identifier conflicts with an existing global resource.
- The content of a resource must be updated due to the change and the dependency is not an XML Schema.

Resource Details

This section displays details about the currently selected resource:

- **System ID:** The URI of the resource.
- **Description:** A description of the resource.
- **Status:** The resource import status.
- **Status Detail:** A more verbose description explaining why the particular status was assigned.
- **Dependencies:** Shows the dependencies for the resource. You can choose to **Show all dependencies**, **Show only direct dependencies**, or **Show only transitive dependencies** (these are shown in italicized text). **Note:** Transitive dependencies of *existing* resources are not displayed here.
 - *Uses:* The resource(s) that the selected resource uses.
 - *Used by:* The resource(s) in which the selected resource is used.

Note: The **[Finish]** button is activated only when there are resources that can be imported. If it is not possible to import any resources, use the **[Back]** button to return to previous steps to make corrections, or click **[Cancel]** to exit the wizard and try again later.

Analyzing a Global Resource

You can analyze any resources from the Manage Global Resources table. During the analysis, you can do the following:

- View details about the resource
- Validate the resource
- View dependencies for the resource: what the resource uses and what the resource is used by
- Reset the resource to its default value (contents and system identifier), if available

➤ *To analyze global resource:*

1. In the Policy Manager, choose **[Manage] > Manage Global Resources** from the Main Menu. The [Manage Global Resources](#) dialog appears.

2. Choose one or more resource to analyze (hold down the [Ctrl] key to select multiple resources).
3. Click **[Analyze]**. The Analyze Global Resources dialog appears.

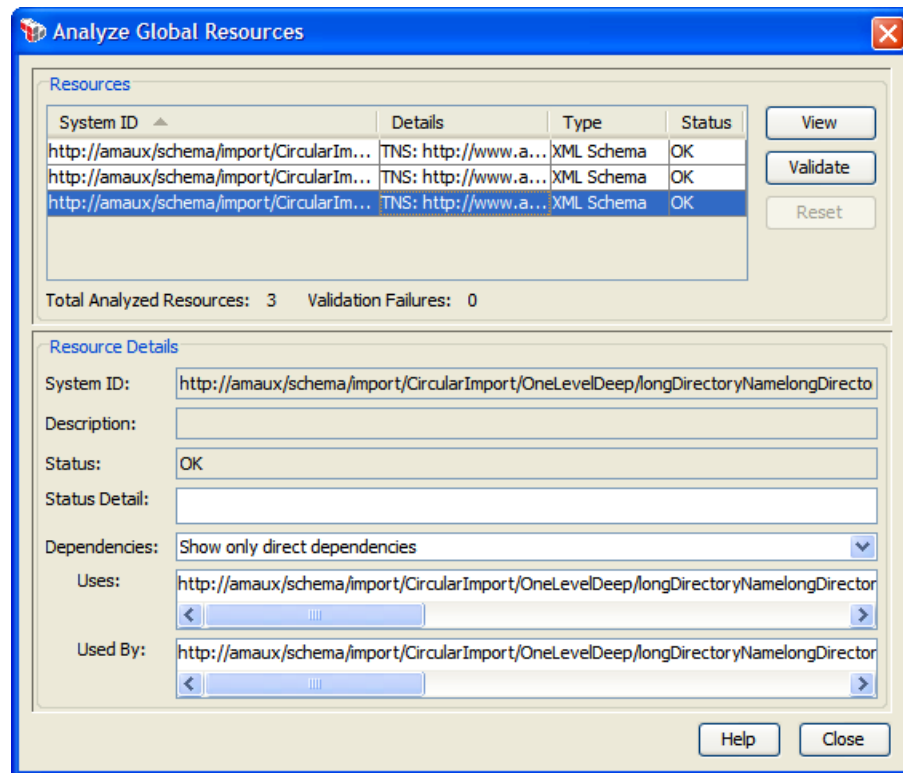


Figure 25: Analyze Global Resource dialog

4. The following table describes the dialog in detail:

Table 14: Analyze Global Resource dialog

Setting	Description
Resources	<p>The Resources table at the top lists the resources that were selected before the [Analyze] button was used and any dependencies:</p> <ul style="list-style-type: none"> • System ID: The URI for the selected resource(s), as entered in "Adding a New Global Resource" on page 75. • Details: The target namespace (for XML Schemas) or the public identifier of the resource (for DTDs). • Type: The resource is either a XML Schema or DTD. • Status: The status for the resource.
[View]	Choose this to display the content of the selected resource.
[Validate]	Choose this to validate all the displayed resources. The number of

Setting	Description
	<p>validation failures is displayed next to "Validation Failures" below the list of resources. The text "(not validated yet)" indicates that validation has not yet been performed.</p> <p>Note: For resources of type "DTD", validation will occur only if the resource is used.</p>
[Reset]	<p>Choose this to reset the system identifier and/or contents of the selected resource back to its default values. If resetting both is possible, you will be prompted to choose either or both to reset. If resetting neither is possible, then this button will not be activated.</p> <p>The non-default contents and /or the system identifier will be reflected in the 'Status' for the resource. For example, the "Status" for resources that have been reset might look like this:</p> <p><i>OK (System ID and content modified from default value)</i> <i>Failed (System ID modified from default value)</i> <i>OK (content modified from default value)</i></p>
Resource Details	<p>This section displays details about the resource:</p> <ul style="list-style-type: none"> • System ID: Displays the URI for the resource. • Status: Whether the resource has passed validation. • Status Detail: The target namespace (for XML Schemas) or the public identifier of the resource (for DTDs). If the system identifier for a resource can be reset, then the default system identifier is also displayed in the Status Detail for the resource.
Dependencies	<p>This section shows the dependencies for the resource. You can choose to show all dependencies, only direct dependencies, or only transitive dependencies (shown in italicized text). Note: Only dependencies within the global resources are displayed. Schemas that are configured within policies will be not be included.</p> <ul style="list-style-type: none"> • Uses: The resource(s) that the selected resource uses. • Used by: The resource(s) in which the selected resource is used. <p>IMPORTANT: If a resource has a Status of "Fail", the Dependencies section does not display the other dependencies that the resource may have.</p>

5. Click [**Close**] when done.

Managing UDDI Registries

The Gateway can publish a web service by using a WSDL located in a UDDI (Universal Description, Discovery and Integration) registry. The Gateway supports the following registry types:

- CentraSite ActiveSOA
- CentraSite Governance Edition
- CentraSite UDDI Enterprise Edition
- Systinet UDDI Registry
- Generic UDDI v3

The *Manage UDDI Registries* task is used to create, remove, or edit UDDI registries.

Note: If the Gateway only has HTTPS endpoints, ensure that the specific UDDI registry has been configured to trust the Gateway's SSL certificate. For assistance, please contact your administrator for the UDDI registry.

➤ To manage UDDI registries:

1. In the Policy Manager, select **[Tasks] > Manage UDDI Registries** from the Main Menu (on the browser client, from the **Manage** menu). The Manage UDDI Registries dialog appears.

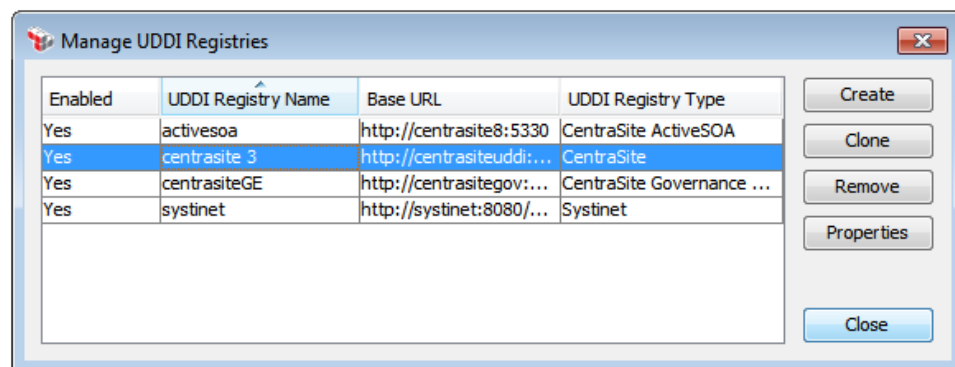


Figure 26: Manage UDDI Registries dialog

2. The following table describes each column; these are set in the UDDI registry's [properties](#):

Table 15: Manage UDDI Registries columns

Column	Description
Enabled	Indicates whether the UDDI registry is enabled. If disabled, the UDDI registry is no longer usable, but it will continue to be displayed in drop-

Column	Description
	down lists showing all registries.
UDDI Registry Name	A name that identifies the UDDI registry. This name must be unique.
Base URL	The URL for the UDDI registry. This URL is unique because it is not possible to register the same UDDI registry more than once.
UDDI Registry Type	The type of UDDI registry that was published.

3. Select a task to perform:

Table 16: Manage UDDI Registries tasks

To...	Do this...
Add a new UDDI registry	<ol style="list-style-type: none"> 1. Click [Create]. 2. Complete the "UDDI Registry Properties" on page 89.
Clone an existing UDDI registry	<ol style="list-style-type: none"> 1. Select the registry to clone. 2. Click [Clone]. 3. Edit the "UDDI Registry Properties" on page 89 as required.
Remove a UDDI registry	<ol style="list-style-type: none"> 1. Select the registry to remove. 2. Click [Remove]. 3. Click [OK] to confirm. The UDDI registry is removed from the Gateway's records. <p>Notes: (1) While it is possible to remove a UDDI registry that is currently in use, information published to that registry will not be removed. You must manually delete the various published items or use the native UDDI interface of the registry. (2) When removing a UDDI registry, previously published information will be deleted from the Gateway (the UDDI registry itself is not affected).</p>
View or edit the properties of a UDDI registry	<ol style="list-style-type: none"> 1. Select the UDDI registry to view. 2. Click [Properties]. See "UDDI Registry Properties" on page 89 for details.

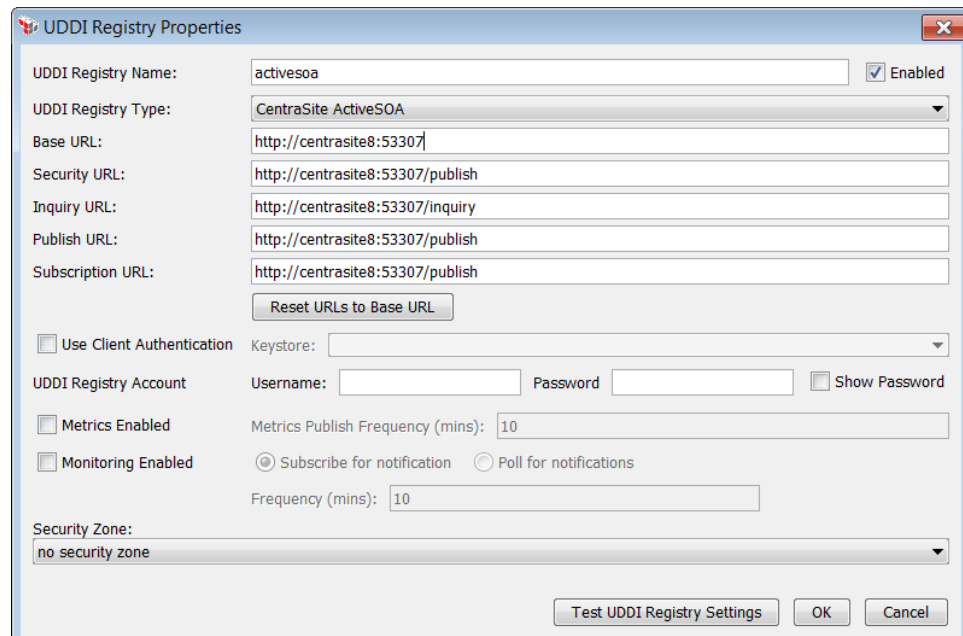
4. Click [**Close**] when done.

UDDI Registry Properties

When creating or viewing details about a [UDDI registry](#), the UDDI Registry Properties dialog is displayed. This dialog lets you configure a UDDI registry to be recognized by the Gateway.

➤ To access the properties for a UDDI registry:

1. Run the [Manage UDDI Registries](#) task.
2. Select a registry from the list and then click **[Properties]**. You can also click **[Create]** to enter the properties for a new UDDI registry. The UDDI Registries Properties appear.



The UDDI Registry Properties dialog box contains the following fields and controls:

- UDDI Registry Name:** Text field with value "activesoa".
- UDDI Registry Type:** Dropdown menu with value "CentraSite ActiveSOA".
- Base URL:** Text field with value "http://centrasite8:53307".
- Security URL:** Text field with value "http://centrasite8:53307/publish".
- Inquiry URL:** Text field with value "http://centrasite8:53307/inquiry".
- Publish URL:** Text field with value "http://centrasite8:53307/publish".
- Subscription URL:** Text field with value "http://centrasite8:53307/publish".
- Reset URLs to Base URL:** Button.
- Use Client Authentication:** Check box (unchecked).
- UDDI Registry Account:**
 - Keystore:** Dropdown menu.
 - Username:** Text field.
 - Password:** Text field.
 - Show Password:** Check box (unchecked).
- Metrics Enabled:** Check box (unchecked).
- Monitoring Enabled:** Check box (unchecked).
- Metrics Publish Frequency (mins):** Text field with value "10".
- Subscribe for notification:** Radio button (selected).
- Poll for notifications:** Radio button (unselected).
- Frequency (mins):** Text field with value "10".
- Security Zone:** Dropdown menu with value "no security zone".
- Buttons:** "Test UDDI Registry Settings", "OK", and "Cancel".

Figure 27: UDDI Registries Properties dialog

3. Configure the properties as follows:

Table 17: UDDI Registries settings

Setting	Description
UDDI Registry Name	Enter a name to identify the UDDI registry. This name must be unique.
[Enabled]	<p>Select this check box to enable the UDDI registry. Clear the check box to disable the registry.</p> <p>Tip: A disabled registry will not appear in searches where the UDDI registry serves as the "source" for an entity (for example, when searching for a UDDI registry from the Publish SOAP Web Service</p>

Setting	Description
	Wizard). However, a disabled registry will still appear in the Publish to UDDI Settings dialog and the [UDDI] tab under Service Properties. If the settings in these dialogs need to be changed for a disabled registry, it will not be possible until the registry is re-enabled..
UDDI Registry Type	Select the type of UDDI registry from the drop-down list. If you are not using one of the listed registry types, select Generic . Note: If the 'Generic' UDDI registry type is chosen, certain features may not be available (for example, the collection of metrics because the Gateway will have no prior knowledge of the registry).
Base URL	Enter the base URL for the UDDI registry. The Policy Manager will automatically copy this URL to all the other URL fields by default. The Base URL is required.
Security URL Inquiry URL Publish URL Subscription URL	These URLs are automatically populated when the Base URL is first entered. If any of these URLs differ from the Base, edit as necessary. If the 'Generic' UDDI registry type was selected, you must manually complete each field. Tip: You can quickly revert all these URLs back to the Base by clicking [Reset URLs to Base URL] .
[Reset URLs to Base URL]	Click this to update the URL fields with the Base URL. If the UDDI registry type is not 'Generic', then the specific relative parts of the URL is also auto populated.
[Use Client Authentication]	Select this check box to present a certificate to the server during the SSL handshake, if one is requested. Clear this check box to never present a certificate, even if one is requested. Note that access may be denied in this case.
[Keystore]	From the drop-down list, select the keystore from which to retrieve the certificate. This is available only if the Use Client Authentication check box is selected.
UDDI Registry Account	Enter the Username and Password to access the UDDI registry. Note: Although you may enter the actual password here, it is recommended that you use a secure password reference instead. To do this, define your password using the Manage Stored Passwords task and then reference it here using the <code>\${secpass.<name>.plaintext}</code> context variable.
[Metrics Enabled]	Select this check box to enable the collection of metrics data. This is available only for the UDDI registry type "CentraSite ActiveSOA".
Metrics Publish Frequency	The interval in minutes between successive publications of metrics data.
[Monitoring Enabled]	This setting allows monitoring to be enabled or disabled at a UDDI

Setting	Description
	<p>registry level. Monitoring may be enabled or disabled for Individual published services, but the Monitoring Enabled check box affects all services using this UDDI registry.</p> <p>Note: The Monitoring Enabled check box is available only when a Subscription URL is supplied.</p>
[Subscribe for notification]	<p>When monitoring is enabled, select this option to subscribe to be notified about changes to the UDDI. This is a global setting that affects how the UDDI is monitored. When this option is selected, the UDDI registry will notify the Gateway about changes via the UDDI notification internal service. This is an asynchronous notification from UDDI.</p> <p>To enable monitoring by subscription, the following must be done:</p> <ol style="list-style-type: none"> 1. A "UDDI Notification" internal service has been published to the Gateway. 2. The "UDDI Notification" internal service is published to the UDDI registry. <p>If the Gateway being published contains only HTTPS endpoints, ensure that the UDDI registry to which you are publishing has been configured to trust the Gateway's SSL certificate.</p> <p>Note: When subscribing to receive notifications, a WSDL document may need to be downloaded. The maximum size of this document is controlled by the <i>wsdlDownload.maxSize</i> cluster property.</p>
[Poll for notifications]	<p>When monitoring is enabled, select this option to automatically check the UDDI for notifications after a specified time period.</p> <p>Note: When polling for notifications, a WSDL document may need to be downloaded. The maximum size of this document is controlled by the <i>wsdlDownload.maxSize</i> cluster property.</p>
Frequency	<p>If [Polling for notifications] was selected, specify the how frequently to check for notifications. The default is every 10 minutes.</p>
[Test UDDI Connection]	<p>Click this button to test the connection to the UDDI Registry. If credentials were supplied, they will be used when trying to connect to the registry, otherwise the connection attempt will use no credentials.</p> <p>The test involves looking up a well known UDDI tModelKey: <i>uddi:uddi.org:specification:v3_policy</i>, which also validates that the registry is a V3 registry.</p>
Security Zone	<p>Optionally choose a security zone. To remove this entity from a security zone (security role permitting), choose "No security zone".</p> <p>For more information about security zones, see Understanding Security Zones in the <i>Layer 7 Policy Manager User Manual</i>.</p> <p>Note: This control is hidden if either: (a) no security zones have been</p>

Setting	Description
	defined, or (b) you do not have Read access to any security zone (regardless of whether you have Read access to entities inside the zones).

Note: If you change monitoring methods (for example, from monitoring to polling or vice versa) or disable/enable monitoring, this will cause an update of each service that is monitoring a Business Service in the modified UDDI registry.

- Click **[OK]** when done.

Publish to UDDI Settings

The Publish to UDDI Settings dialog is used to publish information to a UDDI registry. You can do the following:

- Publish a Gateway WSDL to a UDDI registry
- Publish a Gateway endpoint as a BindingTemplate in a Business Service in the UDDI Registry
- Overwrite an existing Business Service in the UDDI registry with corresponding WSDL information from the Gateway

Notes: (1) If any Gateway endpoint information (for example, cluster hostname or port settings) is changed after the Gateway has published to UDDI, the Gateway will automatically update UDDI so that it contains the correct gateway endpoint URLs. You can control this behaviour using the `uddi.auto_republish` cluster property. By default, this property is "true" enables auto update. Set it to "false" to disable the automatic update. (2) If service entity ID resolution is disabled in the Service Resolution Settings dialog, then any service URLs published to UDDI by the Gateway will not resolve if consumed.

➤ *To configure settings for publishing to UDDI:*

- Do either of the following:
 - Right-click a web service under the Services and Policies list and then select **Publish to UDDI**.
 - Select **[File] > Publish to UDDI** from the Main Menu.

The Publish to UDDI Settings dialog appears. This dialog organizes the settings across these tabs: **Service**, **WS-Policy**, and **Metrics**.

Configuring the [Service] tab

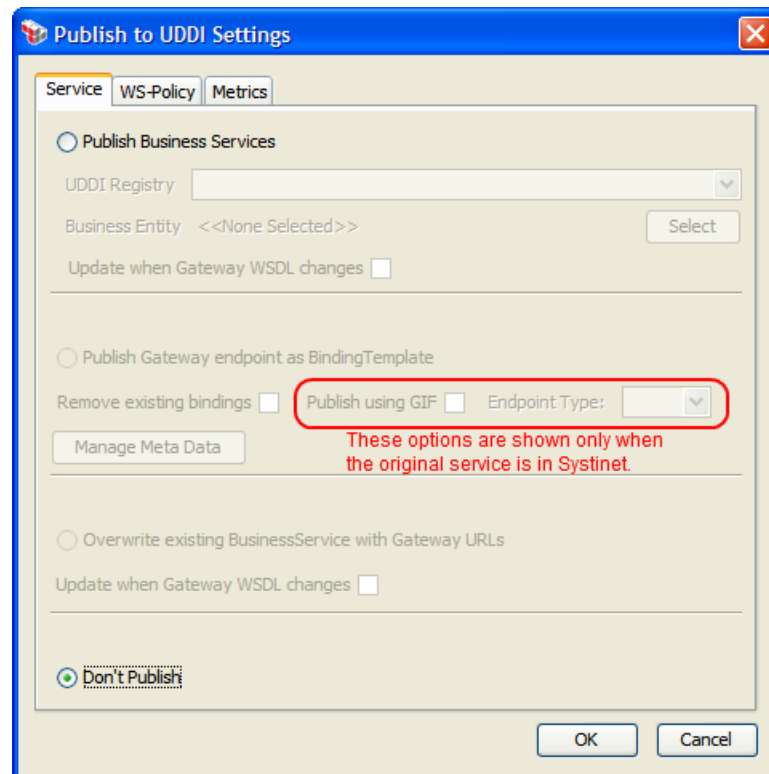


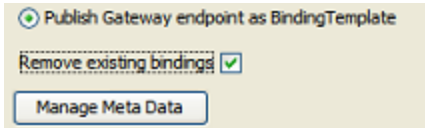

Figure 28: Publish to UDDI Settings - [Service] tab

The [Service] tab lets you publish the Gateway's WSDL as a 'proxy binding Template' Business Service to a UDDI registry. You can also use it to update an existing Business Service with a new proxy bindingTemplate with a valid keyReference attached for a Gateway endpoint. The options that are available depend on whether the published service was created from a UDDI registry.

Publishing establishes a link between the Gateway and the UDDI registry. Once a publish action has been performed, you cannot perform another publish unless you use the **[Don't Publish]** option in Figure 28 to reverse the publish action. However, you can still update the Gateway endpoint, which is the proxy bindingTemplate in the service, by using the **[Manage Meta Data]** option.

If any meta data is added to a BusinessService or bindingTemplate published by the CA API Gateway, that meta data will be preserved should the Gateway need to update the UDDI.

Table 18: Publish to UDDI Settings - [Service] tab

Setting	Description
Publish Business Services	<p>This option publishes the WSDL from the Gateway as Business Services in a UDDI. Only SOAP with HTTP endpoints from the Gateway's WSDL are published to UDDI. Note: If the Gateway WSDL contains more than one wsdl:service, then more than one UDDI Business Service will be created. The wsdl:service maps 1:1 to a UDDI Business Service.</p> <p>Complete the following:</p> <ul style="list-style-type: none"> • UDDI Registry: From the drop-down list, select the UDDI registry to publish to. The registry must be configured in the Managing UDDI Registries task to appear in this list. • Business Entity: Click [Select] and search for the destination Business Entity. For more information, see "Searching the UDDI Registry" on page 48. • Update when Gateway WSDL changes: Select this check box to have the Gateway update the UDDI when the Gateway's WSDL changes. These changes may arise from manual user edits, from refreshing the WSDL, or from a UDDI notification causing the WSDL to be redownloaded.. <p>This setting may be changed later, after publishing has occurred.</p> <p>After publishing, all the URLs in this Business Service in the UDDI will point to the Gateway.</p>
Publish Gateway endpoint as BindingTemplate <p><i>Available only when the Gateway has a record of the 'original' service in UDDI and the published service is not under UDDI control: Service Properties -> [UDDI] tab -> uncheck [WSDL under UDDI control]</i></p>	<p>Once this publishing action is performed, the WSDL under UDDI control check box will be disabled; it will be re-enabled only when the [Don't Publish] action is taken.</p> <p>Publishing a Gateway Endpoint</p> <p>You can indicate whether existing bindings should be removed during publishing.</p>  <ul style="list-style-type: none"> • Remove existing bindings: Select this check box to have the Policy Manager remove all bindings contained in the original Business Service. <p>GIF Publishing</p>  <ul style="list-style-type: none"> • Publish using GIF: When the original service is from a Systinet UDDI, you can choose to publish the Gateway

Setting	Description
	<p>endpoint according to the GIF (Governance Interoperability Framework) specification. This option is available only when the WSDL is not under UDDI control.</p> <p>Tip: Endpoints previously published without using this option are not GIF compliant. To make them compliant, you must first unpublish the endpoint (using the "Don't Publish" option), then re-publish.</p> <ul style="list-style-type: none"> • Endpoint Type: If more than one endpoint type is available, select which one to use when publishing using GIF.
Publish Gateway endpoint as BindingTemplate (<i>cont'd</i>)	<ul style="list-style-type: none"> • Manage Meta Data: This opens the Manage Meta Data dialog to manage the list of keyedReferences. Use it to add, edit, or remove keyed References. For more information on this dialog, see "Managing Meta Data" on page 99. <p>Tip: It is possible to change the meta data after the endpoint has been published. Modification of the meta data will cause UDDI to be updated.</p> <p>Note: This option cannot remove or edit the existing keyedReferences attached to the bindingTemplate in the Business Service on the UDDI registry. This is because the [Remove] and [Edit] options can only be used on keyedReference entries that were entered using [Add].</p>
Overwrite existing BusinessService with Gateway URLs <i>Available only when the Gateway has a record of the 'original' service in UDDI.</i>	<p>This option updates the entire Business Service to point to the Gateway cluster hostname for all URLs. Note: This overwriting cannot be undone. If the Business Service contains any non SOAP bindings, then they are not removed. Only SOAP + HTTP bindings are removed.</p> <p>The information that is published to UDDI is taken from the Gateway's WSDL.</p> <ul style="list-style-type: none"> • Update when Gateway WSDL changes: Select this check box to have the Gateway update the UDDI when the Gateway's WSDL changes. <p>IMPORTANT: Once this publishing action is performed, the WSDL under UDDI control check box in the [UDDI] tab of the service properties will be permanently disabled. Using the [Don't Publish] action will remove all the published bindings.</p>
Don't Publish	<p>This option reverses the publishing effects of the [Publish Business Services] and [Publish Gateway endpoint as BindingTemplate] actions. It will not reverse the overwriting made by the [Overwrite existing BusinessService with Gateway URLs] action, apart from deleting the published bindings.</p> <p>Once the [Don't Publish] action has been used to reverse a publishing action, the other three actions are once again available.</p>

Publishing Status

After a publishing action from Table 18 is performed, the status is shown in the [Service] tab, next to the action selected:

Table 19: [Service] tab - Publishing Status

Status	Description
Published	The information has been successfully published to the UDDI registry.
Publishing	The information is actively being published or updated to the UDDI registry.
Publish failed x times. Set to retry	The publish failed. See Gateway Audit Events or Viewing Logs for details. The publish will be retried if the retry attempts is less than the <i>uddi.wsd/publish.maxretries</i> cluster property.
Cannot publish. Tried x times. Please select 'Don't Publish' to retry	Unable to publish to the UDDI registry after exhausting the maximum number of times in the <i>uddi.wsd/publish.maxretries</i> cluster property. To clear this status to try again, run the [Don't Publish] action first.
Deleting	The information is being actively deleted from the UDDI registry.
Delete failed x times. Set to retry	The delete failed. The delete will be retried if the retry attempts is less than the <i>uddi.wsd/publish.maxretries</i> cluster property.
Cannot delete. Tried x times. Please select 'Dont Publish' to retry	Unable to delete from the UDDI registry after exhausting the maximum number of times in the <i>uddi.wsd/publish.maxretries</i> cluster property. To clear this status to try again, run the [Don't Publish] action first.

Configuring the [WS-Policy] tab

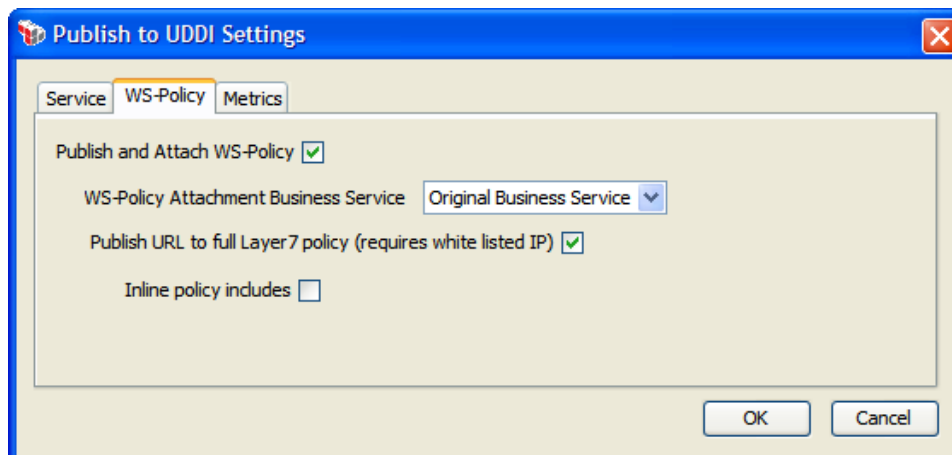


Figure 29: Publish to UDDI Settings - [WS-Policy] tab

This tab is used to configure publishing settings for the WS-Policy.

Table 20: Publish to UDDI Settings - [WS-Policy] tab

Setting	Description
Publish and Attach WS-Policy	Select this check box to publish the WS-Policy to the UDDI registry. The policy can be attached to the original business service or to all published business services.
<i>The following settings are available only when the Publish and Attach WS-Policy check box is selected:</i>	
WS-Policy Attachment Business Service	<p>From the drop-down list, select the target for the WS-Policy attachment. The options that are available depend on the [Service] tab and whether there is an "Original UDDI Business Service" selected:</p> <ul style="list-style-type: none"> • If a service has a selected "Original UDDI Business Service", then "Original Business Service" is available. • If business services were published (first option on [Service] tab), then "Published Business Services" is available. • If both of the above, then "Both Business Services" is available.
Publish URL to full Layer 7 policy	The policy published to the UDDI registry can be either the client view of the policy (which is sufficient to allow a client to consume the service) or the full policy (as shown when editing the policy in the Policy Manager). The full policy is only available when the policy is downloaded from a white-listed IP address. Tip: Use the <i>service.passthroughdownloads</i> cluster property to configure white-listed addresses.
Inline policy includes	<p>This check box is used to control how to handle included policy fragments in a WS-Policy registered in the UDDI. (These fragments are added using the "Include Policy Fragment Assertion" on page 635.) It is enabled only when [Publish URL to full....] is selected.</p> <ul style="list-style-type: none"> • Select this check box to insert all the assertions from the policy fragment into the main policy. This way you can see all the assertions from the fragment, but the hierarchy is lost. • Clear this check box to show just the "Include Policy Fragment" assertion in the policy. This will retain the policy structure in the fragment but you will not see the individual assertions.

Configuring the [Metrics] tab

The [Metrics] tab is available only for services that have been published to a CentraSite ActiveSOA UDDI registry. Ensure that a Layer 7 Policy Enforcement Point target type has been created in Centrasite ActiveSOA and that a target has been created.

Tip: To publish metrics to UDDI, ensure that the *uddi.centrasite.activesoa.target* cluster property is configured with the name of the target as configured target in CentraSite ActiveSOA.

- Select the **Publish Service Metrics for Published Business Services** check box to enable metrics.

The following metrics will be collected:

- Total Count
- Success Count
- Failure Count
- Minimum Response Time
- Maximum Response Time
- Average Response Time Availability

Controlling Access to the WSDL or WS-Policy

When publishing a service (WSDL) or its WS-Policy to UDDI, a remote requestor may attempt to download the published WSDL/WS-Policy from the Gateway after obtaining the WSDL/WS-Policy URL from UDDI. To control who is permitted to download and the extent of what is downloaded, configure these two cluster properties:

- *service.passthroughdownloads*: This property defines a "whitelist" of who is permitted to download WSDL and policy documents without credentials. By default, only the localhost is permitted. **Tip:** This cluster property allows the use of IP prefixes/masks to configure a wide range of IPv4 or IPv6 addresses for a whitelist. For example, use "10.7.32.0/24" to permit the address range 10.7.32.0 to 10.7.32.254.
- *service.wsdlDependenciesEnabled*: This property defines whether any WSDL dependencies may be downloaded in addition to the WSDL itself. Examples of dependencies include child WSDLs or XML schemas that enable the main WSDL to be fully functional.

For more information on these cluster properties, see "Service Settings" in the Gateway Cluster Properties

Managing Meta Data

When publishing a proxied Gateway endpoint based on the GIF (Governance Interoperability Framework) specifications, it may be necessary to add keyedReference meta data to identify the management server's businessService. The Manage Meta Data dialog allows you to do this when publishing to a UDDI registry.

➤ *To manage meta data:*

1. Access the Publish to UDDI Settings dialog. For more information, see "Publish to UDDI Settings" on page 92.
2. In the [Service] tab, select the '**Publish Gateway endpoint as Binding Template**' option.
3. Select [**Manage meta data**]. The Manage Meta Data dialog appears. This dialog lists the tModelKey, keyName, and keyValue for each keyedReference.

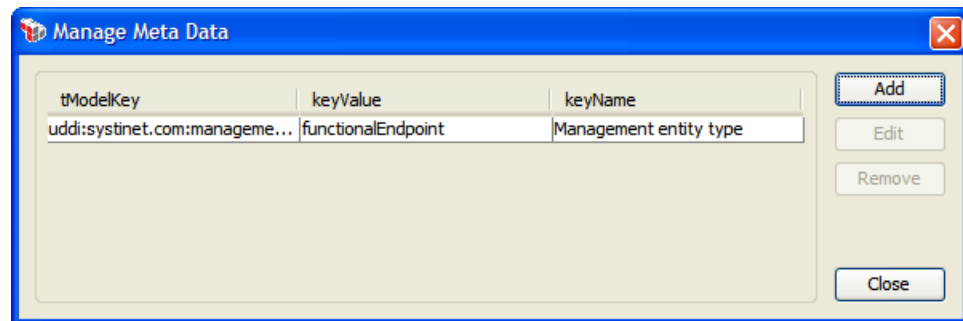


Figure 30: Manage Meta Data dialog

4. Configure the dialog as follows:

Table 21: keyedReference settings

To...	Do this...
Add a new keyedReference	<ol style="list-style-type: none"> 1. Click [Add]. 2. Enter the tModelKey, keyName, and keyValue. 3. Click [OK].
Edit a keyedReference	<ol style="list-style-type: none"> 1. Select the item to edit. 2. Click [Edit] and then modify the values as required. 3. Click [OK].
Remove a keyedReference	<ol style="list-style-type: none"> 1. Select the item to remove. 2. Click [Remove].

5. Click [**OK**] when done.

Chapter 2: Working with Policy Fragments

Policy fragments provide a convenient way to create a group of assertions that can be used in any published service. These "fragments" behave as boilerplate text to help maintain consistency when constructing a policy: once a fragment is created, it can be added to any service policy only as a "read only" entity. This allows you to enforce global rules across any number of services. Maintenance is also simplified: when the active version of a fragment is updated, the changes are instantly applied in every policy where the fragment is used. When a policy is exported or imported, any fragments present are also included.

Tip: Policy fragment can also be used in encapsulated assertions, to create a self-contained package that looks and behaves like a standard assertion. For a detailed explanation on encapsulated assertions, including their similarities and differences with policy fragments, see "Working with Encapsulated Assertions" on page 126.

As with conventional policies, the ability to use a policy fragment depends on the role and permissions of the user currently logged in. At the very least, the user must have Read access to the fragment and Update access to the policy to which the fragment is being added. For more information, see Predefined Roles and Permissions in the *Layer 7 Policy Manager User Manual*.

Example:

The following example illustrates how a service policy can be constructed by users in different roles throughout an organization, using policy fragments that were predefined earlier:

1. The first phase of a policy (e.g., IP address and authentication assertions) is edited by DMZ network operations staff.
2. A subsequent phase containing WS-Security and schema validation assertions could be under the control of a security architect.
3. The routing assertion and other assertions related to Protected Service behaviour assertions are added by the application deployer.

Types of Fragments

There are three types of fragments that you can create:

- **Global policy fragments:** These fragments are predefined by the administrator and will run at specific points during message processing, depending on the global policy tag. These fragments do *not* appear in the service policy. For more information, see "Working with Global Policy Fragments" on page 106.
- **Included policy fragments:** These are fragments that are manually added to a service policy as required, using the "Include Policy Fragment Assertion" on page 635. The fragment appears in the service policy as "*Include Policy Fragment: <name>*" and can be repositioned as necessary.
- **Policy-Backed Identity Provider Policy Fragment:** These fragments contain identity provider policies and are intended for use with Policy-Backed Identity Providers (PBID). (Only policy fragments of this type may be selected for use in a PBID.) For more information, see Policy-Backed Identity Providers in the *Layer 7 Policy Manager User Manual*.

Choose a task from the following table:

Table 22: Policy fragment tasks

For information on how to...	See
Create a new policy fragment (see also "Policy Fragment Shortcut" below)	"Creating a Policy" on page 21
Create a new version of a policy fragment	"Editing a Policy Fragment" on page 104
Add a policy fragment to a policy	"Adding a Policy Fragment to a Service Policy" on page 103
Delete a policy fragment	"Deleting a Policy" on page 105
Edit a policy fragment	"Editing a Policy Fragment" on page 104

Policy Fragment Shortcut

➤ To quickly create a fragment based on existing assertions:

1. Open the [policy revision](#) containing the assertions to be added to a fragment.
2. Select one or more assertions in the policy window.
3. Right-click and select **Create Include Fragment**.
4. Enter a name for the fragment and then click **[OK]**.

The Policy Manager replaces the selected assertions with a new Include Policy Fragment that contains those assertions. The new fragment is added to the Services and Policies list, where it will be available to be added to any other policy.

Policy Fragment Tips

The following tips apply to included policy fragments. For tips and suggestions related to global policy fragments, see "Working with Global Policy Fragments" on page 106.

- Policy fragments are listed in the Services and Policies list.
- Fragments can be as short as a single assertion or as long as a complete policy.
- The assertions in an included policy fragment cannot be edited when the fragment is inserted into a policy. However you can edit an included policy fragment and the changes instantly apply everywhere the fragment is used.
- You can import items from a [policy template](#) into a fragment, but you cannot import from a UDDI registry.
- You can drag and drop one policy fragment into another (i.e., a fragment can be made up of other fragments).
- Policy fragments have their own [revision history](#).
- If the fragment contains assertions that create their own context variables and these variables need to be available to the parent policy (that is, outside of the policy fragment), ensure that a [Export Variables from Fragment](#) assertion appears in the fragment. An example of assertions that create context variables are the XPath-based assertions ([Evaluate Request XPath](#) or [Evaluate Response XPath](#)).
- If the fragment is to be used in an [encapsulated assertion](#), it is not necessary to include the a [Export Variables from Fragment](#) assertion unless XPath-based assertions are involved.

For more information on policies, see "Configuring a Policy" on page 18.

Adding a Policy Fragment to a Service Policy

You can add an [included policy fragment](#) to any service [policy](#) provided that the permissions in your role permit it. Keep the following in mind:

- Adding a fragment adds all the assertions defined within that fragment. It is not possible to remove any assertion in a policy added by a fragment.
- You can view the properties for assertions added by a fragment, but you cannot make changes.

- The policy is parsed as if the assertions in the fragment were manually added to the policy. In other words, the fragment does not interrupt the normal policy logic.

Note: *Global policy fragments* do not need to be manually added to a service policy. These global fragments have predefined rules as to when and where they are run. For more information, see "Working with Global Policy Fragments" on page 106.

➤ *To add an included policy fragment to a service policy:*

1. Open the service policy that will receive the fragment. For more information, see "Editing a Service Policy" on page 22.

Tip: You can choose to open an existing fragment as it is possible to nest a fragment within another fragment. For more information, see "Editing a Policy Fragment" on page 104.

2. Add the [Include Policy Fragment](#) assertion to the appropriate location in service policy.
3. Choose the policy fragment to be added from the list displayed. The fragment appears as: "Include Policy Fragment: <name>" in the policy window. Some tips to note:
 - If the list of fragments is empty, this means no included policy fragments have been created yet. For information on creating a policy fragment, see "Creating a Policy" on page 21.
 - You can activate another version of the fragment before adding it to the policy. For more information, see "Policy Revisions" on page 6.
4. Use the Assertions Tool Bar to reposition the policy fragment if necessary.
5. Repeat steps 2 to 4 to add additional policy fragments if required.

Editing a Policy Fragment

[Policy fragments](#) are edited in the same manner as [service policies](#). Keep the following in mind:

- When you save the active version of a fragment, the changes are effective immediately in all policies using that fragment.
- A new revision is created every time you save changes to a policy fragment. For more information, see "Policy Revisions" on page 6.

➤ *To edit a policy fragment:*

1. Load the fragment into the policy development window using one of the following methods:
 - Double-click the fragment name in the Services and Policies list. This loads the "active" version of the fragment.
 - Right-click the fragment name in the Services and Policies list and then select **Active Policy Assertions**. This loads the "active" version of the fragment.
 - Right-click the fragment name in the policy window and then select **Active Policy Assertions**. This loads the "active" version of the fragment.
 - Right-click the fragment name in the Services and Policies list and then select **Revision History**. This lets you edit any version of the fragment or set any version as the "active" version. For more information, see "Policy Revisions" on page 6.
2. Modify the fragment as necessary. For more information, see "Configuring a Policy" on page 18.
3. Save the changes using either [**Save**] or [**Save and Activate**].

Deleting a Policy

There are several ways to delete a [policy](#) in the Policy Manager:

- For included policies (i.e., policy fragments), you can delete it directly from the service policy. This removes the policy and all its assertions from that one service policy only.
- For all policy types, you can delete it from Policy Manager. This removes it from the Services and Policies list and makes it unavailable for use in any service policy.

Deletions may take up to 15 seconds to take effect. **Tip:** Consider [disabling a policy](#) instead if you think you may need it again in the future.

Note: You cannot delete a *service policy* unless you first delete its associated published service. For more information, see *Deleting a Published Service* in the *Layer 7 Policy Manager User Manual*.

➤ *To delete an included policy from a service policy:*

- Delete the "**Include:** <fragment name>" assertion from the policy. For more information, see "Deleting an Assertion" on page 119. You cannot delete individual assertions within the fragment; you must delete the entire fragment. **Tip:** To

remove individual assertions within a policy fragment, you should [edit](#) the fragment instead.

➤ *To delete a policy from the Policy Manager:*

1. Right-click the policy icon in the Services and Policies list and then select **Delete**.
2. Click **Yes** to confirm.

Notes: (1) You cannot delete a policy that is still in use in any service. (2) You can delete only one policy at a time. If more than one policy icon is selected in the Services and Policies list, only the first will be deleted.

Working with Global Policy Fragments

Global policies are [policy fragments](#) that are always applied before or after every service policy in the system. They can be used to configure global behaviors like auditing or logging, where it may not be feasible to manually add the policy logic to all service policies.

Global policies ensure consistency and reduce possible errors, because an administrator no longer needs to remember to manually insert policy fragments to every service policy to achieve a specific outcome (in that scenario, the Policy Manager is not able to detect instances where the administrator forgets to add a policy fragment).

Only users with the role of 'Administrator' can create or delete global policies. Administrators and those with the role 'Manage Web Service' can edit global policies. Administrators, Operators, and Manage Web Service roles can read global policies.

IMPORTANT: It is important to plan your global policies carefully. Careless use of these policies can cause service requests to fail in ways that may be difficult to diagnose.

Types of Global Policies

The following types of global policies are available (these are selected from the "Policy Tag" field in the [policy properties](#)):

- **message-received:** A policy of this type will run when a message is received, but before the service is resolved.
- **pre-security:** A policy of this type will run before any security is processed in the request. This policy runs even if there is no security in the message.

Other items of note about this policy type: A 'pre-security' global policy can run even before the target service is determined, since it may be necessary to decrypt a message in order to resolve the service. As a result, it is possible that the 'pre-security' policy may run without the corresponding 'post-security' policy being run. For example, consider this scenario: a 'pre-security' policy runs, but then service resolution fails. When this happens, the only policy that can run upon resolution failure is the 'message-completed' policy.

- **pre-service:** A policy of this type will run before the service policy is executed.
- **post-service:** A policy of this type will run after the service policy is executed.
- **post-security:** A policy of this type will run after security is processed for the response. This policy runs even if there is no security processing is required.
- **message-completed:** A policy of this type will run when processing for a message completes. It will run even if the service policy fails, an exception occurs, the service could not be resolved, or any other condition that prevents the service policy from being completed.

Note that only one policy of each type is permitted.

How Global Policies are Evaluated

Global policies are evaluated in the order shown under "[Types of Global Policies](#)" above. However note that not all global policies will be evaluated in all cases. If a service cannot be resolved, then only the following global policies would be run:

- *message-received* policy
- *pre-security* policy (only if security processing is required during service resolution, i.e., for an encrypted message body)
- *message-completed* policy

Global policies of these types should be configured to run without a resolved service and should not assume there is a response message. As a result, a WSDL operation assertion (for example) should not be used in these policies.

If policy processing failed due to an exception or policy falsified error, then the following global policies will be run:

- *message-received* policy
- *pre-security* policy
- *pre-service* policy
- *message-completed* policy

The Gateway will stop evaluating global policies on any error or on policy failure, except for the *message-completed* policy, which always runs.

Tip: A global policy that contains no assertions or all disabled assertions will always succeed.

How a Global Policy Relates to the Service Policy

A global policy shares the following with a service policy:

- request/response messages
- [message context mappings](#)
- audit/fault settings
- authentication details
- built-in context variables (all other variables local to the policy being run)

Routing assertions in a global policy will not affect the routing latency or the URL for the service policy.

The policy status of a global policy affects the overall status for a service: a failure of a global policy causes a "policy falsified" error for message processing.

Details in SOAP faults will not include details for global policies.

If an audit sink policy is configured, it will run after the service policy and all global policies complete.

When constructing a global policy, you may use any [policy fragments](#).

Supported Assertions

The following assertions are tested and certified for use in global policies:

- "Add Audit Detail Assertion" on page 600
- "Add Comment to Policy Assertion" on page 618
- "All Assertions Must Evaluate to True Assertion" on page 619
- "Apply Rate Limit Assertion" on page 573
- "At Least One Assertion Must Evaluate to True Assertion" on page 619
- "Audit Messages in Policy Assertion" on page 602
- "Capture Identity of Requestor Assertion" on page 604
- "Compare Expression Assertion" on page 621
- "Continue Processing Assertion" on page 625
- "Customize Error Response Assertion" on page 430

"Customize SOAP Fault Response Assertion" on page 607
"Include Policy Fragment Assertion" on page 635
"Limit Availability to Time/Days Assertion" on page 584
"Limit Message Size Assertion" on page 668
"Restrict Access to IP Address Range Assertion" on page 592
"Send Email Alert Assertion" on page 612
"Send SNMP Trap Assertion" on page 615
"Set Context Variable Assertion" on page 656
"Stop Processing Assertion" on page 664

W A R N I N G

Other assertions not listed above may work but are not recommended and not supported by CA Technologies. Avoid using assertions in a global policy that have significant side effects, as this can make debugging service policies extremely difficult. Please contact CA Technical Support if you are unsure.

Limitations to Global Policies

Note the following scenarios when global policies will not be processed:

- Global policies will not be processed when the Gateway generates a policy for the SecureSpan XML VPN Client.
- Global policies will be not processed when the Gateway generates the WS-SecurityPolicy document attached to a service WSDL.
- Global policies will not be included if [policy debug tracing](#) is enabled for a service.
- Global policies will not be included when the service policy is [exported](#).
- Global policies will not be included in policy migrations.

Validating Global Policies

As with [policy fragments](#), the [policy validator](#) built into the Policy Manager can provide only limited assistance when editing a global policy.

When global policies are in effect, the validator may display unexpected warnings when editing a service policy since the validator will not see the effects of the global policies. For example, if credentials are collected in a global policy, a service policy that uses those credentials may trigger a validator warning that no credentials have been collected. **Note:** Use of such a global policy is not recommended.


Chapter 3: Working with Policy Assertions

A policy defines restrictions for the consumption of a published service that is protected by the Gateway. Policy assertions are the building blocks for policies in the Policy Manager. Located in the [Assertions] tab, the assertions are organized into categories corresponding to the main requirements of a policy:

- [Access Control](#)
- [Transport Layer Security](#)
- [XML Security](#)
- [Message Validation/Transformation](#)
- [Message Routing](#)
- [Service Availability](#)
- [Logging, Auditing, and Alerts](#)
- [Policy Logic](#)
- [Threat Protection](#)
- [Internal Assertions](#)
- [Custom Assertions](#) (visible only when [Custom Assertions](#) are present)

The [Policy Templates](#) folder is the repository for exported policies. (**Note:** The Policy Templates folder does not appear in the browser client version of Policy Manager.)

Unless specified otherwise, all assertions can be used in a web service and XML application policy. A policy is constructed by either:


- Dragging and dropping an assertion from the [Assertions] tab into the policy development window, or by
- Highlighting the target assertion in the [Assertions] tab and clicking the  (Add Assertion) button in the Assertions Tool Bar.

Most assertions require configuration either before or after being added to the policy development window.

Note: Depending on which Gateway product you have installed, not all assertions described in this help system may be available. See Features by Product in the *Layer 7 Policy Manager User Manual* for a list of which features are available for each product.

Adding an Assertion

You can add an assertion to a service as follows:

1. Ensure the policy window for the service is visible. You can open this window by doing one of the following:
 - Right-click the service name in the Services and Policies list and then select **Active Policy Assertions**, or
 - Double-click the service name in the Services and Policies list
2. Locate the assertion to add using either of these methods:
 - Browse for the assertion by expanding each category under the **[Assertions]** tab.
 - Type a few characters of the assertion's name in the Search box and then select a match to jump directly to the assertion.
3. Add the assertion to the policy by doing one of the following:
 - Drag and drop the assertion from the **[Assertions]** tab into the policy development window.
 - Select the assertion in the **[Assertions]** tab and click the  (Add Assertion) button in the Assertions Tool Bar.

Note: Depending on which Gateway product you have installed, not all the assertions listed in Table 23 may be available. See Features by Product in the *Layer 7 Policy Authoring User Manual* for a list of which features are available for each product.

After adding, some assertions may require additional configuration. Please refer to the documentation for the specific assertion for more details.

Table 23 lists all the predefined assertions their categories under the **[Assertions]** tab.

Tips: (1) The encapsulated assertions feature also allows you to populate any category with custom created assertions based on policy fragments. For more information, see "Working with Encapsulated Assertions" on page 126. (2) When adding an encapsulated assertion to a policy, it is recommended that you manually open the assertion properties to review the required inputs, if the properties dialog does not display automatically.

Table 23: Policy Manager assertions

Assertion	Category
Access Resource Protected by JSAM	Access Control

Assertion	Category
	Custom Assertions
Access Resource Protected by OAM	Access Control Custom Assertions
Add Audit Detail	Logging, Auditing and Alerts
Add Comment to Policy	Policy Logic
Add or Remove WS-Security	XML Security
Add or Remove XML Element(s)	Message Validation/Transformation
Add Security Token	XML Security
Add Timestamp	XML Security
Add WS-Addressing	Message Validation/Transformation
All assertions must evaluate to true	Policy Logic
Apply JSON Transformation	Message Validation/Transformation
Apply Rate Limit	Service Availability
Apply Throughput Quota	Service Availability
Apply XSL Transformation	Message Validation/Transformation
At least one assertion must evaluate to true	Policy Logic
Audit Messages in Policy	Logging, Auditing and Alerts
Authenticate Against Identity Provider	Access Control
Authenticate Against Radius Server	Access Control
Authenticate User or Group	Access Control
Authenticate Using Tivoli Access Manager	Access Control Custom Assertions
Authenticate Against SiteMinder	Access Control
Authenticate with Siteminder R12 Protected Resource	Access Control Custom Assertions
Authorize via SiteMinder	Access Control
Build RST SOAP Request	XML Security
Build RSTR SOAP Response	XML Security
Build SAML Protocol Request	XML Security

Assertion	Category
Build SAML Protocol Response	XML Security
Cancel Security Context	XML Security
Capture Identity of Requestor	Logging, Auditing and Alerts
Check Protected Resource Against SiteMinder	Access Control
Collect WSDM Metrics	Internal Assertions
Compare Expression	Policy Logic
Compress Messages to/from SecureSpan XVC	Message Validation/Transformation
Configure WS-Security Decoration	XML Security
Configure Message Streaming	Message Routing
Continue Processing	Policy Logic
Convert Audit Record to XML	Internal Assertions
Copy Request Message to Response	Message Routing
Create Routing Strategy	Policy Logic
Create SAML Token	XML Security
Create Security Context Token	XML Security
Create XACML Request	XML Security
Customize Error Response	Logging, Auditing and Alerts
Customize SOAP Fault Response	Logging, Auditing and Alerts
Decode MTOM Message	Message Validation/Transformation
Encode/Decode Data	Message Validation/Transformation
Encode to MTOM Format	Message Validation/Transformation
Encrypt Element	XML Security
Enforce WS-I BSP Compliance	Message Validation/Transformation
Enforce WS-I SAML Compliance	Message Validation/Transformation
Enforce WS-Security Policy Compliance	Message Validation/Transformation
Establish Outbound Secure Conversation	XML Security
Evaluate JSON Path Expression	Message Validation/Transformation

Assertion	Category
Evaluate Regular Expression	Message Validation/Transformation
Evaluate Request XPath	Message Validation/Transformation
Evaluate Response XPath	Message Validation/Transformation
Evaluate SAML Protocol Response	XML Security
Evaluate WSDL Operation	Message Validation/Transformation
Evaluate XACML Policy	XML Security
Exchange Credentials using WS-Trust	Access Control
Execute Routing Strategy	Policy Logic
Execute Salesforce Operation	Message Routing Custom Assertions
Export Variables from Fragment	Policy Logic
Extract Attributes for Authenticated User	Access Control
Extract Attributes from Certificate	Access Control
Generate OAuth Signature Base String	XML Security
Generate Security Hash	XML Security
Generate UUID	Policy Logic
Handle UDDI Subscription Notification	Internal Assertions
Include Policy Fragments	Policy Logic
Join Variable	Policy Logic
Limit Availability to Time/Days	Service Availability
Limit Message Size	Threat Protection
Look Up Certificate	XML Security
Look Up Context Variables	Policy Logic
Look Up in Cache	Service Availability
Look Up Item by Index Position	Policy Logic
Look Up by Item by Value	Policy Logic
Look Up Outbound Secure Conversation Session	XML Security
Manage Cookie	Message Routing

Assertion	Category
Manage Gateway	Internal Assertions
Manage Transport Properties/Headers	Message Routing
Map Values	Policy Logic
Manipulate Multivalued Variable	Policy Logic
(Non-SOAP) Check Results from XML Verification	XML Security
(Non-SOAP) Decrypt XML Element	XML Security
(Non-SOAP) Encrypt XML Element	XML Security
(Non-SOAP) Sign XML Element	XML Security
(Non-Soap) Validate SAML Token	XML Security
(Non-SOAP) Verify XML Element	XML Security
Perform JDBC Query	Access Control
Process Routing Strategy Result	Policy Logic
Process RSTR Response	XML Security
Process SAML Attribute Query Request	Message Validation/Transformation
Process SAML Authentication Request	Message Validation/Transformation
Protect Against Code Injection	Threat Protection
Protect Against Cross-Site Request Forgery Assertion	Threat Protection
Protect Against Document Structure Threats	Threat Protection
Protect Against JSON Document Structure Threats	Threat Protection
Protect Against Message Replay	XML Security Threat Protection
Protect Against SQL Attack	Threat Protection
Query LDAP	Access Control
Query Rate Limit	Service Availability
Query Throughput Quota	Service Availability
Replace Tag Content	Message Validation/Transformation
Require Encrypted Element	XML Security
Require Encrypted UsernameToken Profile Credentials	Access Control

Assertion	Category
Require FTP Credentials	Access Control
Require HTTP Basic Credentials	Access Control
Require HTTP Cookie	Access Control
Require NTLM Authentication Credentials	Access Control
Require Remote Domain Identity	Access Control
Require SAML Token Profile	Access Control
Require Signed Element	XML Security
Require SSH Credentials	Access Control
Require SSL or TLS Transport	Transport Layer Security
Require SSL or TLS Transport with Client Authentication	Access Control
Require Timestamp	XML Security
Require Windows Integrated Authentication Credentials	Access Control
Require WS-Addressing	Message Validation/Transformation
Require WS-Secure Conversation	Access Control
Require WS-Security Kerberos Token Profile Credentials	Access Control
Require WS-Security Password Digest Credentials	Access Control
Require WS-Security Signature Credentials	Access Control
Require WS-Security UsernameToken Profile Credentials	Access Control
Require XPath Credentials	Access Control
Resolve Service	Service Availability
REST Manage Gateway	Internal Assertions
Restrict Access to IP Address Range	Service Availability
Retrieve Credentials from Context Variable	Access Control
Retrieve Kerberos Authentication Credentials	Access Control
Retrieve SAML Browser Artifact	Access Control

Assertion	Category
Return Template Response to Requestor	Message Routing
Route via FTP(S)	Message Routing
Route via HTTP(S)	Message Routing
Route via JMS	Message Routing
Route via MQ Native	Message Routing
Route via Raw TCP	Message Routing
Route via SSH2	Message Routing
Run All Assertions Concurrently	Policy Logic
Run Assertions for Each Item	Policy Logic
Scan Using ICAP-Enabled Antivirus	Threat Protection
Scan Using Sophos Antivirus	Threat Protection Custom Assertions
Scan Using Symantec Antivirus	XML Security Custom Assertions
Send Email Alert	Logging, Auditing and Alerts
Send SNMP Trap	Logging, Auditing and Alerts
Set Context Variable	Policy Logic
Set SAML Response Status Code	Message Validation/Transformation
Sign Element	XML Security
Split Variable	Policy Logic
Stop Processing	Policy Logic
Store to Cache	Service Availability
Subscribe to WSDM Resource	Internal Assertions
Translate HTTP Form to MIME	Message Validation/Transformation
Translate MIME to HTTP Form	Message Validation/Transformation
Use WS-Federation Credential	Access Control
Use WS-Security version 1.1	XML Security
Validate Certificate	Message Validation/Transformation
Validate or Change Content Type	Message Validation/Transformation

Assertion	Category
	Threat Protection
Validate HTML Form Data	Message Validation/Transformation
Validate JSON Schema	Message Validation/Transformation Threat Protection
Validate MTOM Message	Message Validation/Transformation
Validate OData Request	Threat Protection
Validate SOAP Attachments	Message Validation/Transformation
Validate XML Schema	Message Validation/Transformation Threat Protection

Deleting an Assertion

You can delete an [assertion](#) that you no longer need in a policy.


Note: Removing an assertion may affect the integrity of a policy. Be sure to check the messages in the Policy Validation Messages window after deleting.

Tip: If you only need remove an assertion from a policy temporarily, consider [disabling](#) the assertion instead.

➤ *To delete an assertion:*

1. In the policy window, right-click the assertion to remove and then select **Delete Assertion**.

Or:

In the policy window, select the assertion to remove, then click .

Tip: You can delete several assertions at the same time by using **[Ctrl]**-click to select them first.

2. Click **[Yes]** to confirm the deletion.

Disabling an Assertion

You can disable specific [assertions](#) in a published policy or policy fragment. This has the same effect as [deleting](#) the assertion, while preserving the assertion's properties and structure of the policy. Disabling an assertion can help you troubleshoot or test a policy. Disabled assertions are ignored by the Gateway during policy consumption and by the

policy validator.

Note: Disabling an assertion may affect the integrity of a policy. Be sure to check the messages in the Policy Validation Messages window after disabling.

Disabled assertions will stand out in the policy development window as follows:

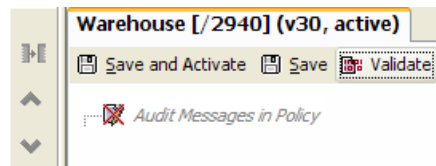


Figure 31: A disabled assertion

- Red "X" over the assertion icon
- Assertion name shown as italicized gray text

A disabled assertion can be [enabled](#) at any time.

Note the following when using the disabling feature with the composite assertions ("[All assertions must evaluate to true](#)", "[At least one assertion must evaluate to true](#)", [Run All Assertions Concurrently](#)):

- Disabling a composite assertion will disable all child assertions contained within it.
- Disabling all child assertions does not disable the parent composite assertion. The end result is the same as a composite assertion with no child assertions.
- If a composite assertion is disabled, enabling any child assertion within it will also enable all parent composite assertions.
- Any assertion added to a disabled composite parent will be disabled, regardless of its original state. Any assertion added to an enabled composite parent retains its previous state.
- New assertions added from the palette to a composite assertion will assume the state of the parent composite assertion.

Tip: To select multiple assertions for disabling, hold down the [Ctrl] key while clicking on the assertion.

➤ *To disable an assertion:*

- In the policy window, right-click the assertion to disable and then select **Disable Assertion**.

Or:

In the policy window, select one or more assertions to disable, then click .

The assertion is removed from the policy logic.

Enabling an Assertion

You can manually re-enable any [disabled](#) assertion individually in a policy. Enabling the assertion restores the assertion properties that were in effect at the time of disabling.

In general, enabling an assertion has the same effect as [adding](#) an assertion to a policy. For the composite assertions ("[All assertions must evaluate to true](#)", "[At least one assertion must evaluate to true](#)", [Run All Assertions Concurrently](#)), note the following:


- If a composite assertion is disabled, enabling a child assertion within it automatically enables all parent composite assertions.
- Enabling a disabled composite assertion does not automatically enable all of its child assertions - they will be restored to the state they were in when the composite assertion was disabled.
- Child assertions that were disabled at the time of disabling the composite assertion will remain disabled. You need to manually enable the appropriate assertions.

Tip: To select multiple assertions for disabling, hold down the **[Ctrl]** key while clicking on the assertion.

➤ *To enable an assertion:*

1. In the policy window, right-click the assertion to enable and then select **Enable Assertion**.


Or:

2. In the policy window, select one or more assertions to enable, then click . You should now [validate](#) the policy.

➤ *To enable all child assertions under a parent assertion:*

- In the policy window, right-click the parent assertion and then select **Enable All Assertions**.

Or:

In the policy window, select one or more child assertions to enable, then click .

Understanding Assertion Latency

The CA API Gateway can calculate the latency for virtually any assertion in a policy. This latency information may be useful in helping to troubleshoot issues, for example connection issues with the back-end service.

The following built-in context variables record the latency:

- **`${assertion.latency.ms}`**: Stores the assertion latency in milliseconds.
- **`${assertion.latency.s}`**: Stores the assertion latency in seconds.

Keep in mind the following important information about assertion latency:

- To reduce the system overhead, the Gateway will calculate the latency only when required. You indicate that the latency is "required" by referencing either the `${assertion.latency.ms}` or `${assertion.latency.s}` variables in the *next* assertion (see the example in Figure 34).
- Calculating assertion latency may affect the outcome of the "At least one assertion must evaluate to true" composite assertion. Reason: The [Set Context Variable](#) or [Add Audit Detail](#) assertion (required to trigger the capture of assertion latency values) will always return "true". This may affect the outcome of the composite assertion (for example, prior to calculating the latency, the composite assertion could have returned "false"; after calculating the latency, the composite assertion will always return "true"). To avoid this, you will need to restructure the policy logic.
- It is not possible to determine assertion latency for child assertions within the [Run All Assertions Concurrently](#) composite assertion. **Note:** If the child assertion is one of the "At least one..." or "All assertions..." composite assertions, then *its* children are eligible for latency calculation. For an illustration of this, see line 12 in Figure 34 below.

How to Use the Assertion Latency Variables

The two latency variables are overwritten each time latency is calculated. To preserve the latency values, you can use either of the following strategies:

- Log the message
- Save the variable for later use

These are described in more detail below, using the `${assertion.latency.ms}` as an example.

Logging the Message

To log the assertion latency, you can use the Add Audit Detail assertion with the `${assertion.latency.ms}` variable defined in the message body—see the following policy sample :

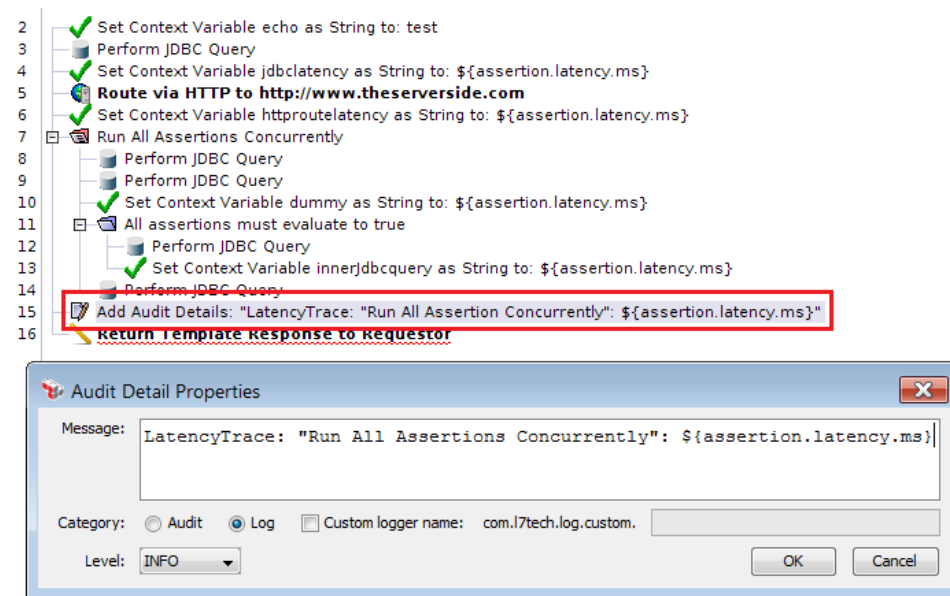


Figure 32: Saving assertion latency by logging the message

In this example, the output log will contain the assertion latency for the composite assertion [Run All Assertions Concurrently](#):

```
2012-04-26T11:37:38.349-0700 INFO 159
com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: -4: LatencyTrace: "Run
All Assertion Concurrently": 16
```

For more information on the output log, see [Viewing Logs](#) in the *Layer 7 Policy Manager User Manual*.

Saving the Variable for Later Use

To save the assertion latency, use the [Set Context Variable](#) assertion to copy the `${assertion.latency.ms}` value to another variable:

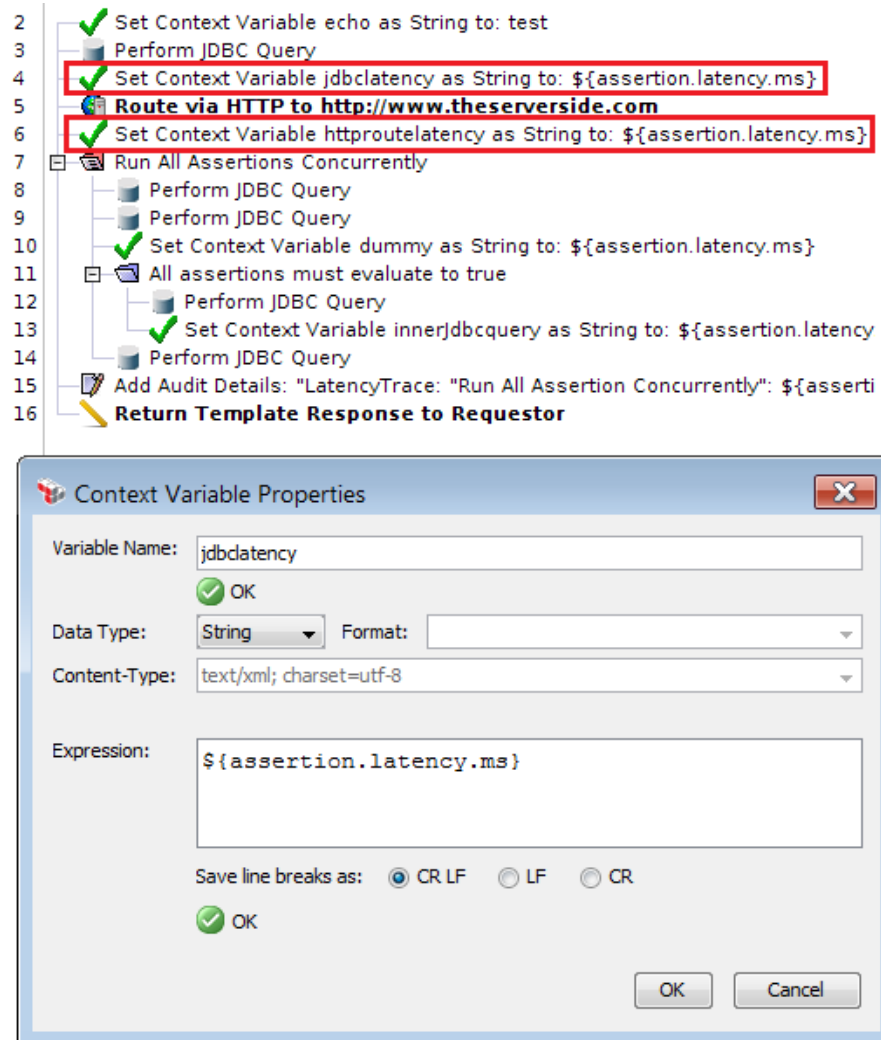


Figure 33: Saving assertion latency by copying the value to another variable

In this example, the `${jdbclatency}` variable contains the latency for the [Perform JDBC Query](#) assertion in line 3, while the `${httproulatency}` variable contains the latency for the [Route via HTTP\(S\)](#) assertion in line 5.

When the Assertion Latency is Calculated

Consider the following example policy:

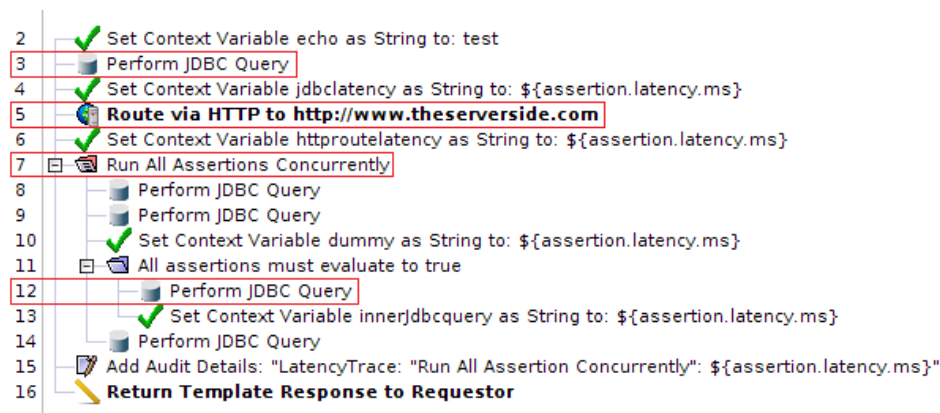


Figure 34: Example of when assertion latency is calculated

In the sample above:

- Latency will be calculated for Perform JDBC Query in line 3, because Set Context Variable is used in line 4 to capture the value from `${assertion.latency.ms}`.
- Latency will be calculated for Route via HTTP in line 5, because Set Context Variable is used in line 6 to capture the value from `${assertion.latency.ms}`.
- Latency will be calculated for the **Run All Assertions Concurrently** composite assertion in line 7, because the **Add Audit Details** assertion is used in line 15 to capture the value from `${assertion.latency.ms}`. (Note that line 15 is used because that is the first line after the composite assertion.)
- Latency will be calculated for Perform JDBC Query in line 12, because Set Context Variable is used in line 13 to capture the value from `${assertion.latency.ms}`. Note that line 12 is eligible for assertion latency capture because it is the child of the "All assertions..." composite assertion in line 11 and is not affected by the disabling of latency capture from the parent Run All Assertions Concurrently assertion in line 7.

It is equally important to understand when latency is *not* calculated:

- Latency is not calculated for lines 2, 4, 6, 8, 11, 13, 15, 16 because neither the `${assertion.latency.ms}` nor `${assertion.latency.s}` variables are referenced in the next assertion.

- In line 9, the latency is not calculated even though `${assertion.latency.ms}` is used in line 10 because the Set Context Variable assertion appears under the Run All Assertions Concurrently assertion (which disables the capture of assertion latency).
- In line 14, the latency is not calculated because this line is the last child in the composite assertion (line 7) and the Add Audit Details assertion in line 15 is considered the next assertion for line 7, not line 14.

Working with Encapsulated Assertions

Encapsulated assertions is a feature within the Policy Manager that lets you turn any [policy fragment](#) into a self-contained "assertion" that accepts input values and sets output values. These encapsulated assertions can be placed in the [assertion palette](#) in any folder that you choose and they can be [added to a service policy](#), [deleted from a policy](#), or [disabled within a policy](#) in the same fashion as the normal assertions. They can also be manipulated using the Assertions Tool Bar.

Tips: (1) Make a note of the encapsulated assertions created, to prevent potential confusion should a policy author need to consult the Policy Manager documentation or contact CA Technical Support for assistance. (2) Encapsulated assertions created outside of the Policy Manager (for example, using the Gateway Management API) will not be visible until the next time a Policy Manager connects to the Gateway. To make them appear immediately, disconnect and then reconnect the Policy Manager to the Gateway.

Encapsulated Assertions vs. Policy Fragments

Though the encapsulated assertions behave similar to policy assertions superficially, they more closely resemble policy fragments from a functional perspective. This is reinforced by the fact that each encapsulated assertion uses a policy fragment as its foundation—if no fragments have been defined, then is it not possible to create an encapsulated assertion.

Encapsulated assertions and policy fragments share the following similarities:

- Both facilitate modularity and policy reuse.
- Both make use of the existing policy assertions.
- Both accept input and produces output, with similar runtime behavior.
- Both use the predefined roles and permissions to control who can create and access these entities.
 - For policy fragments, the "Manage *<fragmentName>*" role controls who has access to the fragment.

- For encapsulated assertions, the "Manage Encapsulated Assertion" role controls who is able to configure an encapsulated assertion. Policy authors who need to use the assertion in a policy can add them, even though they may lack permission to create new ones.

However, there are notable differences between the two, as shown in the following table:

Table 24: Encapsulated assertions vs. Policy fragments

Encapsulated Assertions	Policy Fragments
Authors of encapsulated assertions can see which underlying policy fragment was used, but this is hidden from the policy authors.	Users of the policy fragment (i.e., policy authors) can expand the fragment to see the assertions inside it.
No impact on validation speed in a policy with encapsulated assertions. This is because the underlying policy fragment is not expanded during validation.	Validation speed may be impacted if many policy fragments or large policy fragments are inserted into a policy. This is because the fragment is always expanded during validation.
Each is displayed as its own entry within the assertion palettes, with a configurable name, description, and icon.	Can only be added to a policy using the Include Policy Fragments assertion. No description or icon.
Runs in its own "policy context" so that it can use its own local context variables without conflicting with those in the parent policy. Tip: The request and response themselves are not privately scoped—only the context variables. The request and response for encapsulated assertions point to the actual default request and response.	Runs as part of the parent policy and uses the same context variables as the parent.
Has mechanisms to control the sharing of context variables with the parent policy, via input parameters and output results.	No special mechanism for sharing—policy fragments behave as if you manually inserted the assertions in the policy.
No need to grant permission to the underlying policy fragment in order to use an encapsulated assertion. Anyone with permission to edit a policy may use any encapsulated assertion.	Must be explicitly granted permission to the policy fragment via the "Manage <policy>" role before it can be used.

Visibility of Context Variables

It is important to understanding the visibility of context variables between the parent service policy and the underlying policy fragment ("backing policy") of an encapsulated assertion:

- Context variables set in the parent service policy are not visible to the encapsulated assertion's backing policy.

- Context variables set within an encapsulated assertion's backing policy are local only to that encapsulated assertion. These variables are not visible to the parent policy afterward.
- An encapsulated assertion's input and output arguments are visible to both the parent policy and the backing policy (see "[Understanding How Values are Passed to the Parent Policy](#)" below for details).

Understanding How Values are Passed to the Parent Policy

Every instance of an encapsulated assertion runs in its own "policy context" that is separate from the context of the parent policy. It is important to understand how values are passed between the assertion and its parent. You may assume that all values are passed via context variables.

When you create an encapsulated assertion, you can define a series of *inputs* and *outputs* (see "Encapsulated Assertion Configuration Properties" on page 134). For inputs, you choose a preferred data type and then specify whether that input is shown in the assertion properties or whether it will remain invisible to the interface—the choices you make determines how values are passed to the parent policy.

Example #1: Message input shown in the properties dialog

Suppose you have an encapsulated assertion "ABC Assertion" with the following input definition:

Input Name: foobar
Input Type: Message
Show in assertion properties dialog: true

In the parent policy you have the following:

- [Set Context Variable Assertion](#)

Variable Name: fromParent
Data Type: Message
Content-Type: text/plain
Expression: "Hi there!"

- ABC Assertion

In the assertion properties, choose **fromParent** from the drop-down list for the **foobar** field.

When the ABC Assertion executes, the following context variable is visible to the underlying policy fragment:

- *foobar* of type "Message", which is an alias to the *fromParent* variable in the parent policy's context

If the variable *foobar* is changed within the ABC Assertion, these changes will be reflected in the *fromParent* variable. (Examples of changes include: changes to the content-type using the [Validate or Change Content Type](#) assertion, or setting the variable as the response target message within the [Route via HTTP\(S\)](#) assertion.)

Example #2: String input not shown in the properties dialog

The encapsulated assertion "ABC Assertion" has this input definition:

Input Name: widget

Input Type: String

Show in assertion properties dialog: false

In the parent policy you have the following:

- [Set Context Variable Assertion](#)

Variable Name: widget

Data Type: String

Content-Type: text/plain

Expression: "Pass it along!"

- ABC Assertion

No properties are available for this assertion, as its only input is hidden from the interface. Running the "View Info" option on the ABC Assertion shows that a single variable *widget* is used, with no variables set. The policy XML of this instance of the ABC Assertion within the parent policy will not contain any "widget" parameter.

When the ABC Assertion executes, the following context variable is visible to the underlying policy fragment:

- *widget* of type "String", which is an alias to the *widget* variable in the parent policy's context

If the variable *widget* is changed within the ABC Assertion, these changes will be reflected in the *widget* variable in the parent context, after the ABC Assertion has finished.

Example #3: String input shown in the properties dialog

This example is similar to #2, except it shows the impact of displaying the input in the properties dialog box.

Input Name: widget

Input Type: String

Show in assertion properties dialog: true

In the parent policy you have the following:

- [Set Context Variable Assertion](#)

Variable Name: widget

Data Type: String

Content-Type: text/xml; charset=utf-8

Expression: "Don't pass it along!"

- ABC Assertion

In the assertion properties, enter **"This is my value!"** for the **widget** field.

Entering this value in the properties causes a parameter named "widget" with the value "This is my value!" to be stored in the ABC Assertion instance in the parent policy.

At runtime, each time the ABC Assertion is invoked, the value "This is my value!" will be copied into a new "widget" context variable in the child policy context before the ABC Assertion's underlying policy fragment is executed.

Running the "View Info" option on the ABC Assertion will not show any variables being used or set. However the policy XML of this instance of the ABC Assertion within the parent policy will include a parameter widget="This is my value!".

Advanced Tip: You can view the policy XML by copying the assertion and then pasting it into any text editor.

When the ABC Assertion executes, the *widget* context variable in the parent context will be ignored completely, and a new *widget* context variable will be created in the child context with the value "This is my value!".

Making Encapsulated Assertions Available in a Role

In order for encapsulated assertions to be visible in the assertion palette, a role must have the following permissions:

- READ all Encapsulated Assertions

and *one* of:

- READ all Assertions
or if it is not desirable to grant permission to all assertions for that role:
- READ Assertions with *name=com.l7tech.policy.assertion.EncapsulatedAssertion*

Optionally, if you wish to make the underlying policy fragment visible:

- READ all Policies of type "policy fragment"

Tip: The encapsulated assertions will still operate correctly when there is no Read access to policy fragments; the users will just not be able to view the underlying policy fragment.

The predefined role "Manage Encapsulated Assertions" provides this access. If you are using custom roles, be sure they conform to the above.

For more information, see these topics in the *Layer 7 Policy Manager User Manual*:

Managing Roles
Understanding Role Permissions

Using Encapsulated Assertions

Choose a task from the following table:

Table 25: Encapsulated assertion tasks

For information on how to...	See
Create a new encapsulated assertion	"Managing Encapsulated Assertions" on page 132
Remove an encapsulated assertion from the system	"Managing Encapsulated Assertions" on page 132
Edit an encapsulated assertion	"Encapsulated Assertion Configuration Properties" on page 134
Enable debug tracing in an encapsulated assertion	"Encapsulated Assertion Configuration Properties" on page 134
Add an encapsulated assertion to a policy	"Adding an Assertion" on page 112
Remove an encapsulated assertion from a policy	"Deleting an Assertion" on page 119
Disable an encapsulated assertion in a policy	"Disabling an Assertion" on page 119
Reposition an encapsulated assertion within a policy	Assertions Tool Bar in the <i>Layer 7 Policy Manager User Manual</i>

Managing Encapsulated Assertions

The *Manage Encapsulated Assertion Configurations* task is used to create, modify, and delete [encapsulated assertions](#). These assertions are then available from the assertion palette configured in the [encapsulated assertion properties](#).

To learn more about encapsulated assertions, see "Working with Encapsulated Assertions" on page 126.

Note: One or more [policy fragments](#) must be defined before you can create an encapsulated assertion.

➤ To manage encapsulated assertions:

1. In the Policy Manager, select **[Tasks] > Manage Encapsulated Assertions** from the Main Menu (on the browser client, from the **Manage** menu). The Manage Encapsulated Assertion Configurations dialog appears.

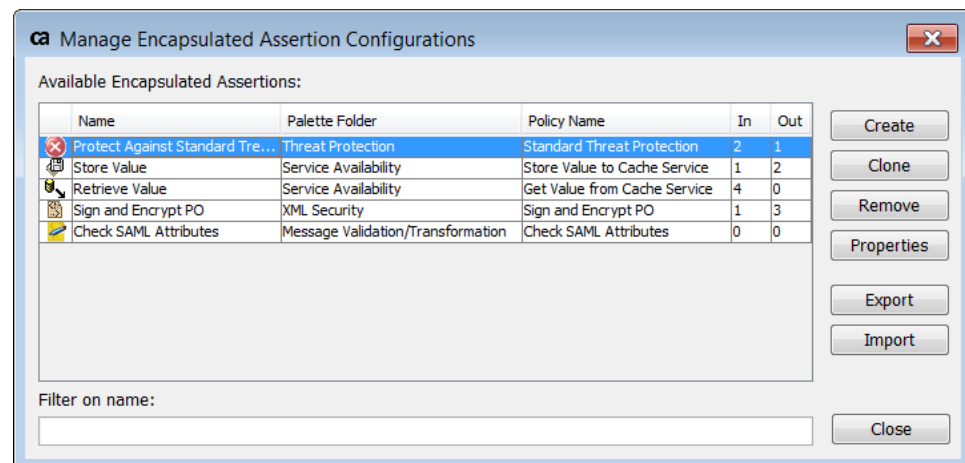


Figure 35: Manage Encapsulated Assertion Configurations dialog (with sample assertions)

Tip: This dialog box is used solely for the creation and maintenance of the encapsulated assertions. How to add, remove, and use them in a policy is the same as the factory created assertions.

2. The following table describes each column (these are set in the encapsulated assertion's [properties](#)):

Table 26: Encapsulated assertion configuration columns

Column	Description
Name	The name of the encapsulated assertion as it appears in the assertions palette.

Column	Description
Palette Folder	The palette in which this assertion appears.
Policy Name	The name of the underlying policy fragment for the encapsulated assertion.
In	How many input variables are defined for the encapsulated assertion.
Out	How many output variables are defined for the encapsulated assertion.

3. Select a task to perform:

Table 27: Manage Encapsulated Assertions tasks

To...	Do this...
Create a new encapsulated assertion	<ol style="list-style-type: none"> Click [Create]. Complete the Encapsulated Assertion Configuration Properties. <p>Tip: You can also create a new encapsulated assertion by right-clicking the policy fragment in the Services and Policies list and then selecting Create Encapsulated Assertion. If the fragment is already associated with an encapsulated assertion, select Encapsulated Assertion Properties to view the settings.</p>
Clone an existing encapsulated assertion	<ol style="list-style-type: none"> Select the assertion to clone. Click [Clone]. A new encapsulated assertion is created, populated with information from the clone source. The name of this assertion defaults to "Copy of <assertion name>". Edit the Encapsulated Assertion Configuration Properties as required.
Remove an encapsulated assertion	<ol style="list-style-type: none"> Select the assertion(s) to remove. Click [Remove]. Select the confirmation check box, and then click [OK]. <p>IMPORTANT: Ensure that the encapsulated assertion is not in use in any policy prior to removal. If it is in use, removing the assertion will make the policy invalid.</p>
View or edit the properties of an encapsulated assertion	<ol style="list-style-type: none"> Select the assertion to view. Click [Properties]. See "Encapsulated Assertion Configuration Properties" on page 134 for details.
Export an encapsulated assertion	<ol style="list-style-type: none"> Select the assertion to export. Click [Export]. Enter a name for the exported file and choose a target directory, then click [Save]. Both the encapsulated assertion and its underlying policy fragment are saved to an XML file.

To...	Do this...
	<p>Note: If a policy containing an encapsulated assertion will be migrated to a different Gateway using the Layer 7 Enterprise Service Manager, ensure the encapsulated assertion is first exported from the source Gateway and then imported into the destination Gateway.</p>
Import an encapsulated assertion	<ol style="list-style-type: none"> 1. Click [Import]. 2. Choose the encapsulated assertion file to import and then click [Open]. 3. If there is a conflict with the GUID for the encapsulated assertion, choose an action: <ul style="list-style-type: none"> • Overwrite: Select this to update the existing encapsulated assertion with attributes from the imported assertion <i>except for the name</i>, which remains unchanged. The underlying policy fragment of the existing assertion will also be overwritten by fragment associated with the imported encapsulated assertion. • Create New: Select this to give a new name for the imported encapsulate assertion and its underlying policy fragment. You will be asked to resolve naming conflicts if you enter a name already in use. • Cancel: Close the dialog box without importing anything. <p>Notes: (1) The encapsulated assertion and its underlying policy fragment are imported. If the policy fragment includes references to other entities (for example, JDBC connections or other policy fragments), these will also be imported. If conflicts occur, the "Resolve External Dependencies Wizard" on page 51 will display. (2) See the note under "Export the Encapsulated Assertion" for information about migrating policies.</p>
Filter list of encapsulated assertions	<p>To locate a specific encapsulated assertion more easily, type a few characters of its name into the Filter on name box. The list is filtered to display only those encapsulated assertions that contain the typed characters anywhere within their names.</p> <p>To reset the display, clear the filter text.</p>

4. Click **[Close]** when done.

Encapsulated Assertion Configuration Properties

When creating, cloning, or viewing details about an encapsulated assertion, the Encapsulated Assertion Configuration Properties appear. These properties allow you to configure the behavior and appearance of the assertion:

- The name, description, and icon that will appear in as assertion palette.
- The palette from which the assertion is available.

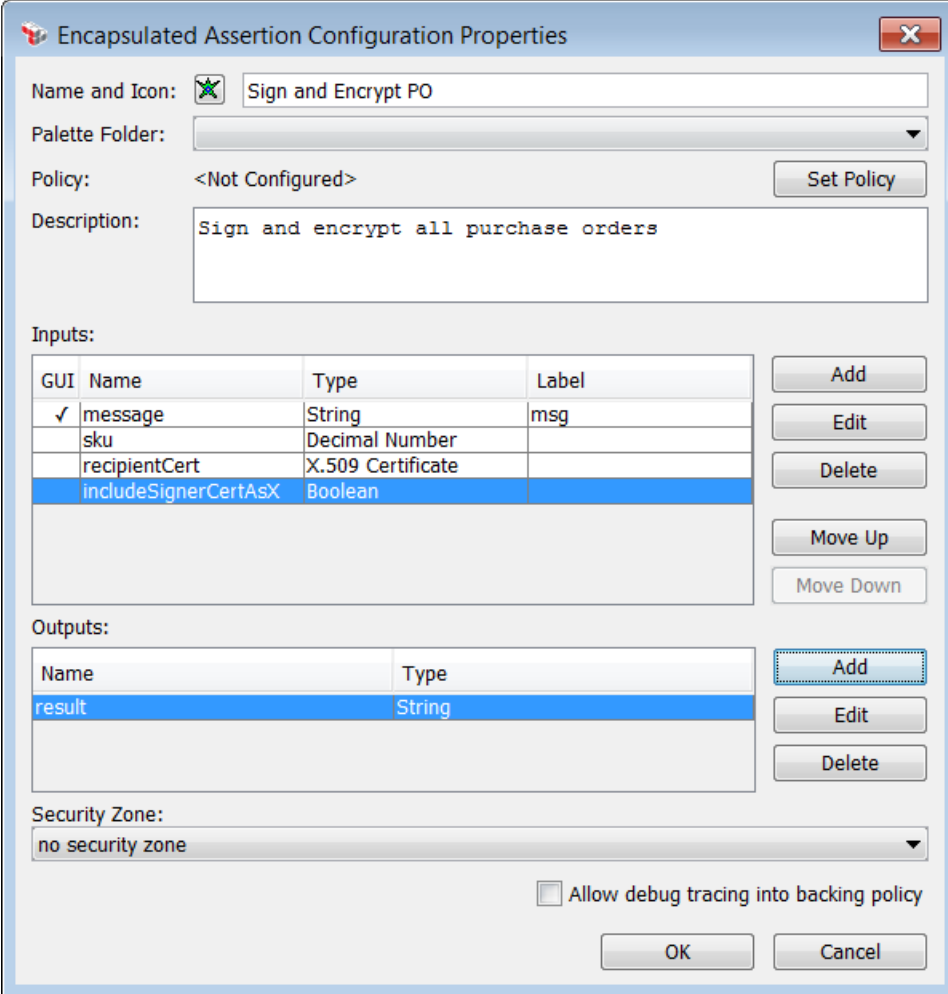
- The underlying policy fragment that forms the foundation of the assertion.
- Configurable inputs and outputs.

For a detailed description of how encapsulated assertions work, see "Working with Encapsulated Assertions" on page 126.

➤ *To access the Encapsulated Assertion Configuration Properties:*

1. Run the [Manage Encapsulated Assertions](#) task. A list of the available encapsulated assertions is displayed.
2. Perform any of the following [actions](#):
 - Create a new encapsulated assertion.
 - Clone an existing encapsulated assertion.
 - View or edit the properties of an encapsulated assertion.

The Encapsulated Assertion Configuration Properties appear.



The dialog box is titled "Encapsulated Assertion Configuration Properties". It contains the following fields and controls:

- Name and Icon:** A text field with a green 'X' icon and the text "Sign and Encrypt PO".
- Palette Folder:** A dropdown menu.
- Policy:** A text field with the value "<Not Configured>" and a "Set Policy" button.
- Description:** A text area with the text "Sign and encrypt all purchase orders".
- Inputs:** A table with columns "GUI", "Name", "Type", and "Label".

GUI	Name	Type	Label
✓	message	String	msg
	sku	Decimal Number	
	recipientCert	X.509 Certificate	
	includeSignerCertAsX	Boolean	

 To the right of the table are buttons: "Add", "Edit", "Delete", "Move Up", and "Move Down".
- Outputs:** A table with columns "Name" and "Type".

Name	Type
result	String

 To the right of the table are buttons: "Add", "Edit", and "Delete".
- Security Zone:** A dropdown menu with the value "no security zone".
- Allow debug tracing into backing policy:** A checkbox.
- OK** and **Cancel** buttons at the bottom right.

Figure 36: Encapsulated Assertion Configuration Properties dialog

3. Choose an **Icon** and then enter a **Name** for the encapsulated assertion. These will appear in the assertion palette and the policy window. **Tip:** If you use your own icon, the recommended size is 16x16 pixels, with a maximum file size of 32KB.
4. Choose the **Palette Folder** where the encapsulated assertion will be located. You can decide which folder best represents your assertion.
5. Click [**Set Policy**] to select the underlying **Policy** for the encapsulated assertion. The underlying policy can be any [Included Policy Fragment](#). Also specify whether to **Auto-populate inputs and outputs**:
 - Select the check box to have the Policy Manager automatically populate the Input and Output sections based on the definition of the chosen policy fragment. The auto population will not update or remove any existing entries

with the same name. You can still make changes to the fields after auto population.

- Clear the check box to populate the inputs and outputs yourself.

Note: Although it is possible to create multiple encapsulated assertions using the same underlying policy fragment, CA recommends against doing this.

6. Enter a **Description** for your encapsulated assertion. This will appear on the Policy Manager interface when the encapsulated assertion is selected. **Tip:** It may be helpful to identify the encapsulated assertion, to prevent possible confusion should policy authors need to consult the Policy Manager documentation or contact CA Technical Support.
7. An **Artifact Version** identifier is displayed for all encapsulated assertions that have been exported or imported using the [Manage Encapsulated Assertions](#) task. This number uniquely identifies the encapsulate assertion plus its associated policy fragment. Identical encapsulated assertions will have the same Artifact Version identifier. Any differences, even to the underlying policy fragment, will trigger a different version number when the encapsulated assertion is exported (no change occurs prior to export). **Tip:** You can use this number to help determine whether an exported encapsulated assertion is the same as one that has already been imported.

The Artifact Version identifier is also visible in the Comment field of the [policy revision](#) created for the underlying policy fragment, to make it possible to roll the policy fragment back to its original state.

Notes: (1) The Artifact Version is not a version number and newer versions may not have an incremented number. It is simply a unique identifier, similar to a generated hash value. (2) The Artifact Version identifier does *not* change if you modify the encapsulated assertion. It will change only if another file (with a different artifact version) is used to import and overwrite the encapsulated assertion.

8. The **Inputs** section lists the context variables and GUI fields that will be used to configure this encapsulated assertion. See "[Configuring Inputs](#)" below for details.
9. The **Outputs** section lists the context variables that will be made available to the parent context after this encapsulated assertion has run. See "[Configuring Outputs](#)" below for details.

10. Optionally choose a security zone. To remove this entity from a security zone (security role permitting), choose "**No security zone**". For more information about security zones, see Understanding Security Zones in the *Layer 7 Policy Manager User Manual*. **Note:** This control is hidden if either: (a) no security zones have been defined, or (b) you do not have Read access to any security zone (regardless of whether you have Read access to entities inside the zones).
11. Select **Allow debug tracing into backing policy** if you want to include the underlying policy fragment during debug tracing; otherwise the backing policy is invisible to the trace (replicates Policy Manager behavior prior to v8.2.0).

To enable debug tracing, you must select the "Enable debug policy tracing" check box in the [General] tab of the published service's properties. For more information, see "Policy Debug Tracing" on page 66.

Note: This setting does not enable or disable debug tracing. It merely controls whether tracing should include the individual assertions within the backing policy when policy tracing is enabled.

12. Click **[OK]** when done.

Configuring Inputs

Note: It is important to have a sound understanding of how the input definition can affect the flow of information between the encapsulated assertion and its parent policy. For more information, see "[Understanding How Values are Passed to the Parent Policy](#)" in "Working with Encapsulated Assertions" on page 126.

The Inputs section is used to define the input arguments for the encapsulated assertion—in other words, the values that will be passed to the underlying policy fragment. The table contains the following columns:

- **GUI:** Whether the input will appear in the encapsulated assertion's properties.
- **Name:** The name of the input.
- **Type:** The data type of the input.
- **Label:** The label that will appear on the interface, if different from the name.

These column values are described in more detail in Table 28 below.

Tip: The Policy Manager will pre-configure inputs for you if the **Auto-populate inputs and outputs** check box was selected. You can change any auto-populated input as necessary.

Choose an action to perform:

Table 28: Encapsulated assertions: Argument Properties

To...	Do this...
Add an input	<ol style="list-style-type: none"> 1. Click [Add]. 2. Complete the Argument Properties (see "Completing the Argument Properties" below for details).
Edit an input	<ol style="list-style-type: none"> 1. Select the input to change. 2. Click [Edit]. 3. Modify the Argument Properties as required (see "Completing the Argument Properties" below for details).
Delete an input	<ol style="list-style-type: none"> 1. Select the input to change. 2. Click [Delete]. The input is deleted immediately.
Reposition an input in the assertion properties	<ol style="list-style-type: none"> 1. Select the input to reposition. 2. Click [Move Up] or [Move Down]. <p>Tip: Repositioning an input only applies to inputs that are shown in the assertion properties dialog. It has no effect on functionality and does not apply to inputs suppressed from the dialog.</p>

IMPORTANT: Be extremely careful when changing the inputs of an encapsulated assertion that is currently in use by policies. In particular, pay careful attention when adding new inputs or renaming existing inputs: ensure that the underlying policy fragment will respond gracefully if the input is not provided.

Completing the Argument Properties

When adding or editing an input, the Argument Properties dialog is displayed:

Complete the properties as follows:

Table 29: Encapsulated assertions: Argument Properties

Setting	Properties
Name	<p>Enter a name for the input. This name should generally match the name of a context variable from the parent context and should be meaningful to the underlying policy fragment.</p> <p>The name must conform to the "Context Variable Naming Rules", described under Context Variables in the <i>Layer 7 Policy Manager User Manual</i>.</p>
Type	<p>From the drop-down list, choose a data type for the input. This sets the GUI control that is visible if the input is set to show in the assertion properties dialog.</p> <p>Note: The data types "Message" or "Element" will <i>always</i> result in the child policy context containing a reference to the value from the parent context, while the other data types will vary depending on whether input is shown on the assertion properties. For more information, see "Working with Encapsulated Assertions" on page 126.</p>
Show in assertion properties dialog	<p>Select this check box to display the input in the assertion properties. When visible, all inputs of type "Message" and "Element" are aliased in the child policy context. All other data types are copied into the child policy context.</p> <p>Clear this check box to hide the input from the assertion properties. When hidden, all values are aliased in the child policy context, and will appear in the Assertion Information dialog as variables used by the encapsulated assertion.</p> <p>Tip: For more information, see "Understanding How Values are Passed to the Parent Policy" under "Working with Encapsulated Assertions" on page 126. Examples #1 and #2 in that section illustrate "aliasing", while Example #3 demonstrates "copying".</p>
Label	<p>Optionally enter a label that will appear in the assertion properties. If not specified, the Name is used as the label.</p> <p>A label allows you to display a more descriptive or "friendly" name in the assertion properties. Unlike the Name, the Label may contain any string of characters.</p>

Configuring Outputs

The Outputs section is used to define the context variables that will be set by the encapsulated assertion. Only the context variables declared here will be visible to the parent context once the encapsulated assertion has finished running.

Tip: The Policy Manager will pre-configure outputs for you if the **Auto-populate inputs and outputs** check box was selected.

Choose an action to perform:

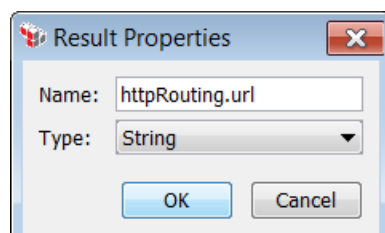
Table 30: Encapsulated assertions: Result Properties

To...	Do this...
Add an output	<ol style="list-style-type: none"> 1. Click [Add]. 2. Complete the Result Properties (see "Completing the Result Properties" below for details).
Edit an output	<ol style="list-style-type: none"> 1. Select the output to change. 2. Click [Edit]. 3. Modify the Result Properties as required (see "Completing the Result Properties" below for details).
Delete an output	<ol style="list-style-type: none"> 1. Select the output to change. 2. Click [Delete]. The output is deleted immediately.

IMPORTANT: Proceed with caution when changing the output of an encapsulated assertion currently in use. Ensure that any new output does not conflict or overwrite any context variables already in use by existing user policies. When modifying or removing an output, consider the behavior of any existing user policies that rely on that output.

Completing the Result Properties

When adding or editing an output, the Result Properties dialog is displayed:



Complete the fields:

- Enter the **Name** of the context variable that will be set by the underlying policy fragment and made available to the parent policy context.
- Choose the data **Type** of the result.

Tip: The output Type is currently useful for your own documentation purposes, but it is not enforced at runtime. The type selected here will be displayed in the Assertion Information dialog for the encapsulated assertion.

Policy Templates

Note: Policy templates have limited support in the browser client version of the Policy Manager. There is no Policy Templates category in the [Assertion] tab, but you can still [import](#) and [export](#) template files.

The Policy Manager allows you to rename and delete an [exported](#) policy that appears in the Policy Templates category of the [Assertions] tab. Although you cannot directly edit the XML content of an exported policy, you can modify an exported template by importing it, updating the assertions, then exporting it back to the same template name.

The following table summarizes the tasks for policy templates.

Table 31: Editing policy templates

Task	Description
Rename a policy template	<ol style="list-style-type: none"> 1. Right-click the policy name under Policy Templates and then select Rename. 2. Enter a new template name. 3. Click [OK]. The template is renamed.
Delete a policy template	<ol style="list-style-type: none"> 1. Right-click the policy name under Policy Templates and then select Delete. 2. Click [Yes] to confirm. The template is removed.
Edit the assertions in a policy template	<ol style="list-style-type: none"> 1. Import the policy to be edited. (Hint: Save your current policy first by exporting it.) 2. Modify the assertions as required. 3. Export the edited policy back to the same file name, in the default directory.

Managing Kerberos Configuration

The Manage Kerberos Configuration task displays information about your Windows Domain Login configuration (Kerberos). Use it to install a Kerberos keytab file and to verify your Kerberos configuration.

Prerequisite: Ensure that Kerberos and the Active Directory is configured and operational.

Note: For information on creating keytab files, refer to "Using Windows Domain Login" in the *Layer 7 Installation and Maintenance Manual*.

➤ To manage Kerberos configuration:

1. In the Policy Manager, select **[Tasks] > Manage Kerberos Configuration** from the Main Menu (on the browser client, from the **Manage** menu). The Kerberos Configuration dialog appears.

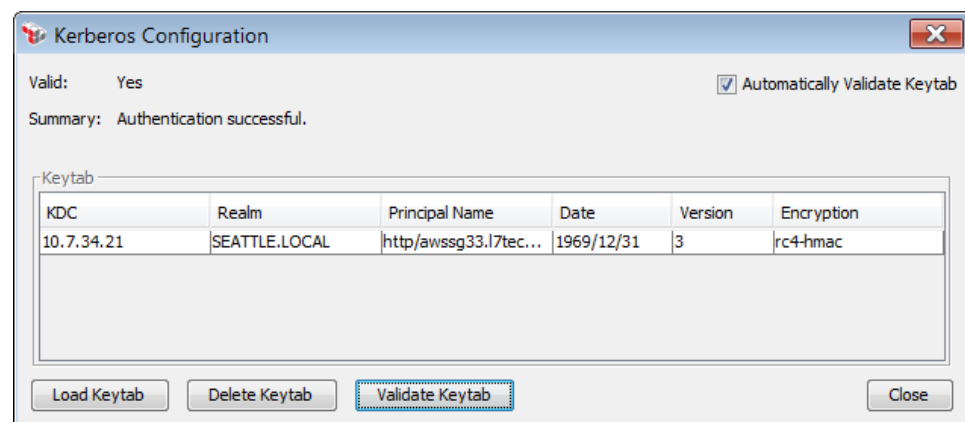


Figure 37: Kerberos Configuration dialog

2. The following table describes each setting and control in the configuration dialog.

Table 32: Kerberos Configuration settings

Field	Description
Valid	Displays the status of the keytab: <ul style="list-style-type: none"> • Yes = valid keytab file has been loaded • No = no valid keytab file has been loaded • "—" = a keytab file has been loaded, but not validated
Summary	Summarizes the state of your Kerberos configuration. Message is one of: <p><i>Keytab file not present</i></p> <p><i>Keytab file is invalid</i></p>

Field	Description
	<i>Authentication failed</i> <i>Authentication successful</i> <i>Checking configuration...</i> <i>Updating configuration...</i>
Automatically Validate Keytab	<p>Select this check box to validate the keytab principal against the corresponding KDC. This validation occurs automatically whenever:</p> <ul style="list-style-type: none"> the Kerberos Configuration dialog is displayed a new keytab is loaded <p>Clear this check box to not automatically validate the keytab. In this case, no validation status or summary is displayed until you click [Validate Keytab].</p>
<i>Keytab details:</i>	
KDC	Key Distribution Center
Realm	Identifier for the secured network
Principal Name	Service (gateway cluster) identifier
Date	Keytab date, if available
Version	Keytab version number 1-X
Encryption	Keytab algorithms (rc4-hmac, des-cbc-md5, etc.)
<i>Keytab configuration controls:</i>	
[Load Keytab]	<p>Loads a keytab file directly into the Gateway database. Select the keytab file to upload, then click [OK] to confirm.</p> <p>If automatic validation is enabled, this keytab will be validated upon loading, otherwise you should use [Validate Keytab] to trigger a validation.</p> <p>For information on how to create the keytab file, see <i>Using Windows Domain Login</i> in the <i>Layer 7 Installation and Maintenance Manual</i>. If you are working with multiple principals, ensure that you select a keytab that has been configured with multiple principals.</p> <p>Tip: Ensure that you have a backup of the keytab file, as it cannot be downloaded once uploaded.</p> <p>Note: Loading a keytab file here will overwrite any existing keytab file.</p>
[Delete Keytab]	<p>Removes the loaded keytab file. As deleting a keytab file is permanent and may have consequences, you must confirm by first selecting the To enable [OK] ... check box before you can click [OK].</p> <p>Tip: If you are simply replacing the keytab file with another one, you can use [Load Keytab] without needing to delete the old keytab first.</p>

Field	Description
[Validate Keytab]	<p>Validates the keytab against the corresponding KDC. The results are displayed in the Summary above. If the keytab is invalid, a message is displayed.</p> <p>Tip: Clicking [Validate Keytab] is not necessary if the Automatically Validate Keytab check box is selected.</p>

3. Click [**Close**] when done.

Authenticating a Client via Kerberos

There are two ways to authenticate a client via Kerberos:

- **Authenticate via the XML VPN Client:** If the SecureSpan XML VPN Client is used in the workflow, then Kerberos authentication can be configured there. For information on how to do this, see "Authenticating a Client via Kerberos" in the *SecureSpan XML VPN Client User Manual*.

Note: If the SecureSpan XML VPN Client is in use, it must be connected to a Gateway policy that contains the "Require WS-Security Kerberos Token Profile Credentials Assertion" on page 243.

- **Authenticate via the Gateway:** In the absence of the SecureSpan XML VPN Client, Kerberos authentication can be configured on the CA API Gateway. This is described below.

➤ *To authenticate a client via Kerberos using the Gateway:*

1. Ensure that the client is logged into the domain trusted by the Key Distribution Center (KDC). The client must be able to acquire the Kerberos ticket from the KDC that issued the keytab.

The KDC is also known as the "Active Directory" and is listed on the Kerberos Configuration dialog (see next step).

2. Run the **Manage Kerberos Configuration** task and import the keytab into the Kerberos Configuration dialog. For more information, see "Managing Kerberos Configuration" on page 143
3. Ensure that the service policy contains both these assertions:

Require Windows Integrated Authentication Credentials
Route via HTTP(S)

4. Access the HTTP(S) Routing Properties and select the [Security] tab.
5. Under **Service Authentication**, choose the **Use Windows Integrated** option and then **choose Use Delegated Credentials** (see Figure 38).

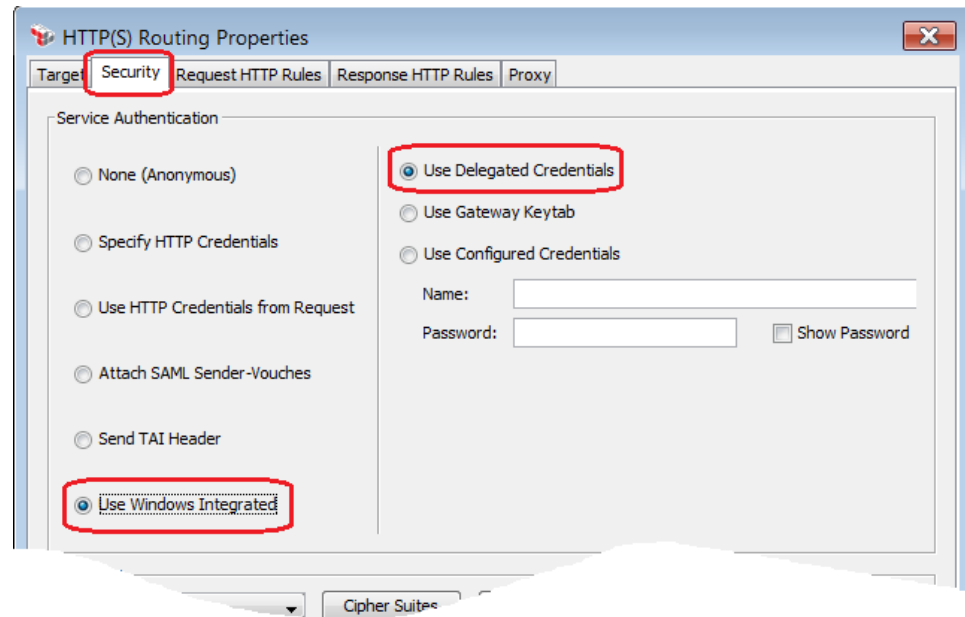


Figure 38: Configuring the HTTP(S) Routing Properties for Kerberos authentication

Once a client is authenticated via Kerberos, Authorization Data attributes from the Kerberos ticket are exposed via context variables. For a list of the available attributes, see "Kerberos Ticket Authorization Info Variables" under Context Variables in the *Layer 7 Policy Manager User Manual*.

Changing the WSS Assertion Recipient

You can change the default WSS assertion recipient in the Policy Manager. The effect of the change will differ depending on the type of assertion:

- **Request security assertions**

These assertions control the requests entering the Gateway. The following are the request security assertions:

- [Add Security Token](#) (with target [set](#) to "Request")
- [Encrypt Element](#) (with [target](#) set to "Request")
- [Protect Against Message Replay](#)
- [Require SAML Token Profile](#)
- [Require WS-Addressing](#)

[Require WS-Secure Conversation](#)
[Require WS-Security Kerberos Token Profile Credentials](#)
[Require WS-Security Password Digest Credentials](#)
[Require WS-Security Signature Credentials](#)
[Require WS-Security UsernameToken Profile Credentials](#)
[Sign Element \(with target set to "Request"\)](#)

The client (typically the Securespan XML VPN Client) creates the security header in the request before it is sent to the Gateway.

How this affects the Gateway

On the Gateway, configuring the WSS recipient for these assertions has little effect: the Gateway will ignore any security decorations in a security header that is not addressed to the Gateway itself. If the assertion specifies a foreign WSS recipient, then the assertion will immediately succeed on the Gateway with no further checks on the request.

Note: A security header is considered addressed to the Gateway when it contains one of the following Actor attributes:

`secure_span`

`http://www.layer7tech.com/ws/policy`

`http://schemas.xmlsoap.org/soap/actor/next/`

an empty or unspecified Actor

How this affects the Securespan XML VPN Client

On the Securespan XML VPN Client, configuring the WSS recipient for these assertions will configure which security header and recipient certificate to use for the security decorations. Using these assertions will often cause the Securespan XML VPN Client to include security decorations not intended for the Gateway. When this happens, the Gateway will process the decorations that are intended for it and ignore the others. The Gateway can be configured to promote the foreign security header to the default security header when the request is routed to the back-end system.

The Securespan XML VPN Client includes no *Actor* attribute on the *Security* header when it is intended for the default WSS Recipient for a generic (i.e., not a Gateway) web service.

- **Response security assertions**

These assertions control the responses returned by the Gateway to the client. The following are the response security assertions:

The security header in the response is created by the Gateway after the policy is finished.

How this affects the Gateway

On the Gateway, configuring the WSS recipient for these assertions causes the response security header to have a specific Actor attribute value; it also causes the Gateway to use the specified certificate for any message decorations that require a recipient certificate.

Example: The [Encrypt Element](#) assertion will encrypt the element for the specified certificate's public key instead of using the public key in the client certificate from the request (if any).

How this affects the Securespan XML VPN Client

On the Securespan XML VPN Client, configuring the WSS recipient for these assertions will have little effect. The Securespan XML VPN Client will ignore any response security decorations within a security header not addressed to it (in other words, with an Actor attribute of "secure_span", or "http://schemas.xmlsoap.org/soap/actor/next/", or an empty or unspecified Actor). If the assertion specified a foreign WSS recipient, then the assertion will immediately succeed on the Securespan XML VPN Client with no further checks on the response.

- **Routing assertions that support SAML sender-vouches attachment**

These assertions control the requests from the Gateway to the back-end system. The following routing assertions support SAML sender-vouches attachment:

[Route via HTTP\(S\)](#)

[Route via JMS](#)

The security header is created by the Gateway while the routing assertion is executing.

How this affects the Gateway

On the Gateway, configuring the WSS recipient for these assertions has no effect unless "Attach SAML sender vouches" is selected for the back-end authentication. When this is the case, the Gateway will use the specified Actor attribute value for the security header that contains the SAML token.

How this affects the Securespan XML VPN Client

On the Securespan XML VPN Client, this is not applicable since the Securespan XML VPN Client never receives the routing assertions (they are filtered out from the policy it downloads).

When a WSS assertion is added to a policy, the associated WSS decoration will be written in the Security header, intended by default for the "next-in-line" recipient (the Gateway) in the SOAP message. Changing the recipient of a WSS assertion configures a different downstream recipient, with a unique Actor attribute value, for the WSS decoration, essentially by-passing the Gateway. A policy can contain multiple alternate recipients, each resulting in a separate Security SOAP header with its associated unique Actor attribute.

W A R N I N G

If the intended recipient does not accept or recognize Security headers that contain Actor attributes, then you must configure the [Route via HTTP\(S\)](#) assertion in the policy to instruct the Gateway to promote one of the downstream WSS recipients as the next default WSS header. To do so, select the "Promote other Security header as default before routing" option in the WSS Header Handling section of the HTTP(S) Routing Properties dialog.

The procedure below provides general instruction on how to change the WSS assertion recipient. For more information, refer to the configuration instructions for each assertion. The Gateway supports both versions 1.0 and 1.1 of the WS-Security standard.

➤ *To change the WSS recipient for an individual WSS assertion:*

1. Right-click a WSS assertion in the policy development window and then select **WSS Recipient**. The assertion properties are displayed.

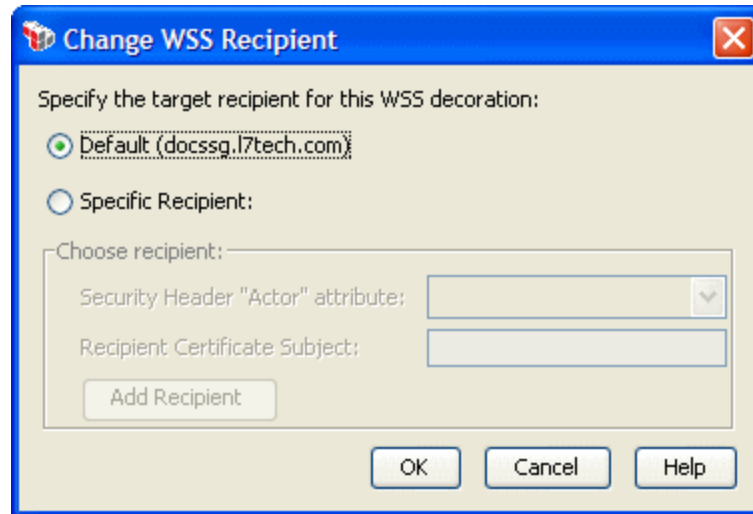


Figure 39: Change WSS Recipient dialog

2. The current Gateway is selected as the default target recipient for the WSS decoration. To change the recipient, select the **Specific Recipient** option.
3. At this point, you can either choose an existing recipient to change to, or you can change to a new recipient.

Table 33: WSS Assertion Recipient tasks

Task	Description
Choose an existing recipient	<p>If the intended downstream recipient was already defined for another WSS assertion in the policy, then:</p> <ol style="list-style-type: none"> 1. Select the corresponding recipient's Actor attribute value from the Security Header "Actor" Attribute drop-down list. The certificate subject information for the previously configured target appears in the Recipient Certificate Subject field. 2. Click [OK]. The recipient's Actor attribute value appears as an extension of the WSS assertion's name in the policy development window.
Adding a new recipient	<p>Note: To create a new downstream recipient, you will need access to the recipient's certificate.</p> <p>To configure a new recipient for use in the WSS assertions:</p> <ol style="list-style-type: none"> 1. Click [Add Recipient]. 2. Complete the Add WSS Recipient Wizard. When the wizard is complete, the new recipient's information appears in the Security Header "Actor" Attribute and Recipient Certificate Subject fields. 3. Click [OK]. The recipient's Actor attribute value appears as an extension of the WSS assertion's name in the policy

Task	Description
	<p>development window.</p> <p>To view or edit the recipient for a WSS assertion, right-click the assertion and select WSS Recipient from the drop-down menu.</p>

- Click **[OK]**.

New WSS Recipient Wizard

The *New WSS Recipient Wizard* helps you create a new WSS recipient. This wizard starts when you click **[Add Recipient]** on the Change WSS Recipient dialog. For more information, see "Changing the WSS Assertion Recipient" on page 146.

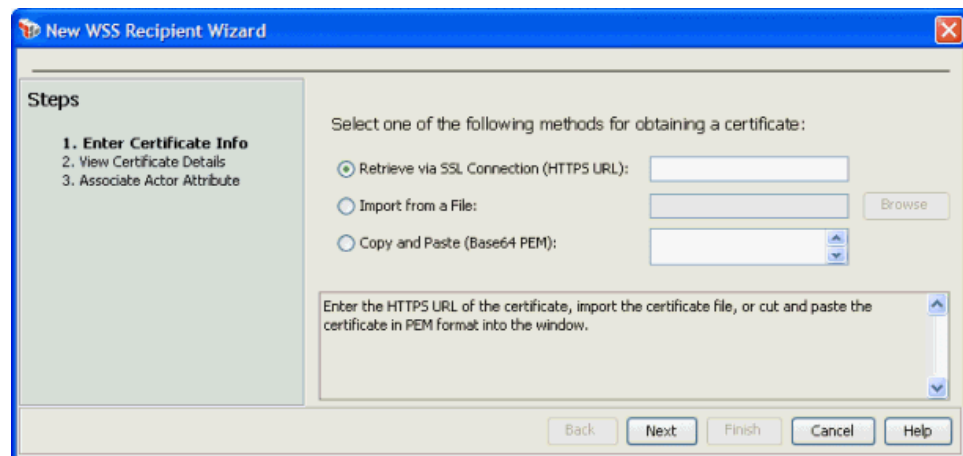


Figure 40: New WSS Recipient Wizard

For more information about wizards, see "Wizard" under Interfaces in the *Layer 7 Policy Manager User Manual*.

Table 34: Using the New WSS Recipient Wizard

Wizard Step	Description
Step 1: Enter Certificate Info	<p>This step lets you specify the source of the new certificate. Specify how to obtain the certificate:</p> <ul style="list-style-type: none"> • Retrieve via SSL Connection: Select this option to get the certificate from an HTTPS URL. • Import from a File: Select this option to get the certificate from a local file. Either enter the file path in the field, or use [Browse] to locate the file. • Copy and Paste: Select this option to copy and paste the entire certificate from the originating file into the code window. <p>Note: You can only cut and paste a certificate that is in Base 64 PEM</p>

Wizard Step	Description
	<p>format. In the Code window, the Policy Manager will only add a pasted certificate that begins with "BEGIN TRUSTED CERTIFICATE."</p> <p>If you encounter an error moving to the next step of the wizard, verify that the certificate information entered is correct and then try again. If you require assistance, contact CA Technical Support.</p>
Step 2: View Certificate Details	<p>This step appears if the Policy Manager was able to obtain the certificate successfully.</p> <ul style="list-style-type: none"> • Certificate Name: Optionally enter a descriptive name for the certificate. • Details: Examine the certificate details.
Step 3: Associate Actor Attribute	<p>Enter a unique Actor attribute for the recipient certificate into the Actor Attribute Value field. Uniqueness is necessary because a recipient is identified in the Change WSS Recipient dialog by its associated Actor attribute.</p> <p>In the Policy Manager, the certificate and Actor attribute are a locked combination that can be used multiple times by multiple WSS assertions.</p>

Selecting a Target Identity

When [multiple signatures](#) are in use, you must specify which identity is the signing identity for each assertion that requires a signature.

Note: If multiple signatures are present in a message, you must specify the signing identity. Otherwise, the assertion will fail even if the element is signed.

➤ *To select a target identity:*

1. In the policy window, right-click on an assertion that deals with signatures and then choose **Select Target Identity**. Only assertions that support target identities will display this option. The Select Identity dialog appears.
2. Select the target identity from the drop-down list. Note that you can either select a previously authorized user (i.e., "Bob [Internal Identity Provider]") or an identity tag (i.e., "tag1") that was defined earlier.
3. Click **[OK]**. The selected identity is displayed in the policy as follows:

[User: <Login>, <Provider Name>] (example: "[User: Alice, Internal Identity Provider]")

[Group Membership: <Group Name>, <Provider Name>] (example: "[Group Membership: A Group, Internal Identity Provider]")

[Authenticated against: <Provider Name>] (example: "[Authenticated against: Internal Identity Provider]")

[Identity Tag: <Tag>] (example: "[Identity Tag: A-User_1, internal]")

Selecting a Target Message

Many assertions can apply to a specific target message: request, response, or a context variable. The default target depends on whether the assertion appears before or after a routing assertion.

Tip: Be sure the assertion is located correctly in a policy after selecting a target message (see the policy validator for warnings). For example, specifying "Response" as the target message when the assertion appears before the routing assertion will not return correct results.

➤ To select a target message:

1. [Add](#) the assertion to the policy development window.
2. Right-click on the assertion and then choose "**Select Message Target**" from the context menu. The Message Target dialog is displayed.



Figure 41: Selecting a message target

Tip: If the Message Target dialog is "read only" (i.e., not editable), it may be caused by the policy being imported into a Gateway where the licensing does not include the specified assertion. For a list of assertions licensed in each version of the Gateway, see Features by Product in the *Layer 7 Policy Manager User Manual*.

3. Specify the target for the message:
 - **Request:** The target is the request message. This includes both the *inbound request* (message from the client to the Gateway) and *outbound request* (message from Gateway to the web service).

- **Response:** The target is the response message. This includes both the *inbound response* (message from the web service to the Gateway) and *outbound response* (message from Gateway to the client).
 - **Other Context Variable:** The target is the specified context variable. This variable must be of type Message and must be predefined or has been set in the policy prior to the assertion. For more information on Message variables, see "Context Variable Data Types" under Context Variables in the *Layer 7 Policy Manager User Manual*.
 - Enter the context variable in the field in the format: `${variableName}`
4. Click **[OK]**. The message target is indicated by a "Request:", "Response:" or "\${variableName}" prefix in the assertion name in the policy window.

Examples:

Request: Authenticate against XYZ

Response: Add signed Timestamp

Selecting an XPath

Some assertions require you to specify an XPath on which to perform an operation. For example, you may be [evaluating a request XPath](#), [encrypting an element](#), or [signing a non-SOAP element](#). Figure 42 shows an example of the interface used to select or modify an XPath.

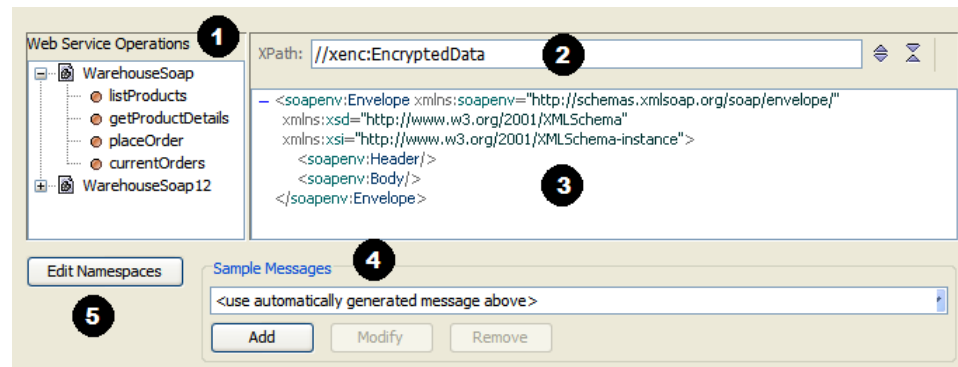





Figure 42: User interface for selecting an XPath

This interface contains the following elements. Every assertion with an XPath selection task contains these elements; some assertion contain additional elements specific to that assertion.

- 1 List of available web services and their operations. The operations shown are retrieved from the WSDL document for the web service. For information about the WSDL that defines a web service, see Working with SOAP Web Services in the *Layer 7 Policy Manager User Manual*.

- 2 The XPath expression selected for the task. If you know the XPath, you can type it directly into this box. For greater flexibility, you may reference context variables within the expression or you may specify a fully-dynamic XPath expression (in the format "{\$xpathVar}"). For more information, see "Fully Dynamic XPath Expressions" in Context Variables for XPaths in the *Layer 7 Policy Manager User Manual*. Alternatively, you can let the Policy Manager build it for you by selecting the target element in the code window. 


Note: While context variables within the expression are supported by all XPath assertions, fully-dynamic XPath expressions are only available in the [Evaluate Request XPath](#) and [Evaluate Response XPath](#) assertions.

- 3 The code box that displays a sample message for the selected operation. Use the  and  buttons to collapse and expand the nodes (respectively).
- 4 By default, the sample message shown in the code window will be used. You can modify the sample message or enter your own message. For more information, see Sample Messages in the *Layer 7 Policy Manager User Manual*.
- 5 You can optionally edit the namespace map if necessary. You cannot remove namespaces that originate from the WSDL document. For more information, see "Namespace Map" on page 156.

➤ *To select an XPath:*

1. In section 1, select the operation containing the element you want to use for the task (evaluating, encrypting, etc.) A sample message is displayed in the code window 3.
2. Examine the sample message generated by the Policy Manager in 3 to see if it meets your needs. If not, use the **Sample Messages** section in 4 to create your own message. Messages generated by the system are limited to elements defined in the WSDL document.
3. In the code box 3, click a target element to build your XPath expression. The XPath for this element is displayed in the **XPath** field 2.

Note: The Policy Manager will only build an XPath expression to an element. If you need a more complex expression, you must edit the XPath manually in 2.

4. Edit the **XPath** in 2 if necessary. For greater flexibility, you may reference context variables within the expression.  For more information, see Context Variables for XPaths in the *Layer 7 Policy Manager User Manual*.
5. Edit the namespace map in 5 if necessary.

Namespace Map

The Policy Manager cannot predict which namespace prefixes, if any, might be found in the messages that will be received by the Gateway. To allow the use of namespace prefixes in XPath expressions, the Gateway supports a user-editable namespace map attached to any XPath-based assertion (the [Evaluate Request XPath](#), [Evaluate Response XPath](#), and [Require XPath Credentials](#) assertions). Without namespace prefixes, constructing correct and namespace-aware XPath expressions is extremely complicated. For example, a correct, namespace-aware XPath expression that will match the top-level Envelope element in a SOAP 1.1 message without using prefixes is:

```
/*[local-name()="Envelope" and namespace-uri()
="http://schemas.xmlsoap.org/soap/envelope"]
```

Since the `http://schemas.xmlsoap.org/soap/envelope` namespace URI is typically declared in the namespace map under the prefix "soapenv", the same XPath expression can be rewritten just as correctly as:

```
/soapenv:Envelope
```

To allow a given prefix to be used in the XPath expression in an XPath-based assertion, an entry for the prefix must be present in the assertion's namespace map. Using the custom mapping feature, a namespace map can be defined and saved in the [Evaluate Request XPath](#) assertion. For XML applications, custom namespace mapping is essential for defining the XPath query pattern for incoming XML request messages. For SOAP web services, the namespace list is automatically populated with the namespaces declared in the WSDL. New namespaces can be added and removed from the default namespace list, but the default namespaces themselves cannot be removed or changed.

Namespace prefixes that may appear in messages received by the Gateway are unrelated to those found in the namespace map of XPath-based assertions, even if they may happen to be identical. In order for an XPath expression using prefixes to match namespace-qualified nodes in a message, the message's namespace URIs must exactly match those found in the assertion's namespace map. For example, consider the following simplified SOAP 1.1 message:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope">
  <Body>
    <someApplication:operationName
xmlns:someApplication="urn:example.com:someApp"/>
  </Body>
</Envelope>
```

Even though this message's Body element appears without a namespace prefix, it is still qualified with the standard SOAP 1.1 namespace URI. The naive XPath expression `"//Body"` might match this particular message, but would fail to match another message that happened to employ a prefix for the SOAP 1.1 namespace URI. With the namespace map feature, the expression `"//soapenv:Body"` can match any message that uses the correct namespace URI, regardless of what prefix, if any, it uses to declare it.

Editing the Namespace Map

You cannot change or remove the namespaces originating from the WSDL document, but you can modify the namespace map in the following assertions:

- [Encrypt Element](#) (see page 346)
- [Evaluate Request XPath](#) (see page 458)
- [Evaluate Response XPath](#) (see page 461)
- [Sign Element](#) (see page 407)
- [Require XPath Credentials](#) (see page 248)

➤ To modify the namespace map:

1. Open the properties for any of the assertions listed above.
2. Click **[Edit Namespaces]**. The Edit Namespaces and Prefixes dialog appears with the default WSDL namespaces and prefixes.

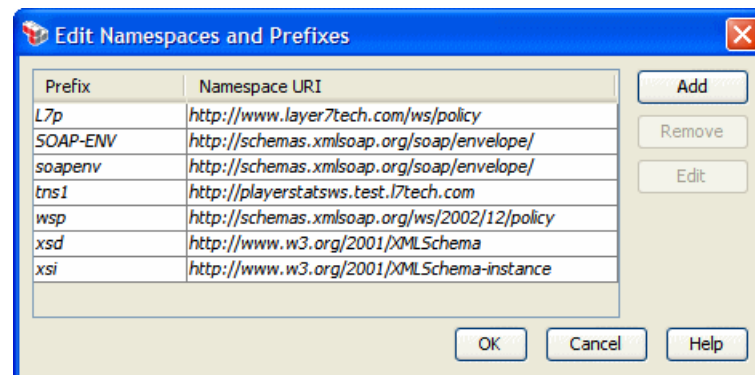


Figure 43: Edit Namespaces and Prefixes dialog

3. Choose an action to perform:

Table 35: Namespace map actions

To...	Do this
Add a new namespace prefix	<ol style="list-style-type: none"> 1. Click [Add]. 2. Enter the Prefix and Namespace URI for the new

To...	Do this
	namespace. 3. Click [OK]
Edit a user-defined namespace prefix	<ol style="list-style-type: none"> 1. Select a user-defined namespace prefix to edit. These are shown in bold. 2. Click [Edit]. 3. Modify the Prefix or Namespace URI. 4. Click [OK]. <p>Note: Only user-defined namespaces may be edited (those entered using the [Add] button). Predefined namespaces cannot be modified.</p>
Delete a user-defined namespace prefix	<ol style="list-style-type: none"> 1. Select the user-defined namespace prefix. These are shown in bold. 2. Click [Remove]. The entry is removed immediately. <p>Note: Only user-defined namespaces may be removed (those entered using the [Add] button). Predefined namespaces cannot be removed.</p>

4. Click **[OK]** to close the dialog.

Migrating Namespaces

The Migrate Namespaces feature allows you to quickly update all XPath-based assertions from one namespace to another. The XPath-based assertions include the following:

- "Encode to MTOM Format Assertion" on page 437
- "Encrypt Element Assertion" on page 346
- "Evaluate Request XPath Assertion" on page 458
- "Evaluate Response XPath Assertion" on page 461
- "Require Encrypted Element Assertion" on page 400
- "Require Signed Element Assertion" on page 402
- "Require XPath Credentials Assertion" on page 248
- "Sign Element Assertion" on page 407

➤ *To migrate namespaces:*

1. In the policy window, select the assertion(s) to migrate. You can select multiple assertions by holding down the **[Ctrl]** key while selecting the assertions. To update all eligible assertions in the policy, select the first assertion in the policy, hold down the **[Shift]** key, then select the last assertion in the policy.

2. In the Policy Manager, select **[Edit] > Migrate Namespaces** from the Main Menu (browser client: use the "Edit" menu next to the "Help" menu within the client). **Tip:** If the option is not available, click on an assertion in the policy window first.

The Migrate Namespaces dialog appears.

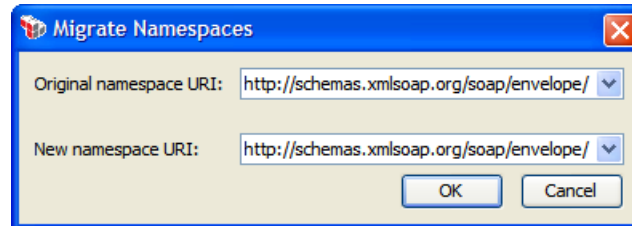


Figure 44: Migrate Namespaces dialog

3. Enter the original namespace or select a namespace from the drop-down list.
4. Enter the new namespace or select from a list of common namespaces from the drop-down list.
5. Click **[OK]**. All XPath-based assertions listed above are automatically updated to use the new namespace.

Using the XML Editor

When XML code is required in some Policy Manager forms, there is a set of built-in tools that can help you work with the code. These tools are known as the XML Editor and they are available from the following locations:

"Apply XSL Transformation Assertion" on page 424

"Managing Global Resources" on page 72

Sample Messages in the *Layer 7 Policy Manager User Manual*

"Validate XML Schema Assertion" on page 703

To access the XML Editor, right-click anywhere within the XML code box. A context menu will appear with the following options (options that are currently unavailable are dimmed):

Table 36: XML Editor options

Menu Option	Description
Document > Insert file	Inserts a file containing XML code into the box, at the location of the cursor. Once the code is inserted, you may edit it within the XML code box if necessary.

Menu Option	Description
Undo	Reverses the last action, such as character typed, text pasted/deleted, or file inserted. Multiple levels of undo are available.
Redo	Reverses the last undo action. Multiple levels of redo are available.
Cut	Removes the selected text and places it on the clipboard for pasting elsewhere.
Copy	Copies the selected text and places it on the clipboard for pasting elsewhere.
Paste	Inserts the contents of the clipboard.
Search a node	Displays a search dialog containing an expandable list of the nodes. Clicking on a node in the search dialog highlights its location in the XML code window.
XML > Parse	Parses the XML code and notifies you of any syntax errors.
XML > Format	Formats the XML code for improved readability.
XML > Comment	Used to enter text that will be automatically formatted as comments in the XML code box. The comment is inserted at the cursor location.

In addition to the XML Editor options, the following keystrokes can also be used:

Table 37: XML Editor keyboard shortcuts

Keystroke	Action
[Ctrl-A]	Selects all the text within the XML code box
[Ctrl-X]	Performs a Cut operation
[Ctrl-V]	Performs a Paste operation
[Home]	Moves the cursor to the beginning of the line
[End]	Moves the cursor to the end of the line
[Ctrl-Home]	Moves the cursor to the beginning of the XML code box
[Ctrl-End]	Moves the cursor to the end of the XML code box
[Ctrl-right arrow] [Ctrl-left arrow]	Moves the cursor one item to the right or left within a line of XML code.
[Ctrl-Shift-right arrow] [Ctrl-Shift-left arrow]	Selects the item to the right or left in a line of XML code.

Chapter 4: Access Control Assertions

Notes: (1) Depending on which Gateway product you have installed, not all the assertions shown below may be available. See Features by Product in the *Layer 7 Policy Manager User Manual* for a list of which features are available for each product. (2) This category may also include custom-created encapsulated assertions. For more information, see "Working with Encapsulated Assertions" on page 126.

In the Policy Manager, the following assertions are available in the Access Control category of the [Assertions] tab:

Authenticate Against Identity Provider Assertion	163
Authenticate Against Radius Server Assertion	164
Context Variables Created by This Assertion	164
Authenticate Against SiteMinder Assertion	167
Authenticate User or Group Assertion	170
Authenticating Against a Simple LDAP Identity Provider	171
Authorize via SiteMinder Assertion	173
Check Protected Resource Against SiteMinder Assertion	175
Exchange Credentials using WS-Trust Assertion	177
Extract Attributes from Certificate Assertion	180
Context Variables for Subject/Issuer DN	180
Context Variables for Extended Attributes	182
Extract Attributes for Authenticated User Assertion	185
Perform JDBC Query Assertion	187
Unsupported Functionality	188
Context Variables Created by This Assertion	189
SQL Query Tips	196
Caching Metadata	206
Query LDAP Assertion	209
Require Encrypted UsernameToken Profile Credentials Assertion	213
Require FTP Credentials Assertion	214
Require HTTP Basic Credentials Assertion	215
Require HTTP Cookie Assertion	215
Context Variables Created by This Assertion	216
Require NTLM Authentication Credentials Assertion	217
Context Variables Created by This Assertion	219
Creating a Computer Account for NTLM Authentication	221
Require Remote Domain Identity Assertion	226
Context Variables Created by This Assertion	227

Require SAML Token Profile Assertion	228
Context Variables Created by This Assertion	229
SAML Token Profile Wizard	231
Require SSH Credentials Assertion	237
Require SSL or TLS Transport Assertion	238
Require Windows Integrated Authentication Credentials Assertion	241
Require WS-Secure Conversation Assertion	242
Context Variable Created by This Assertion	243
Require WS-Security Kerberos Token Profile Credentials Assertion	243
Require WS-Security Password Digest Credentials Assertion	244
Require WS-Security Signature Credentials Assertion	246
Require WS-Security UsernameToken Profile Credentials Assertion	248
Require XPath Credentials Assertion	248
Retrieve Credentials from Context Variable Assertion	250
Retrieve Kerberos Authentication Credentials Assertion	251
Using the Protocol Transition Delegation Method	252
Using the Constrained Proxy Delegation Method	254
Kerberos Service Ticket/Session Caching	255
Retrieve SAML Browser Artifact Assertion	258
Use WS-Federation Credential Assertion	263

The Access Control assertions establish the authorized identities and corresponding credential authentication protocols and requirements in a [policy](#). A policy can contain more than one access control assertion, but only one can appear in a single policy execution path (parent or child assertion folder) before the single [Grant Access to Users/Groups](#) assertion to which it is assigned.

Note: The SiteMinder Protected Resource, [Sun Java System Access Manager Protected Resource](#), and [Tivoli Access Manager](#) assertions, if present, are optional [custom assertions](#) that are purchased and installed separately. For more information on acquiring custom assertions, please contact CA Technologies.

Authenticate Against Identity Provider Assertion

The *Authenticate Against Identity Provider* assertion authenticates the current credentials against a selected identity provider, using credentials gathered from a credential source assertions (for example, [Require HTTP Basic Credentials](#), [Require SAML Token Profile](#), or [Require SSL or TLS Transport](#)). It is similar to using the [Authenticate User or Group](#) assertion except that it does not match the authenticated user against any particular user or group.

Use this assertion when you need to separate authentication and authorization, for example:

- You want to authenticate the credentials already gathered in the policy, but you don't need to authorize that the resulting user is a particular user or member of a particular group.
- The policy contains many "User" or "Group" assertions. You want to authenticate first so that if it fails, the identity assertions can be skipped, saving processing time.
- You wish to perform branching based on the results of authentication (for example, "If the authentication fails, do this; otherwise do this...")

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more applying a tag to the identity, see Identity Tags in the *Layer 7 Policy Manager User Manual*.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **Change Authentication Identity Provider** dialog automatically appears; when modifying the assertion, right-click **<target>: Authenticate against...** in the policy window and choose **Change Authentication Identity Provider** or double-click the assertion in the policy window.

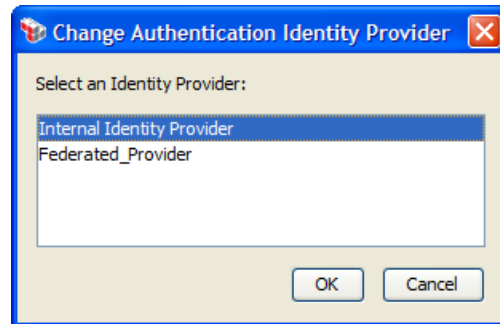


Figure 45: Changing the identity provider used for authentication

3. Choose the identity provider that will be authenticated against. Only configured identity providers appear on the list.
4. Click **[OK]** when done.

Authenticate Against Radius Server Assertion

The *Authenticate Against Radius Server* assertion is used to authenticate credentials against a RADIUS (Remote Authentication Dial In User Service) Server.

Note: This assertion only provides authentication—authorization and accounting against the RADIUS server is not supported.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Be sure to precede this assertion with a username/password credential source.

Example: The following simple policy fragment authenticates a user's HTTP Basic Credentials against the Radius server:

```

Require HTTP Basic Credentials
Request: Authenticate Against Radius Server Sample Radius Server [radius]
  
```

Context Variables Created by This Assertion

The Authenticate Against Radius Server assertion sets the following context variables.

Note: The default *<prefix>* is "radius" and can be changed in the assertion properties (Figure 46).

Table 38: Context variables created by Authenticate Against Radius Server assertion

Context variable	Description
<i><prefix>.<AttributeName></i>	Returns the value for the Radius attribute name (for example,

Context variable	Description
	<code>\${radius.User-Name}</code> will return the user's name).
<code><prefix>.reasonCode</code>	Returns a success/fail reason code from Table 39 below. Note: A reason code is not returned if the authentication failed due to invalid credentials.

Table 39: Radius reason codes

Code	Name	Description
0	Success	Authenticated against the Radius server successfully
-1	Radius Server Error	Any Radius server error (view the audit events for the failure reason)
-2	Radius Server Timeout	Radius server timed out. This code will be returned when the Radius server is unresponsive or if an invalid IP address is specified.
-3	Unknown Host	Unknown Radius server host. This code will be returned when the Radius server cannot be found or if an invalid hostname was specified.
-4	Secret Not Found	Secret key cannot be found from the stored password
-5	Configuration Error	The "Auth Port" or "Timeout" fields contain a context variable that did not resolve to a numeric value

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the **Authenticate Against Radius Server Properties** automatically appears; when modifying the assertion, right-click **<target> Authenticate Against Radius Server** in the policy window and choose **Authenticate Against Radius Server Properties** or double-click the assertion in the policy window. The properties dialog appears.

Authenticate Against Radius Server Properties

Radius Server Configuration

Host: Sample Radius Server

Secret: Sample_Secure_Password Manage Passwords

Auth Port: 1812

Timeout (sec): 5

Authentication Protocol

Authenticator: eap-mschapv2

Attributes

Name	Value
------	-------

Add Edit Remove




Radius Variable Prefix: radius

OK Cancel

Figure 46: Authenticate Against Radius Server Properties

3. Configure the properties as follows:

Table 40: Authenticate Against Radius Server settings

Setting	Description
Host 	Enter the Radius Server host name. You may reference context variables.
Secret	Choose the stored secret to use from the drop-down list. Note: Only stored passwords may be used here—you cannot type in the secret. To define a stored password, click [Manage Passwords] . For more information, see Managing Stored Passwords in the <i>Layer 7 Policy Manager User Manual</i> .
Auth Port 	Enter the Radius Server authorization port number. You may reference context variables. The default port is: 1812
Timeout 	Enter the Radius Server authorization timeout value. You may reference context variables. The default timeout is: 5 seconds

Setting	Description
Authenticator	Choose the authentication protocol to use from the drop-down list.
Attributes	<p>Optionally add Radius attributes for the access request.</p> <p>Tip: For a list of the Radius attributes, see http://freeradius.org/rfc/attributes.html. Note that the attributes <i>User-Name</i> and <i>User-Password</i> are overridden by the user credentials.</p> <p>To add an attribute:</p> <ol style="list-style-type: none"> 1. Click [Add]. 2. Enter a Name and Value for the attribute. You may reference context variables for the Value, but not the Name. 3. Click [OK]. <p>To modify an attribute:</p> <ol style="list-style-type: none"> 1. Select the attribute and then click [Edit]. 2. Modify the Value. You may reference context variables for the Value, but not the Name. 3. Click [OK]. <p>To remove an attribute:</p> <ol style="list-style-type: none"> 1. Select the attribute to remove. 2. Click [Delete]. The attribute is removed.
Radius Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>Note: Context variables are not supported for the prefix. If a variable is specified, the assertion will strip away the enclosing characters and use the variable name itself as the prefix. For example, specifying the variable "\${abc}" will result in "abc" as the variable prefix.</p>

4. Click **[OK]** when done.

Authenticate Against SiteMinder Assertion

The *Authenticate Against SiteMinder* assertion is used to authenticate credentials against the CA SiteMinder Policy Server.

For a description of the context variables that this assertion can set or use, see Context Variables for CA SiteMinder in the *Layer 7 Policy Manager User Manual*.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Tip: The *Authenticate Against SiteMinder* assertion offers a different solution for interacting with the CA SiteMinder policy server as compared to the current custom [Authenticate with SiteMinder R12 Protected Resource](#) assertion. The policy-based approach is more flexible than what is employed by the custom assertion. The *Authenticate Against SiteMinder* assertion also offers advanced features such as caching SSO tokens and multiple authorizations of the token.




Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **Authenticate Against SiteMinder Properties** automatically appears; when modifying the assertion, right-click **Authenticate Against SiteMinder [<prefix>]** in the policy window and choose **Authenticate Against SiteMinder Properties** or double-click the assertion in the policy window. The properties dialog appears.

Figure 47: Authenticate Against SiteMinder Properties

3. Configure the properties as follows:

Table 41: Authenticate Against SiteMinder settings

Setting	Description
SiteMinder Variable Prefix	<p>Enter a prefix that will be added to the context variables created and used by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy. This field is required.</p> <p>For a list of the variables set by this assertion, see Context Variables for CA SiteMinder in the <i>Layer 7 Policy Manager User Manual</i>.</p>
Credentials 	<p>Choose where to retrieve the credentials to authenticate:</p> <ul style="list-style-type: none"> • Use Last Credentials: Choose this option to use the most recently-collected user credentials of the specified type (under "Supported Credential Types"). This is the default. • Specify Credentials: Choose this to use the specific credentials entered under "Supported Credential Types". <p>Tip: See "Understanding the Credential Combinations" below for additional information.</p>
Supported Credential Types 	<p>Specify the credentials to be used for authentication. Note: If the Credentials option is "Use Last Credentials", then at least one credential type must be selected, otherwise the assertion will fail during policy execution.</p> <ul style="list-style-type: none"> • Username Password: Select this option to use basic authentication credentials to authenticate the user. Enter the Username if you have chosen to specify the credentials. You may reference context variables. This is the default. • X509 Certificate: Select this option to authenticate a user via a client certificate. Enter the subject name under Certificate CN or DN if you have chosen to specify the credentials. You may reference context variables. <p>The subject name of the X509 certificate can be a fully-specified DN (in which case it is matched exactly) or the CN attribute of a DN (in which case it is matched against just the CN value).</p> <p>Tip: See "Understanding the Credential Combinations" below for additional information.</p>
Use SSO Token from Context Variable 	<ul style="list-style-type: none"> • Select this check box to specify a context variable containing the SiteMinder SSO Token, then enter the name of the context variable that will contain this token. • Clear this check box to not use the SSO Token for authentication. Collected user credentials will be used instead (for example, via the Require HTTP Basic Credentials assertion).

Understanding the Credential Combinations

The Authenticate SiteMinder Properties offers multiple combinations of credentials settings for flexibility. Here is a brief explanation of the results of various combinations:

- If you select "Use Last Credentials" and then select both the "Username Password" and "X.509 Credentials" check boxes, the actual credentials used will depend on the authentication scheme present in the policy:
 - If only HTTP is used, then the X.509 Credentials is ignored.
 - If only client certificate authentication is used, then the Username Password is ignored.
 - If *both* authentication schemes are present in the policy, then the client certification authentication is chosen first, followed by HTTP Basic.
- If you select "Use Last Credentials" and then fail to select a credential type, then the service policy will fail because no credentials are collected.
- If you select "Specify Credentials" and then select both credential type options, then you must enter the appropriate credentials for the same user, otherwise authentication will fail during policy execution.
- If you select "Specify Credentials" and then fail to select a credential type option, an error will be displayed when you try to close the properties.

4. Click **[OK]** when done.

Authenticate User or Group Assertion

The *Authenticate User or Group* assertion allows you to authenticate users and/or groups from specific LDAP Identity Providers, Simple LDAP Identity Providers, Federated Identity Providers (FIP), or Internal Identity Providers (IIP), using credentials gathered from a credential source assertions (for example, [Require HTTP Basic Credentials](#), [Require SAML Token Profile](#), or [Require SSL or TLS Transport](#)).

If you need to add more than one user or group to a policy, add several Authenticate User or Group assertions into an [At Least One Assertion Must Evaluate to True](#) folder.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more applying a tag to the identity, see Identity Tags in the *Layer 7 Policy Manager User Manual*.

Authenticating Against a Simple LDAP Identity Provider

The authentication process differs slightly when a Simple LDAP Identity Provider is involved:

- Only users, not groups, can be authenticated against a Simple LDAP.
- When using the Policy Manager interface to search for a user in a Simple LDAP, the LDAP server is not actually consulted and no validation of the user name is performed by either the Policy Manager or the Gateway. The user name will always be displayed in the Search Results window, even if no such user exists (in other words, a "virtual" user is created).

Note: The Gateway will reject the user if the user name contains characters not permitted by the regular expression defined in the *ldap.simple.username.pattern* cluster property.

- At policy runtime, the Authenticate User or Group assertion succeeds only if the username and password provided by the client authenticates successfully *and* if the client-provided username matches the "virtual" user name from the Authenticate User or Group assertion.

Using the Assertion

1. Add the assertion to the policy development window using one of the methods described in [Adding an Assertion](#).

Tip: You can also right-click within either the "[All assertions must evaluate to true](#)" or "[At least one assertion must evaluate to true](#)" assertion folders and then choose **Add User or Group**.

The Search Identity Provider dialog appears:

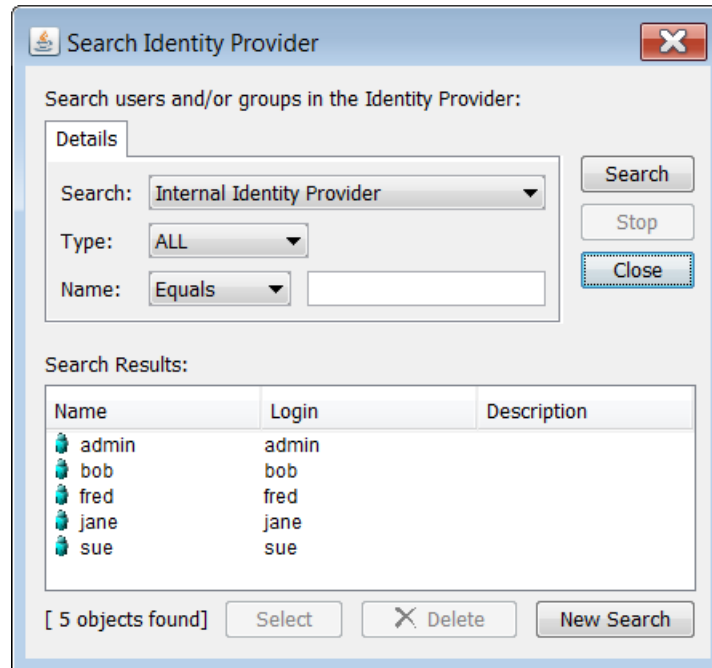


Figure 48: Search Identity Provider dialog

2. Configure your search details as follows:

Table 42: Search Identity Provider settings

Detail	Description
Search	Choose the identity provider that contains the target user and/or group.
Type	Specify whether to search for groups, users, or all. Note: Groups are not supported when authenticating against a Simple LDAP Identity Provider.
Name	Optionally refine your search by specifying whether the name should be Equal to or Starts with a specific string of characters. You can use the asterisk (*) wildcard to match any number of characters, or the question mark (?) to match any single character. Note: The "Starts with" and "Equals" settings have no effect when search a Simple LDAP Identity Provider.

3. Click **[Search]**. Matching groups/users appear in the Search Results box. Note that if searching against a Simple LDAP Identity Provider, the user will always be "found" (see ["Authenticating Against a Simple LDAP Identity Provider"](#) above for details).
4. Choose the users and/or groups to be added to the policy.

You can choose a continuous block of rows by dragging the mouse over the rows you want; or, choose the first row, hold down the **[Shift]** key, then choose the last row. You can choose individual rows by holding down the **[Ctrl]** key while clicking on the rows you want.

5. Click **[Select]**. The Search Identity Provider dialog closes and an assertion for each user or group is added to the policy development window.
6. Repeat this process to grant access to other users or groups.

Authorize via SiteMinder Assertion

The *Authorize via SiteMinder* assertion is used to authorize a user against the CA SiteMinder Policy Server. This assertion also sets a SiteMinder cookie and adds it to the response.

For a description of the context variables that this assertion can set or use, see Context Variables for CA SiteMinder in the *Layer 7 Policy Manager User Manual*.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Setting SiteMinder Cookies

Prior to version 8.2.0, SiteMinder cookies were set in the Authorize via SiteMinder assertion. This functionality has now been moved to the "Manage Cookie Assertion" on page 512. As a result:

- Instances of the Authorize via SiteMinder assertion in use prior to v8.2.0 will continue to display the SiteMinder cookie controls until the "Set SiteMinder Cookie" check box is deselected. At this point, the cookie controls are removed and the properties will resemble Figure 50. To set a SiteMinder cookie in the future, use the "Manage Cookie Assertion" on page 512 (see Figure 49 for an example).
- New instances of the Authorize via SiteMinder assertion added to a policy in version 8.2.0 or later will only display the properties shown in Figure 50. If a SiteMinder cookie is required, use the "Manage Cookie Assertion" on page 512 to set it (see Figure 49 for an example).

The following policy sample shows how you might replace the Setting SiteMinder Cookies functionality. Note that the name of the cookie is SMSESSION by default and the value is `${siteminder.smcontext.ssotoken}`. Note the double quotes in this context variable; these quotes are required in this instance.

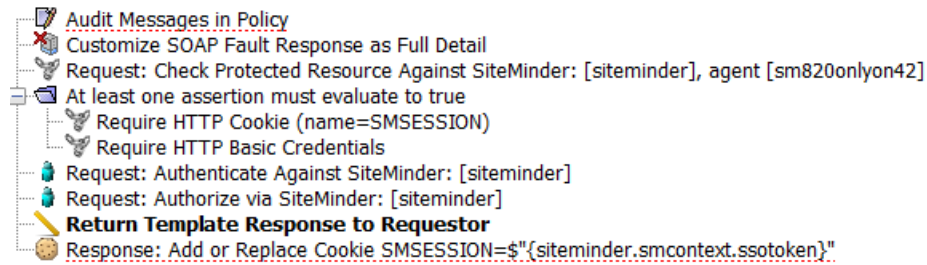


Figure 49: Sample policy for setting the SiteMinder Cookie

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **Authorize via SiteMinder Properties** automatically appears; when modifying the assertion, right-click **Authorize via SiteMinder: [<prefix>]** in the policy window and choose **Authorize via SiteMinder Properties** or double-click the assertion in the policy window. The properties dialog appears.

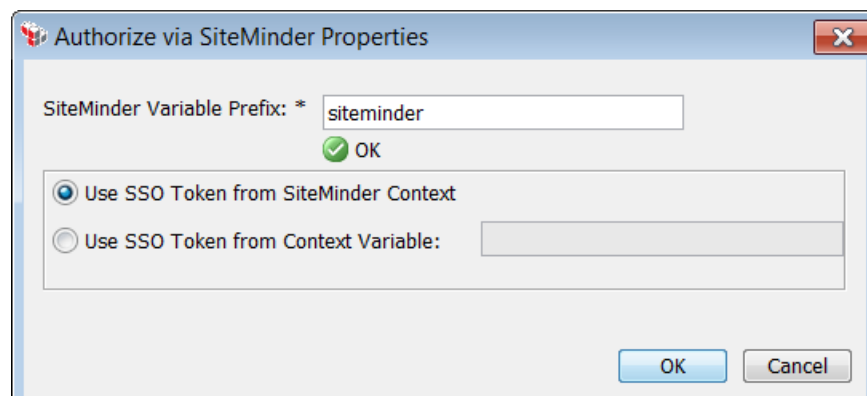



Figure 50: Authorize via SiteMinder Properties

3. Configure the properties as follows:

Table 43: Authorize via SiteMinder settings

Setting	Description
SiteMinder Variable Prefix	Enter a prefix that will be added to the <i>smcontext</i> context variables created and used by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when

Setting	Description
	multiple instances of this assertion appear in a policy. This field is required. For a list of the variables set by this assertion, see Context Variables for CA SiteMinder in the <i>Layer 7 Policy Manager User Manual</i> .
<location of SSO Token> 	Specify where to obtain the SSO Token: <ul style="list-style-type: none"> • Use SSO Token from SiteMinder context: Select this option to attempt to gather the SSO token from the SiteMinder context object. For more information about the SiteMinder context object, , see Context Variables for CA SiteMinder in the <i>Layer 7 Policy Manager User Manual</i>. • Use SSO Token from Context Variable: Select this option to obtain the SSO token from the context variable specified in the adjacent box.

4. Click **[OK]** when done.

Check Protected Resource Against SiteMinder Assertion

The *Check Protected Against SiteMinder* assertion is used to determine whether the specified resource (URL) is protected via a CA SiteMinder Policy Server, and then it establishes the authentication method.

For a description of the context variables that this assertion can set or use, see Context Variables for CA SiteMinder in the *Layer 7 Policy Manager User Manual*.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion




1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **SiteMinder Check Protected Resource Properties** automatically appears; when modifying the assertion, right-click **Check Protected Resource Against SiteMinder <prefix>, agent <SM configuration name>** in the

policy window and choose **SiteMinder Check Protected Resource Properties** or double-click the assertion in the policy window. The properties dialog appears.

Figure 51: SiteMinder Check Protected Resource Properties

3. Configure the properties as follows. All fields are required.

Table 44: SiteMinder Check Protected Resource settings

Setting	Description
Configuration Name	Choose the SiteMinder configuration to use from the drop-down list. These configurations are defined using the Manage SiteMinder Configurations task.
Agent 	Enter the name of the CA SiteMinder agent associated with the resource. You may reference context variables.
Protected Resource 	Enter the name of the resource being protected by the CA SiteMinder Policy Server. You may reference context variables.
Action 	<p>Choose an action for the Web Agent from the drop-down list:</p> <p>GET POST PUT</p> <p>Other actions may be available, depending on the CA SiteMinder Policy Server Rule associated with the domain of the protected resource. You may specify a context variable in lieu of choosing from the drop-down list. This field is blank by default.</p>
Source IP Address	Optionally, specify the source IP address that is used in the authentication/authorization procedure. You may reference context variables.

Setting	Description
	<p>Notes: (1) If a source IP is not specified, then the client's remote address from the target message is used. If this remote address is null, then the value of the Address field from the SiteMinder Configuration Properties is used instead. (2) The source IP address is ignored if the IP Check check box in the the SiteMinder Configuration Properties is not selected.</p>
SiteMinder Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>For a list of the variables set by this assertion, see Context Variables for CA SiteMinder in the <i>Layer 7 Policy Manager User Manual</i>.</p>

- Click **[OK]** when done.

Exchange Credentials using WS-Trust Assertion

The *Exchange Credentials using WS-Trust* assertion takes credentials gathered by a preceding credential source assertion, such as the transport-level [Require HTTP Basic Credentials](#) or message-level [Require WS-Security UsernameToken Profile Credentials](#) assertions, and sends them via a WS-Trust RequestSecurityToken (RST) SOAP request to a WS-Trust Security Token Service (STS). If the resulting SOAP response is a RequestSecurityTokenResponse (RSTR) and not a fault, and its RequestedSecurityToken element contains a valid security token (either a SAML token or a UsernameToken) the assertion will replace the current request's credentials with that token. If the message's original credentials were XML-based, then the XML element containing those credentials will be removed from the message and replaced with the RequestedSecurityToken element.

For more information about the Security Token Service, see Working with the Security Token Service in the *Layer 7 Policy Manager User Manual*.

W A R N I N G

The Exchange Credentials using WS-Trust assertion will be invalidated if the routing assertion in the policy is set to remove processed Security headers. When using the Exchange Credentials using WS-Trust assertion, you must configure the Route via HTTP(S) assertion to maintain the Security header in the message. To do so, select the "Leave current Security header in request before routing" option in the HTTP(S) Routing Properties that is used by both assertions. If the credentials in a message are covered by an XML Signature using the Sign Element assertion, then the signature will be invalidated when the credentials are replaced by the Exchange Credentials using WS-Trust assertion.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **WS-Trust Credential Exchange Properties** automatically appear; when modifying the assertion, right-click **Exchange Credentials using WS-Trust Request...** in the policy window and select **WS-Trust Credential Exchange Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 52: WS-Trust Credential Exchange Properties

3. Configure the properties as follows:

Table 45: WS-Trust Credential Exchange settings

Setting	Description
WS-Trust Namespace	<p>From the drop-down list, select the WS-Trust namespace to use. This will determine the version of WS-Trust used by the assertion.</p> <ul style="list-style-type: none"> • <Not Specified>: The system default WS-Trust namespace is used. • http://docs.oasis-open.org/ws-sx/ws-trust/200512: When this namespace is selected, the RST request messages will no longer use the "wst:Base" element for the token in "Issue" requests. Instead, a security header will be added to the message containing the token and a timestamp. • http://schemas.xmlsoap.org/ws/2005/02/trust: When this namespace is selected, the RST message uses the selected namespace and corresponding "RequestType"; the "Base" element is used. This namespace is typically used when "<Not Specified>" is selected. <p>Note: When this namespace is used, the cluster property <code>wss.decorator.wsTrustRequestTypeIndex</code> is respected. Changes to this property requires a restart of the Gateway. For more information, see Gateway Cluster Properties.</p> • http://schemas.xmlsoap.org/ws/2004/04/trust: When this namespace is selected, the RST message uses the selected namespace and corresponding "RequestType"; the "Base" element is used.
Token Service URL	<p>Enter the complete URL of the WS-Trust Security Token Service (STS).</p> <p>The STS must be running and configured to accept RequestSecurityToken (RST) requests containing the values configured below</p>
wsp:Applies to URI	<p>Enter a URI that describes the service for which the token will be used. For example: <code>urn:example.com:services:echo</code> or <code>http://services.example.com/EchoService</code>.</p>
wst:Issuer URI	<p>Enter a URI that describes the issuer (identity provider) of the security token being sent. For example, <code>urn:example.com:users</code> or <code>http://example.com</code>.</p>
Request Type	<p>Select the type of request from the drop-down list.</p> <p>Tip: Most WS-Trust Security Token Service implementations deal with validation requests.</p>

4. Click **[OK]** when done.

Extract Attributes from Certificate Assertion

The *Extract Attributes from Certificate* assertion extracts information from the X.509 Certificate of the last authenticated user and places them in context variables. Specifically, the subject/issuer DN fields are parsed and made available as context variables, as well as some extended attributes.

You can create a custom prefix to be added to context variables created by this assertion, to help make the context variables more readily identified.

In a policy, the Extract Attributes from Certificate assertion must be preceded by:

- At least one credential source assertion:
 - [Require SSL or TLS Transport with Client Authentication](#)
 - [Require WS-Secure Conversation](#)
 - [Require WS-Security Signature Credentials](#)
 - [Require SAML Token Profile](#) (*Subject Confirmation: Holder of Key, Require Message Signature*)
- An identity assertion (for example, [Authenticate User or Group](#))

Context Variables for Subject/Issuer DN

The Extract Attributes from Certificate assertion sets the following context variables for the subject/issuer DN in an X.509 certificate. **Note:** The default *<prefix>* is "certificate" and can be changed in the assertion properties (Figure 53).

The sample values shown are based on the following example subject DN:

cn=jsmith, OU=support, OU=IT, OU=Services, DC=acmecorp, DC=org, C=US

Table 46: Context variables for Subject/Issuer DN in an X.509 certificate

Context variable	Description
<code>\${<prefix>.subject.dn}</code>	Contains the subject DN in a format that is easier to read.
<code>\${<prefix>.subject.dn.canonical}</code>	Contains the subject DN in a format suitable for comparisons (limited subset of entity ID names; strict sorting, whitespace, and case rules).
<code>\${<prefix>.subject.dn.rfc2253}</code>	Contains the subject DN in a format that is technically precise, yet maintains readability. This only includes RFC 2253 entity ID names.
<code>\${<prefix>.subject.cn}</code>	Contains the "cn" value of the subject (e.g., jsmith)

Context variable	Description
<code>\${<prefix>.subject.ou}</code>	Contains the "ou" values of the subject (e.g., support, IT, Services)
<code>\${<prefix>.subject.dc }</code>	Contains the "dc" value of the subject (e.g., acmecorp, org)
<code>\${<prefix>.subject.c }</code>	Contains the "c" value of the subject (e.g., US)
<code>\${<prefix>.subjectPublicKeyAlgorithm }</code>	Contains the Name of the Signature Algorithm for the certificate (e.g., "SHA1withRSA")
<code>\${<prefix>.subjectEmail}</code>	Contains the email address (if any) from the Subject DN
<code>\${<prefix>.subjectAltNameEmail}</code>	Contains the email address (if any) for the Subject Alternative Name (rfc288) (e.g., "example2@oasis-open.org")
<code>\${<prefix>.subjectAltNameDNS}</code>	Contains the DNS Name address (if any) for the Subject Alternative Name (e.g., "example2.oasis-open.org")
<code>\${<prefix>.subjectAltNameURI}</code>	Contains the Uniform Resource Identifier (if any) for the Subject Alternative Name (e.g., "http://example2.oasis-open.org/")
<code>\${<prefix>.issuer.dn}</code>	Contains the issuer DN in a format that is easier to read.
<code>\${<prefix>.issuer.dn.canonical}</code>	Contains the issuer DN in a format suitable for comparisons (limited subset of entity ID names; strict sorting, whitespace, and case rules).
<code>\${<prefix>.issuer.dn.rfc2253}</code>	Contains the issuer DN in a format that is technically precise, yet maintains readability. This only includes RFC 2253 entity ID names.
<code>\${<prefix>.issuer.c }</code>	Contains the "c" (CountryName) value of the issuer
<code>\${<prefix>.issuer.cn}</code>	Contains the "cn" (CommonName) value of the issuer
<code>\${<prefix>.issuer.dc }</code>	Contains the "dc" (DomainComponent) value of the issuer
<code>\${<prefix>.issuer.l }</code>	Contains the "l" (LocalityName) value of the issuer

Context variable	Description
<code>\${<prefix>.issuer.o}</code>	Contains the "o" (OrganizationName) value of the issuer
<code>\${<prefix>.issuer.ou}</code>	Contains the "ou" (OrganizationalUnitName) value of the issuer
<code>\${<prefix>.issuer.st }</code>	Contains the 'st' (StateorProvinceName) value of the issuer
<code>\${<prefix>.issuer.street }</code>	Contains the 'street' (StreetAdress) value of the issuer
<code>\${<prefix>.issuerEmail}</code>	Contains the email address (if any) from the Issuer DN
<code>\${<prefix>.issuerAltNameEmail}</code>	Contains the email address (if any) for the Issuer Alternative Name (rfc288)
<code>\${<prefix>.issuerAltNameDNS}</code>	Contains the DNS Name address (if any) for the Issuer Alternative Name
<code>\${<prefix>.issuerAltNameURI}</code>	Contains the Uniform Resource Identifier (if any) for the Issuer Alternative Name

Note: If an attribute is not recognized, the following variable will be created for it:
`${<prefix>.subject.oid.1.2.3}`, where "1.2.3" is the dotted-decimal entity ID of the attribute.

Context Variables for Extended Attributes

The Extract Attributes from Certificate assertion sets the following context variables for the extended attributes of an X.509 certificate. **Note:** The default `<prefix>` is "certificate" and can be changed in the assertion properties (Figure 53).

Table 47: Context variables for extended attributes in an X.509 certificate

Context variable	Description
<code>\${<prefix>.countryOfCitizenship}</code>	Contains the country of citizenship. Since there can be multiple values, this is an array of 2-letter country codes.
<code>\${<prefix>.signatureAlgorithmName}</code>	Contains the Name of the Signature Algorithm for the certificate (e.g., "SHA1withRSA")
<code>\${<prefix>.signatureAlgorithmOID}</code>	Contains the entity ID of the Signature Algorithm for the certificate (e.g., "1.2.840.113549.1.1.5")
<code>\${<prefix>.serial}</code>	Contains the Certificate Serial#
<code>\${<prefix>.notAfter}</code>	Contains the Certificate Not After Date (e.g.,

Context variable	Description
	"2018-03-19T23:59:59.000Z")
\${<prefix>.notBefore}	Contains the Certificate Not Before Date (e.g., "2005-03-19T00:00:00.000Z")
<i>Key usage information are stored in the following variables. If no key usage extension is present in the certificate, the criticality is set to "none" and all the Boolean variables are set to "false".</i>	
\${<prefix>.keyUsage.criticality}	Whether the extension is present; if so whether it is critical. The following values are used: <ul style="list-style-type: none"> • none = extension not present • noncrit = extension is present but not critical • critical = extension is present and critical
\${<prefix>.keyUsage.digitalSignature}	Digital signature (true/false)
\${<prefix>.keyUsage.nonRepudiation}	Non Repudiation (true/false)
\${<prefix>.keyUsage.keyEncipherment}	Key Encipherment (true/false)
\${<prefix>.keyUsage.dataEncipherment}	Data Encipherment (true/false)
\${<prefix>.keyUsage.keyAgreement}	Key Agreement (true/false)
\${<prefix>.keyUsage.KeyCertSign}	Key Certificate Sign (true/false)
\${<prefix>.keyUsage.cRLSign}	CRL Sign (true/false)
\${<prefix>.keyUsage.decipherOnly}	Decipher Only (true/false)
<i>Extended key usage information is stored in the \${<prefix>.extendedKeyUsage} variable. If no extended key usage information is present in the certificate, the criticality is set to "none" and the arrays are empty.</i>	
\${<prefix>.extendedKeyUsage.criticality}	Whether extended key usage is present; if so whether it is critical. The following values are used: <ul style="list-style-type: none"> • none = extended information not present • noncrit = extended information is present but not critical • critical = extended information is present and critical
\${<prefix>.extendedKeyUsage}	Contains the extended key usage information, stored as an array of strings. Each value is a dotted-decimal entity ID.
\${<prefix>.certificatePolicies}	Contains certificate policies information, stored as an array of strings. Each value is a dotted-decimal entity ID.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **Certificate Attributes Properties** automatically appear; when modifying the assertion, right-click **Extract Attributes from Certificate** in the policy window and select **Certificate Attributes Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

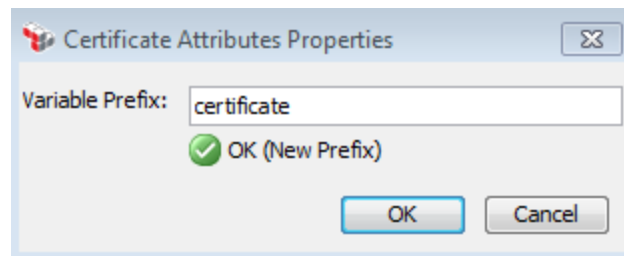


Figure 53: Certificate Attributes Properties

3. Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.

The default variable prefix is **certificate**.

For an explanation of the validation messages displayed, see Context Variable Validation in the *Layer 7 Policy Manager User Manual*.

4. Click **[OK]** when done.

Extract Attributes for Authenticated User Assertion

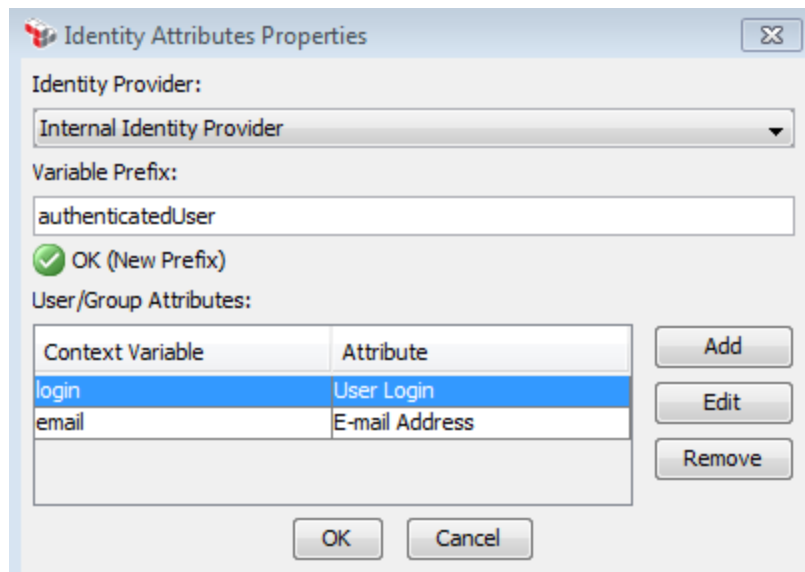
The *Extract Attributes for Authenticated User* assertion is used to create context variables based on the attributes of a previously authenticated user. The context variables created here are primarily intended to be used by the [Create SAML Token](#) assertion, but they can be read by any assertion that uses context variables.

The context variables created by this assertion have user-defined names.

Note: The *Extract Attributes for Authenticated User* assertion must be placed after the [Authenticate User or Group](#) assertion. If the Gateway is unable to authenticate a user, then no context variables will be created.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the properties automatically appear; when modifying the assertion, right-click **Extract Attributes for Authenticated User** in the policy window and select **Identity Attributes Properties**. The assertion properties are displayed.



Identity Attributes Properties

Identity Provider:
Internal Identity Provider

Variable Prefix:
authenticatedUser

OK (New Prefix)

User/Group Attributes:

Context Variable	Attribute
login	User Login
email	E-mail Address

Buttons: Add, Edit, Remove, OK, Cancel

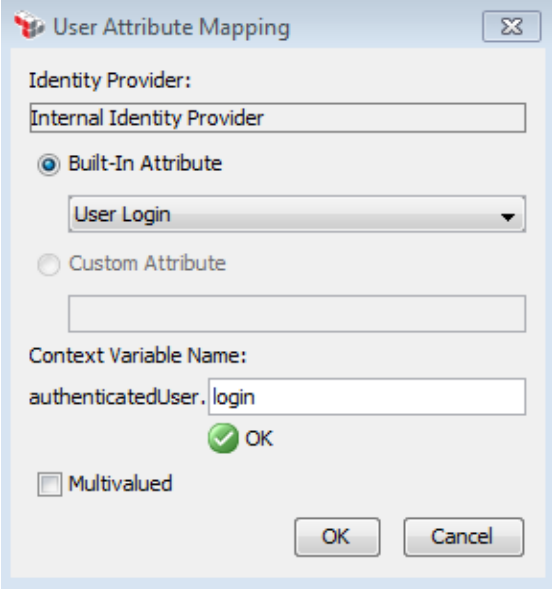
Figure 54: Identity Attributes Properties

3. Configure the properties as follows:

Table 48: Identity Attributes settings

Setting	Description
Identity Provider	Select the identity provider from the drop-down list.
Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>The default is authenticatedUser.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>
User/Group Attribute	<p>This table lists the context variables that have been created for the identity provider and the attributes to be extracted from the authenticated user. Choose one of the following actions:</p> <ul style="list-style-type: none"> To add a new context variable, click [Add] and then complete the User Attribute Mapping dialog in step 4. To change a context variable in the list, select it, click [Edit], and then complete the User Attribute Mapping dialog in step 4. To remove a context variable in the list, select it and then click [Remove].

4. If adding or editing a context variable, the User Attribute Mapping dialog appears:



The image shows a 'User Attribute Mapping' dialog box. It has a title bar with a close button. Inside, there's a section for 'Identity Provider' with a dropdown menu showing 'Internal Identity Provider'. Below that, there are two radio buttons: 'Built-In Attribute' (which is selected) and 'Custom Attribute'. Under 'Built-In Attribute', there's a dropdown menu showing 'User Login'. Under 'Custom Attribute', there's an empty text field. Below these, there's a section for 'Context Variable Name:' with a text field containing 'authenticatedUser.' followed by a dropdown menu showing 'login'. There's a green checkmark icon and the text 'OK' next to the dropdown. At the bottom, there's a checkbox labeled 'Multivalued' which is unchecked. At the very bottom, there are 'OK' and 'Cancel' buttons.

Figure 55: User Attribute Mapping dialog

5. Configure the dialog as follows:

Table 49: User Attribute Mapping settings

Setting	Description
Identity Provider	The identity provider that was selected on the Identity Attributes dialog, displayed here for your reference.
Built-In Attribute	<p>Select this option to create a context variable based on a predefined attribute in the system. Choose the attribute to use from the drop-down list.</p> <p>Note: Not all identity providers can provide every attribute shown in the list. If you select a combination that results in no attribute, the resulting context variable will have no value.</p>
Custom Attribute	Select this option if you are using an LDAP identity provider and you wish to use an attribute not in the built-in list. Type the name of the custom attribute to use. The validator will give you instant feedback as to whether the attribute contains valid characters.
Context Variable Name	The system displays the name of the context variable that will be created, based on the attribute specified and the prefix entered on the previous screen. You may edit the attribute portion of the name if necessary. The validator will give you instant feedback as to whether the variable name contains valid characters
Multivalued	<p>Select this check box if the variable is expected to hold multiple values and all values from the attribute should be stored in the context variable.</p> <p>Clear this check box if the context variable is not expected to be multivalued. Only the first value is stored, even if multiple values are present.</p> <p>For more information on using multivalued variables, including delimiter characters and concatenation options, see <i>Working with Multivalued Context Variables</i> in the <i>Layer 7 Policy Manager User Manual</i>.</p>

- Click **[OK]** when done.

Perform JDBC Query Assertion

The *Perform JDBC Query* assertion is used to query an external database and use the query results later. The query results are stored in context variables created by this assertion.

Tip: To support using multivalued context variables in the JDBC Query assertion, you can build up the values of such a variable using the "Manipulate Multivalued Variable Assertion" on page 642.

Before you can perform a JDBC query, be sure a JDBC connection has been configured through the Manage JDBC Connections task.

W A R N I N G

Do not create connections to the Gateway's MySQL database in general. Any query which writes to this database may render the Gateway inoperable.

The Perform JDBC Query Assertion is able to write to a variety of databases, even during a "Test". Ensure you are aware of the changes you are making as they are irreversible through the JDBC Query Properties.

Unsupported Functionality

Note the following functionality is not currently supported by the Perform JDBC Query assertion:

- Functions and Procedures are not supported on DB2.
- Functions and Procedures are supported on MySQL only when the database name is provided in the JDBC connection URL. Only supported when using the native MySQL driver and not DataDirect.
- Functions and Procedures with nested function calls are not supported.
- PL/SQL blocks are not supported.
- Calling overloading procedures or functions is not supported.
- Calling functions on MySQL Enterprise Edition using the Data Direct MySQL driver is not supported.
- Calling functions on Oracle via the native driver is not supported.
- Calling functions that return a Boolean with the DataDirect driver is not supported.
- Procedures and functions with lowercase names are not supported in Oracle.
- Functions and procedures in Oracle that return NCLOBs or NBLOBs with values greater than 32KB cannot be called from the CA API Gateway.
- The Boolean parameters BOOLEAN and BOOL are not supported for the native MySQL driver.

Context Variables Created by This Assertion

The Perform JDBC Query assertion sets the following context variables with the query results. **Note:** The default `<prefix>` is "jdbcQuery" and can be changed in the assertion properties (Figure 57).

Table 50: Perform JDBC Query context variables

Variable	Description
<code><prefix>.<column_name></code>	Returns the column name specified in the SQL query. This variable is created when one result set is returned.
<code><prefix>.resultSet1.<column_name></code>	If a stored procedure returns multiple result sets, then "resultSet1...N" will be added to the name of the variable.
<code><prefix>.queryresult.count</code>	Returns the number of records returned by the query (if using a SELECT query) or the number of records affected by the query (if using a non-SELECT query). This variable is always created.
<code><prefix>.xmlResult</code>	Returns the XML results and is created when the Generate XML Results check box is selected in the properties.
<code><prefix>.multipleResultSet.count</code>	This variable is set when there is more than one result set. This will occur when a called procedure returns more than one result set. This variable is not set if there is only a single result set. Tip: If OUT parameters are also set, they will be counted as one result set..
<code><prefix>. multipleResultSet.queryresult.count</code>	This variable is only set when there is more than one result set. If it is set, it contains the number of results in total across all result sets. Tip: If OUT parameters are also set, these will count as one row.
<code><prefix>.<out_parameter></code>	Returns the name of an OUT parameter from a procedure.
<code><prefix>.return <prefix>.RETURN_VALUE</code>	These variables are set by calling a function, with the name of the variable depending on the DBMS (see Table 52).

During policy consumption, the Gateway will create one multivalued context variable per column name. The number of values in the multivalued context variable corresponds to the number of records returned.

Tip: If you want to use names other than the SQL column names in the context variables, you can specify a mapping in the assertion properties.

Understanding Result Set Variables and Multiple Result Sets

The Perform JDBC Query assertion supports multiple result sets. When a SQL query is run (for example, "select * from my_table"), a "result set" is returned. This result set is a logical set of rows, with each row made up of a series of columns. When you (for example) select column_a and column_b from my_table, and there are 10 rows in my_table, the result set will contain 10 rows with each row having 2 columns.

When you call a procedure (for example, "CALL MY_PROC"), it is possible that the procedure may return more than one result set (uncommon but possible). When this happens, the `<prefix>.multipleResultSet.count` variable will be set to "2" if two result sets were returned. If the result set #1 has 10 rows and result set #2 has 5 rows, then the `<prefix>.multipleResultSet.queryresult.count` variable will be set to "15".

For example, consider a stored procedure that returns the following result sets:

+-----+-----+-----+-----+			
group	id	name	value
+-----+-----+-----+-----+			
set1	8	name8	test value8
set1	3	name3	test value3
+-----+-----+-----+-----+			
+-----+-----+-----+-----+			
group	field1	field2	field3
+-----+-----+-----+-----+			
set2	1	extra1	test1
set2	4	extra4	test4
+-----+-----+-----+-----+			
+-----+-----+-----+-----+			
group	id	name	value
+-----+-----+-----+-----+			
set3	6	name6	test value6
set3	5	name5	test value5
+-----+-----+-----+-----+			

Figure 56: Multiple result sets example

These will be the context variables that will be returned (using the default prefix "jdbcQuery"):

```

${jdbcQuery.resultSet1.group} = set1,set1
${jdbcQuery.resultSet1.id} = 8,3
${jdbcQuery.resultSet1.name} = name8,name3
${jdbcQuery.resultSet1.value} = test value8,test value3

```



```
${jdbcQuery.resultSet2.group} = set2,set2  
${jdbcQuery.resultSet2.field1} = 1,4  
${jdbcQuery.resultSet2.field2} = extra1,test1  
${jdbcQuery.resultSet2.field3} = extra4,test4  
  
${jdbcQuery.resultSet3.group} = set3,set3  
${jdbcQuery.resultSet3.id} = 6,5  
${jdbcQuery.resultSet3.name} = name6,name5  
${jdbcQuery.resultSet3.value} = test value6,test value5  
  
${jdbcQuery.multipleResultSet.count} = 3
```

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **JDBC Query Properties** automatically appear; when modifying the assertion, right-click **Perform JDBC Query** in the policy window and select **JDBC Query Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

JDBC Query Properties

JDBC Connection
Choose one connection: Sample MySQL Community Edition

SQL Query
☐ Convert Variables to Strings
 Query Timeout: 0 seconds
 WARNING: Use of non-select queries may destroy data or the database.

```
select * from contacts where id in (${myparams})
```


☐ Specify Schema:

☒ Save Results to Context Variables *split bar for resizing*

Context Variables Naming

Column Label	Variable Name
name	test.name

Prefix: jdbcQuery
☒ OK (New Prefix)



Other Settings
 Query Name (GUI use only, Optional):
 The maximum number of records returned by the query: 10
☒ Fail assertion if there are no results returned by the query.
☐ Generate XML Result


Figure 57: JDBC Query Properties


Tip: You can use the split bar between the "SQL Query" and "Context Variables Naming" panels to adjust the relative panels sizes to suit your needs (see Figure 57).

3. Configure the properties as follows:

Table 51: JDBC Query settings

Setting	Description
<i>JDBC Connection</i>	
Choose one connection 	<p>Choose the JDBC connection being queried from the drop-down list. If the connection isn't visible in the list, you may type the connection name in the box. You may reference context variables.</p> <p>Note: If context variables are specified in the connection name, it will not be possible to test the connection using the [Test] button.</p> <p>For information on defining these connections, see Managing JDBC Connections in the <i>Layer 7 Policy Manager User Manual</i>.</p>
<i>SQL Query (see "SQL Query Tips" below for more information)</i>	
Convert Variables to Strings	<p>This check box determines how context variables are processed before being sent to the JDBC driver:</p> <ul style="list-style-type: none"> Select this check box to convert context variable contents into a string. For multivalued variables, their contents are concatenated into a single value, with the values delimited by commas. Clear this check box to add each value of a multivalued or single-value context variable as is. When the policy containing the Perform JDBC Query assertion is executed, the Gateway will construct a statement with the list of parameters per each value of the multivalued context variable. This setting is the default.
Query Timeout	<p>Enter the length of time the assertion will wait (in seconds) for a response to the query before timing out.</p> <p>Default: 0 (zero, which means use the Gateway-wide timeout, described below)</p> <p>Tips: (1) The value entered here overrides the Gateway-wide timeout setting defined by the <i>jdbcqueryManager.maxGatewayStatementTimeout</i> cluster property. The intent is to supply a shorter timeout value than the Gateway default. If a <i>longer</i> timeout value is entered, it will be ignored and the Gateway default is used instead. (2) The JDBC driver may be configured to ignore all calls to set a timeout. If it is so configured, then it is not possible for the Gateway to control the timeout for any queries using that JDBC Connection. For more information, please consult your JDBC administrator. (3) The connection property <i>EnableCancelTimeout</i> set to "true" may be needed when using the DataDirect drivers, to ensure that cancel requests to an unresponsive DBMS do not wait indefinitely.</p>
SQL query box 	<p>Enter the SQL query to perform. Be cautious about any usages of non-Select DML queries. There is no transaction management within the Perform JDBC Query assertion. Once this assertion executes, the results are permanent in the DBMS, regardless of the policy logic surrounding the assertion. You may reference context variables in the SQL query; for example:</p> <pre>SELECT column_name FROM table WHERE username = \${request.user} AND password = \${request.password}</pre>

Setting	Description
	<p>Do not enclose context variables within quotes. This applies to both SELECT and INSERT statements.</p> <p>The maximum length of the query is 4 KB (4096 characters).</p> <p>For more information about SQL queries, refer to "SQL Query Tips" and "Examples Using Functions and Procedures" within this topic.</p>
[Test]	<p>Click [Test] to verify whether the SQL query is valid on the chosen JDBC connection. You must confirm that you understand the consequences of the query before proceeding. You will see a message stating whether or not the SQL query is valid.</p> <p>Note: Testing is not possible if context variables are used in either the connection name or SQL query.</p>
Specify Schema 	<p>This check box is available only under the following conditions:</p> <ul style="list-style-type: none"> A procedure or function call is defined in the SQL query text box. <p>AND</p> <ul style="list-style-type: none"> The database is Oracle or SQL Server. <p>Since the SQL query text box does not accept a schema value as part of the query, select this check box if you need to specify a schema and then enter the name of the schema in the adjacent field (must be a string without spaces or a single-value context variable). This value is passed to the JDBC driver to allow it to obtain the correct metadata from the database.</p> <p>If the SQL query requires a schema value for a procedure or function call, select this check box and then enter the name of the schema. You may reference context variables. If an object is contained within a package, then the SQL query itself should reference the package; for example:</p> <p><i>CALL mypackage.myfunction</i></p> <p>Tip: You may need to specify a schema if your query fails with this error message: <i>"The database object either does not exist or the SQL query contains the object's schema"</i>.</p>
Context Variables Naming	
Save results to context variables	<p>This check box is located above the "Context Variables Naming" table. It is used to quickly enable or disable the saving of SQL results to the context variables specified in the table. Tip: CA recommends leaving this check box enabled. But consider disabling the saving of results if you are experiencing any memory issues.</p> <ul style="list-style-type: none"> Select this check box to operate the table normally: you can add, edit, or remove context variables and the SQL results will be saved to the variables specified. Clear this check box to disable the saving of SQL results to context variables. This will disable the table and its editing controls. Any variable defined in the table will remain. <p>Notes: (1) This check box operates independently of the Generate XML Result</p>

Setting	Description
	<p>check box below. This allows you to populate the <code>\${<prefix>.xmlResult}</code> variable even when opting to not save results to context variables. (2) The following context variables are always created, regardless of the "Save results to context variables" check box: <code>\${<prefix>.queryresult.count}</code>, <code>\${<prefix>.multipleResultSet.count}</code>, and <code>\${<prefix>.multipleResultSet.queryresult.count}</code>. These variables were described under "Context Variables Created by This Assertion" (Table 50).</p>
table	<p>This table allows you to map the SQL column headings to different names. This will change the names of the context variables created. For example, you require more descriptive variable names or if you require the names to conform to naming standards at your organization. For more information, see "Context Variables Created by This Assertion" (Table 50).</p> <p>The assertion supports multiple result sets. For more information, see the variables under "Context Variables Created by This Assertion" and "Understanding Result Set Variables and Multiple Result Sets" above.</p> <p><i>To add a mapping:</i></p> <ol style="list-style-type: none"> 1. Click [Add]. The Context Variable Naming dialog appears. 2. Enter the SQL Column Label. For example: "Column1". 3. Enter the mapping destination in Variable Name. For example: "Cust_Acct". 4. Click [OK]. This will create a context variable named <code>\${<prefix>.Cust_Acct}</code> instead of <code>\${<prefix>.Column1}</code>. <p><i>To edit a mapping:</i></p> <ol style="list-style-type: none"> 1. Select a row and click [Edit]. 2. Modify the fields as necessary. 3. Click [OK]. <p><i>To remove a mapping:</i></p> <ol style="list-style-type: none"> 1. Select a row and click [Remove]. 2. Click [Remove] to confirm. The naming will revert to the SQL column name.
Prefix 	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>You may reference context variables.</p> <p>The default prefix is jdbcQuery.</p> <p>For more information, see "Context Variables Created by This Assertion" (Table 50).</p>
<i>Other Settings</i>	
Query	Optionally enter a name for the query. This name is used only for display

Setting	Description
Name	purposes in the Policy Manager.
Maximum records per query	Specify the maximum number of records to be returned from the SQL query. The default is 10 ; this can be changed using the <i>jdbquery.maxRecords.defaultValue</i> cluster property.
Fail assertion if no results	Select this check box if you want the assertion to fail if the SQL query returns no results. Clear this check box to prevent an assertion failure on no results. (The assertion can still fail for other reasons—for example, an invalid query.)
Generate XML Result	<p>You can place the XML results of the JDBC query in a context variable.</p> <ul style="list-style-type: none"> Select this check box to store the XML string representation of every variable that was set, in the context variable <code>\${<prefix>.xmlResult}</code>. Clear this check box to not place the XML result in the context variable. <p>Note: If <code>\${<prefix>.xmlResult}</code> was populated previously, that content will remain.</p> <p>The following is an illustration of the structure of the XML result:</p> <pre><?xml version="1.0" encoding="UTF-8"?> <L7j:jdbQueryResult xmlns:L7j="http://ns.17tech.com/2012/08/ jdb-query-result"> <L7j:row> <L7j:col name="COLNAME" type= "JAVA_DATA_TYPE">DATA_VALUE</L7j:col> </L7j:row> </L7j:jdbQueryResult></pre> <p>Note: The type shown is the data type <i>after</i> the XML results have been retrieved, not the actual database field type.</p>

- Click **[OK]** when done.

SQL Query Tips

Keep in mind the following when entering an SQL query:

- All variables created by the Perform JDBC Query assertion are multivalued. As such, it not possible to use array syntax when a suffix (i.e., selector) is present. Similarly, the variable `"${jdbQuery.return_value.millis}"` will not work ("millis" can be any suffix).

For a more detailed description, see "Multivalued Variables and Selectors" in Working with Multivalued Context Variables in the *Layer 7 Policy Manager User Manual*.

- If a query requires a null value, use the context variable `$(policy.nullvalue)`. This built-in variable always resolves to null.

Tip: For Oracle databases, null values are equivalent to the empty string for VARCHAR types. Thus, another way to pass null values in Oracle is to pass the empty string. For example, both of the following would result in the same, when `nullFunction` takes a VARCHAR parameter:

```
func nullFunction ""
func nullFunction ${policy.nullvalue}
```

- If a query contains SQL functions such as `AVG()`, `MAX()`, `MIN()`, or nested `SELECT`, you should use the "AS" keyword to alias the returned value; for example:

```
SELECT max(column_name) AS alias_name FROM table;
SELECT column_name, (SELECT COUNT(*) FROM table1 WHERE conditions) AS alias_name
FROM table2 WHERE conditions;
```

Note: The "AS" keyword is fully supported on in version 7.1 or later of the Gateway. Prior to version 7.1, the "AS" keyword does not work on simple column alias definitions. For example, you attempt to join two tables with an identically named column. In version 7.1, you can use an alias to separate the two columns in the result set:

```
SELECT tableA.id, tableA.firstName, tableA.lastName AS lastNameA, tableB.lastName
AS lastNameB,
FROM tableA
INNER JOIN tableB ON tableA.id = tableB.id
WHERE tableA.city = 'Vancouver';
```

This will result in the following context variables available:

```
id
firstName
lastNameA
lastNameB
```

In version 7.0 or earlier, the "AS" keyword is ignored, which results in only these variables being available:

```
id
firstName
```

The workaround is to modify the above SQL query as follows:

```
SELECT tableA.id, tableA.firstName, CONCAT(tableA.lastName, '') AS lastNameA,
CONCAT(tableB.lastName, '') AS lastNameB,
FROM tableA
INNER JOIN tableB ON tableA.id = tableB.id
WHERE tableA.city = 'Vancouver';
```

This query uses the function `CONCAT` and the "AS" keyword takes effect. The query will now produce the expected four context variables listed above.

- The SQL query cannot reference a schema. If you need to specify a schema value, select the **Specify Schema** check box and enter the value there. **Note:** The ability to specify a schema is available only for Oracle and SQL Server.

W A R N I N G

The Policy Manager will **not** prevent you from entering a destructive SQL query. Such a query may corrupt your database irrevocably, even during testing.

Converting Variables into Strings

When writing an SQL Query that will reference context variables, you need to decide whether to use the value *converted to a string* or to use the *raw value*. When the raw value is used, it will be passed directly to the JDBC driver, which will then convert it as needed, if the value is supported.

If the type of your variable is supported by the JDBC driver but you wish to use its string value while at the same time using the raw value of other variables, then you will need to create a new variable first to convert the raw variable into a string variable. For more information, see the "Convert Variables to String" option in Table 51.

Using Functions and Stored Procedures

The query statement may contain calls to functions and stored procedures. The Gateway will determine what parameters a function or procedure require as input, output, or both, by examining the database metadata for it.

To call a stored procedure, use either the CALL OR EXEC keyword followed by the name of the procedure and then the parameters for the procedure. To call a function, use the FUNC keyword.

The procedure/function parameters can be supplied as literal values, single or multivalued context variables. These parameters can either be encoded within parentheses (for example, "CALL myproc (param1, param2,.....,paramN)") or without (for example, "CALL myproc param1, param2,, paramN").

The only parameters for a procedure that must be supplied are the IN or INOUT parameters. The Gateway will automatically handle correctly registering any OUT or INOUT parameters based on the metadata for the procedure.

Note: There is no method/syntax to bind a context variable in the 'SQL Query' text field with an OUT variable from a procedure (or any other SQL statement).

To call a function, use the FUNC keyword followed by the name of the function and its parameters. The same rules for how to supply parameters apply to functions.

The output of a procedure or function will be set automatically after it has been invoked. The following table lists the names of the output variable for each database when calling a function.

Table 52: Default output variables from a function

Database	Default variable name
MySQL (only with native driver)	return
Oracle	RETURN_VALUE
MS SQL Server	RETURN_VALUE
DB2 (not supported)	n/a

All OUT/INOUT parameters are handled automatically. When calling either a procedure or a function, consider the types of the input parameters. For types such as Date, Timestamp, and BLOB to work correctly, you need to ensure the context variable is of the correct type (Date/Time or byte []). Also ensure that the Perform JDBC Query assertion is configured to not convert variables to Strings. This allows the raw type to be passed the JDBC Driver, which can then provide any conversions it supports.

Note that when calling a nonexistent function or procedure, an exception is triggered only if the package name was also specified. For example, calling the nonexistent "mypackage.myfunction" will trigger an exception, but using "myfunction" will not trigger an exception.

Note: For Oracle databases, the function and procedure names should be in uppercase and must not contain spaces or special characters. *Exception:* It is possible to call a function or procedure containing lowercase characters in its name provided that it was not created using quotes around the name.

Messages Returned for Application Users

When a user is connected to the database as an application user (not as the schema owner), the following messages will be returned when a valid stored procedure or function is called with no parameters:

Table 53: Messages for procedures or functions with no parameters

Database	Message returned
MySQL	"query testing failed: [I7tech][MySQL JDBC Driver][MySQL]No database selected"

Database	Message returned
Oracle	"query testing failed: [!7tech][Oracle JDBC Driver][Oracle]ORA-06564: object <object name> does not exist ORA-06512: at "sys.dbms_utility", line 156 ORA-06512: at line 1
MS SQL Server	"query testing failed: [!7tech][SQL Server JDBC Driver][SQLServer]Could not find stored procedure 'sp name'."
DB2	n/a (procedures and functions not currently supported in DB2)

Known Issues

Note the following known issues:

- There is a known issue in MS SQL databases where the OUT parameter is treated as INOUT, which may result in parameters being set improperly. To avoid this, set all the parameters (IN and OUT) explicitly in the query. For further assistance, please contact CA Technical Support.
- Stored procedures that use the OUT/INOUT parameter will always return a value "1" or greater for the variable `jdbcQuery.queryresult.count`. This is because the parameters are always returned in the results and the assertion will never fail. However, the assertion is configured to fail if there are no results.
- Functions and procedures in Oracle that return NCLOBs or NBLOBs with values greater than 32KB cannot be called from the CA API Gateway.

Examples Using Procedures and Functions

The following are some examples showing how to use keywords to execute functions and procedures. **Tip:** The name of the output variable from a function is determined by the database; "outParameter" shown below is just an example.

- Using the FUNC keyword to execute functions:

```
FUNC [package].[function]([IN parameters]...)
Sets jdbcQuery.<outParameter>
```

- Using the EXEC keyword to execute procedures:

```
EXEC [package].[procedure]([IN and INOUT parameters]...)
```

- Using the CALL keyword to execute procedures

```
CALL [package].[procedure]([IN and INOUT parameters]...)
Sets jdbcQuery.<outParameter> etc.
```

- The 'Specify Schema' text field:

using multi-valued context variables

FUNC [package].[function]({vars},{singleVar}) is equivalent to
FUNC [package].[function]({vars.1},{vars.2}....,{singleVar})

The following are examples of how to call functions and procedures including:

- How to access out values from procedures
- How to access one or more output result sets from a procedure
- How to supply non primitive types as parameters

The examples below will use the following variables:

- *{myvars}* = "multivalue1", "multivalue2" - This multivalued variable contains two values.
- *{myvar1}* = "singlevalue1" - This variable contains a single value.
- *{myvar2}* = "singlevalue2" - This variable contains a single value.

Calling a function

Example function definition on Oracle, which takes two parameters and returns a varchar2 value:

```
CREATE or REPLACE FUNCTION MY_FUNC(a IN VARCHAR2, b IN VARCHAR2)RETURN VARCHAR2
```

After calling this function there will be a single output variable set. The name of the return variable depends on the DBMS (for more information, see Table 52).

In this example, if the prefix configured is 'jdbcQuery', then the output variable will be:

jdbcQuery.RETURN_VALUE

Call with a multivalued variable:

```
FUNC MY_FUNC({myvars})
```

Call with single values:

```
FUNC MY_FUNC({myvar1}, {myvar2})
```

Call without parenthesis:

```
FUNC MY_FUNC {myvars}
OR
FUNC MY_FUNC {myvar1}, {myvar2}
```

Call with literal values:

```
FUNC MY_FUNC "input1", 'input2'
OR
FUNC MY_FUNC ("input1", 'input2')
```

Calling a procedure

Example procedure definition on Oracle, which takes 3 parameters, two of which are OUT parameters.

```
CREATE or REPLACE PROCEDURE MY_PROC (a IN VARCHAR2, b INOUT VARCHAR2, c OUT
VARCHAR2, d IN VARCHAR2)
```

After calling this function there will be two output variables set. The default values depend on the name of the OUT variables. If jdbcQuery is the prefix in use the following context variables will be set:

```
jdbcQuery.b
jdbcQuery.c
```

Tip: CALL and EXEC are interchangeable. Neither has any specific meaning; they both indicate that your SQL Query will call a procedure equally.

Call with a multivalued variable and a literal value:

```
CALL MY_PROC (${myvars}, "d value")
```

Call with a multivalued variable and a single variable:

```
EXEC MY_PROC (${myvars}, ${myvar1})
```

Tip: After all variables are processed, the number of values must match the number of expected input parameters. The position of the OUT parameters do not matter. After each context variable has been evaluated, the number of values resolved must match the number of input parameters. The values will be applied based on the order they were resolved.

In the above example the procedure would be called with the following runtime values:

```
multivalue1, multivalue2, singlevalue1
```

Call with single value variables and a literal value:

```
CALL MY_PROC (${myvar1}, ${myvar2}, 'd value')
```

As with functions the parenthesis around the parameters is optional.

Using date type parameters

Example function definition on Oracle:

```
create or replace FUNCTION DATE_FUNC (param1 IN DATE) RETURN DATE
```

Invoke the function with a literal string date value:

```
func DATE_FUNC '2012-12-31 23:55:40.99'
```

Invoke the function using a Date/Time typed context variable:

```
Set Context Variable date as Date/Time
func DATE_FUNC ${date}
```

Invoke the function using a String typed context variable:

```
${dateStr} = "2012-12-31 23:55:40.99"
func DATE_FUNC ${dateStr}
```

If the [Convert Variables to String] check box is selected, then the Date/Time variable will be converted into a string using the default formatting for a Date/Time variable. In order to work with the format required by your DBMS, you may need to explicitly format the Date/Time when referencing it in a function:

```
Set Context Variable date as Date/Time
func DATE_FUNC ${date.yyyy-MM-dd HH:mm:ss.SS}
```

In the above example the Date/Time is actually converted into a String when it is resolved at runtime. This is only needed when "Convert Variables to Strings" is being used.

Using numeric types

Example function definition on Oracle:

```
create or replace FUNCTION NUMBER_FUNC (param1 IN NUMBER) RETURN NUMBER
```

Invoke the function with a literal numeric value:

```
func NUMBER_FUNC 1243
```

Invoke the function using an Integer typed context variable:

```
Set Context Variable integer as Integer to 5
func NUMBER_FUNC ${integer}
```

Invoke the function using a String typed context variable:

```
${integerStr} = "12345"
func NUMBER_FUNC ${integerStr}
```

Using Boolean values

Example function definition on SQL Server:

```
CREATE or ALTER FUNCTION BOOL_FUNC( @a BIT, @b BIT)
```

Invoke the function with a literal boolean value:

```
func BOOL_FUNC 'false', 'true'
or
func BOOL_FUNC 0, 1
or
func BOOL_FUNC '0', '1'
```

Invoke the function with an Integer typed context variable:

```
Set Context Variable a as Integer to 0
Set Context Variable b as Integer to 1
func BOOL_FUNC ${a}, ${b}
```

Invoke the function with a String typed context variable

```
${falseStr} = "false"
${trueStr} = "true"
func BOOL_FUNC ${falseStr},${trueStr}
or
${aStr} = "0"
${bStr} = "1"
func BOOL_FUNC ${aStr},${bStr}
```

Using byte[] and BLOB values

Example function definition:

```
create or replace function BLOB_FUNC (a in BLOB) return BLOB
```

Invoke using a literal hexadecimal string:

```
func BLOB_FUNC '0123456789abcdef'
```

Invoke using a String typed context variable:

```
${hexString} = "0123456789abcdef"
func BLOB_FUNC ${hexString}
```

Invoke using a String value converted into hex using the 'Encode / Decode' assertion:

```
Base16 Encode ${myvar1} into ${hexString}
func BLOB_FUNC ${hexString}
```

Invoke using a byte[] context variable created via an Encapsulated Assertion:

```
Output a byte[] variable called ${bytes} from an encapsulated assertion
func BLOB_FUNC ${bytes}
```

Using null values

It is possible to supply a null value in a SQL Query via a special built in variable. It is also possible to supply a null value via any existing variable or multivalued variable that may contain a null value.

The predefined variable *\$(policy.nullvalue)* supports passing null values into SQL queries. This variable will only pass a null value into a SQL query when the [Convert Variables to Strings] check box is not selected.

Example function definition

```
create or replace function NULL_FUNC (a in NUMBER, b in VARCHAR2) return VARCHAR2
```

Invoke a function using a literal null value:

```
func NULL_FUNC null 'asdf'
```

Invoke a function using a context variable with a null value:

```
func BOOL_FUNC ${policy.nullvalue}, 'b value'
```

Supported Data Types

These DBMS types are supported via DML statements via the 'SQL Query' text box (see Figure 57). When calling procedures or functions they are supported as either input or output values.

Table 54: Supported data types

DBMS type	Context variable type
Char, Varchar, etc	String
Numeric (Integer, Long, etc.)	Integer or String
BLOB	String with hex values or a byte [] (not settable via the set context variable)
CLOB	String
Date	Date/Time
Timestamp	Date/Time
Boolean	String or Integer when supported by the DBMS (see "Boolean values" below)

A String value can be used to supply a value for many DBMS types—for example, all character types including CLOB, Numeric types, Date and Timestamps (if formatted correctly) and Blob (via hex strings).

Boolean values

The following are support for Boolean values for each database type:

Table 55: Boolean value support

DBMS	Support
Oracle	Not supported by either the Data Direct driver or the native driver
MySQL	0 = false, everything else = true Note: Boolean does not work with MySQL with the native driver.
SQL Server	0 or "false" = false, 1 or "true" = true
DB2	No boolean type

Date values

It is possible to supply the value for a DATE or TIMESTAMP using either literal values, String context variables, or Date/Time context variables (recommended for easiest integration).

The formatting required for a literal string value which represents a date or timestamp is determined by a number of factors, including DBMS settings and connection properties. The following default formats are known to work:

- **Oracle:** `yyyy-mm-dd hh:mm:ss.fffffffff` (for example, '1999-01-31 24:24:24:123456')
- **Other DBMS:** `yyyy-mm-dd hh:mm:ss` (for example, '1999-01-31 24:24:24')

BLOB values

To supply a BLOB value in SQL Query, either a context variable or a literal value may be used. You can also supply binary data as hexadecimal strings either via a String context variable or via a literal hex value (for example, "0123456789abcdef"). Please contact CA Technical Support for more information.

Known Oracle issues

- Function and procedure names must be in uppercase
- Binary_Float, Binary_Double, Binary_Integer, Pls_Integer are returned as strings
- Functions which return NCLOB values greater than 32kB are not supported

Caching Metadata

This section provides some insight on how the Perform JDBC Query assertion queries the database for metadata via the JDBC driver.

Function and Procedure metadata are cached. Caching can be performed:

- eagerly, via a background caching task
- lazily, when metadata is downloaded at message traffic processing time (MTPT)

Notes: (1) When caching is configured and no data is available in the cache, then the metadata will be downloaded by the message processing thread. (2) When a nonexistent function or procedure is referenced, an exception will be issued and cached only if a package name has been specified. For example, calling the nonexistent function "mypackage.myfunction" will trigger an exception. But calling the nonexistent "myfunction" will not trigger an exception.

The caching metadata is enabled by the `jdbcQueryManager.cacheMetaData.enable` cluster property set to "true" (default). Note that this property does not affect existing cached data and does not prevent background tasks from caching metadata.

For more information on the caching cluster properties, see JDBC Cluster Properties in the *Layer 7 Policy Manager User Manual*.

Background Caching of Metadata

The Gateway has a background task that downloads metadata eagerly, to ensure that it will be available at message traffic processing time. This eager caching will occur when the Gateway starts or when a policy is saved and activated; it is possible only when no context variables are used for the JDBC connection name or for the schema.

The caching of data is based on tracking a set of unique keys. Each key is defined as:

Connection name + Procedure or Function name + Optional Schema name

Therefore, if two Perform JDBC Query assertions reference the same procedure or function, only a single copy of that metadata will be kept. This background task runs every 10 minutes by default and is enabled or disabled via the `jdbcQueryManager.cacheMetaDataTask.enable` cluster property. The task interval is controlled by the `jdbcQueryManager.cacheRefreshInterval` cluster property.

To improve the processing time of this background task, you can increase the number of background processing threads when a large volume of metadata is being managed. The number of processing threads is 10 by default but can be increased up to 200 via the `jdbcQueryManager.minCacheConcurrency` cluster property. **Note:** Increase the concurrent threads with caution.

When the background task is not able to obtain the metadata for a unique key, an exception is cached. When a policy containing a Perform JDBC Query assertion for this cache key executes, it will return this cached exception (for example, the assertion may fail) repeatedly until it is cleared out by the cleanup background task, or it is corrected by the background task downloading the metadata.

The cleanup cache task runs every minute by default and is controlled by the `jdbcQueryManager.cacheCleanUpInterval` cluster property.

Life cycle of a managed key

After the background task starts to manage a unique key, at some point it may need to stop managing it. This can happen when no Perform JDBC Query assertion is referencing the unique key, however this may be difficult to track when context variables are used (for example, the unique key being referenced is unknown until context variables are resolved

during runtime).

To resolve this problem, the Gateway can track how often the data from the cache for a particular key is used. If it is over the threshold, then the background task will stop managing that meta data.

The value is defined in the `jdbcQueryManager.cacheKeyNoUsageExpiration` cluster property. The default is 31 days and the value is configured in seconds.

Automatic Lazy Caching

If the background task is not enabled but caching is allowed, then all metadata downloaded will be cached to avoid it being downloaded a second time. The unique key for this metadata will be added to the list of keys to manage.

Cache Expiration

Any cached item can be configured to expire, to prevent outdated meta from causing the Perform JDBC Query assertion to fail.

The expiration should be longer than the background task refresh interval. It should also be longer than the estimated time it takes the job to complete.

Cache expiration is controlled via `jdbcQueryManager.cacheStaleTimeout` cluster property. The default is 30 minutes and is configured in seconds.

Manually Populating the Cache

If caching is enabled, you can manually invoke it by clicking the [Test] button (see Figure 57) in the Perform JDBC Query assertion. This will cause metadata to be downloaded and cached, if not already present in the cache.

Cache Logging

Items which indicate that background tasks are running or working are logged at a FINE level. Items which indicate that the cache is working at message traffic processing time are logged at the FINEST level.

FINE logging is used for:

- When metadata is downloaded and added to the cache.
- When metadata could not be downloaded and an exception is added to the cache.
- When the task to maintain metadata in the background starts and when it finishes.

- When a unique key representing a procedure to manage metadata for is removed from the cache by the clean up task.

FINEST logging is used for:

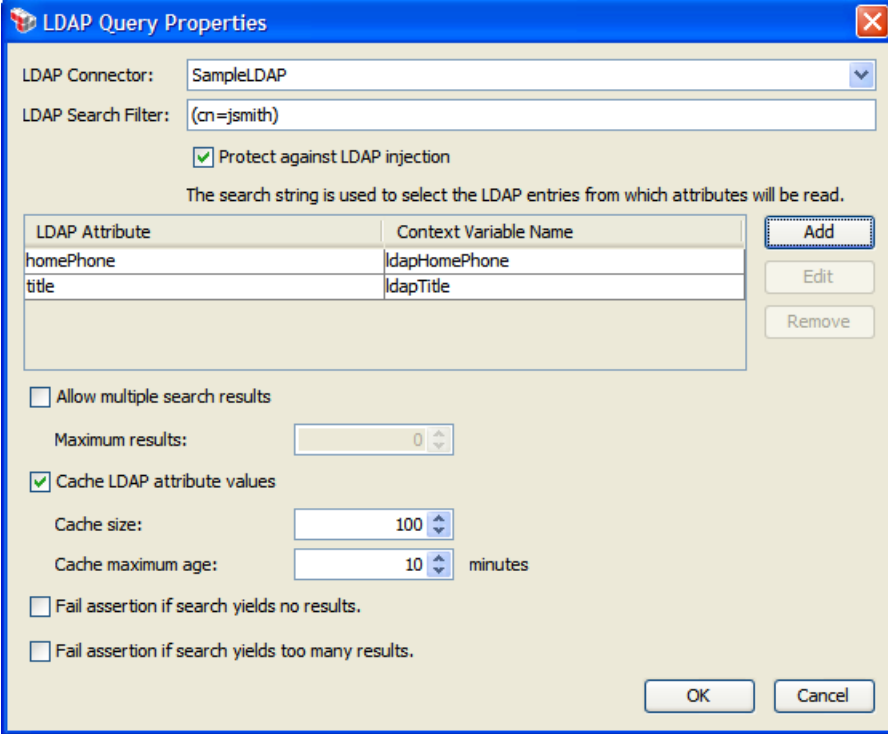
- Metadata cache hit
- Metadata cache hit but data has expired (it is stale)
- Metadata cache miss

Query LDAP Assertion

The *Query LDAP* assertion reads attributes from LDAP entries and stores them in context variables.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **LDAP Query Properties** automatically appear; when modifying the assertion, right-click **Query LDAP** in the policy window and select **LDAP Query Properties** or double-click the assertion in the policy window. The assertion properties are displayed.



The dialog box is titled "LDAP Query Properties". It contains the following fields and controls:


- LDAP Connector:** A dropdown menu with "SampleLDAP" selected.
- LDAP Search Filter:** A text box containing "(cn=jsmith)".
- Protect against LDAP injection:** A checked checkbox.
- Instructions:** "The search string is used to select the LDAP entries from which attributes will be read."
- Table:**

LDAP Attribute	Context Variable Name
homePhone	ldapHomePhone
title	ldapTitle
- Buttons:** "Add", "Edit", and "Remove" buttons are located to the right of the table.
- Allow multiple search results:** An unchecked checkbox.
- Maximum results:** A spinner box set to "0".
- Cache LDAP attribute values:** A checked checkbox.
- Cache size:** A spinner box set to "100".
- Cache maximum age:** A spinner box set to "10" minutes.
- Fail assertion if search yields no results:** An unchecked checkbox.
- Fail assertion if search yields too many results:** An unchecked checkbox.
- OK and Cancel buttons:** Located at the bottom right.

Figure 58: LDAP Query Properties

3. Configure the properties as follows:

Table 56: LDAP Query settings

Setting	Description
LDAP Connector	Select the LDAP connector to use from the drop-down list. This LDAP identity provider must already be configured in the Gateway. For more information, see <i>Creating an LDAP or Simple LDAP Identity Provider</i> in the <i>Layer 7 Policy Manager User Manual</i> .
LDAP Search Filter 	<p>Specify a search string that will be used to select the LDAP entry to query; for example: <i>(cn=jsmith)</i> or <i>(mail=jane)</i>. You may use a context variable.</p> <p>When the request is processed, the Query LDAP assertion connects to the specified LDAP connector and selects a node based on the search filter. The selected node is then used to extract the LDAP attributes; the values from those attributes are then assigned to the specified context variables based on the table below.</p> <p>Note: If the search filter matches no LDAP nodes, then the context variables specified in the table below will not be created.</p> <p>Technical Details</p> <p>When an LDAP "object" is returned to the Gateway, it has a set of attributes such as:</p>

Setting	Description
	<p>cn=jsmith ph=6045551234</p> <p>...</p> <p>Technically, the "dn" of an object is not an attribute of the object itself; rather, it represents its location (name) within the LDAP database. However, the "dn" is treated like other attributes by the Gateway.</p>
Protect against LDAP injection	<p>Select this check box to protect against an LDAP injection attack (for example, attempts to substitute a context variable in place of a static value).</p> <p>Technical Details</p> <p>When the Protect against LDAP injection check box is selected, any substituted variables are escaped in an LDAP search filter, as per RFC 2254 (section "4. String Search Filter Definition"). The following are some examples</p> <pre>myVar = user* // variable cn=\${myVar} // filter cn=user\2a // filter after variable substitution with escaping enabled</pre>
[Add]	<p>Displays the Attribute Variable Mapping dialog to create a new pairing:</p> <ul style="list-style-type: none"> • LDAP Attribute Name: Enter the name of the LDAP attribute to be stored as a context variable. • Context Variable Name: Enter the name of a new context variable to hold the attribute value. Observe the naming rules for your new variables. • If multivalued: If the attribute contains more than one value, indicate how the values should be handled: <ul style="list-style-type: none"> • Use first value: Select this option to use only the first value, regardless of how many values are present. • Join with commas: Select this option to concatenate all the values into one string, separated by commas. For example, if the context variable "medal" has three values <i>bronze, silver, gold</i>, choosing this option will define "medal" as "bronze, silver, gold". Note: Using this option, the individual elements in the value are not separately addressable as with the "Set multivalued context variable" option. • Set multivalued context variable: Select this option to place the values in an array where each element is addressable. Using the "medal" example from above, calling <code>\${medal}</code> will return "bronze, silver, and gold". Calling <code>\${medal[0]}</code> will return "bronze"; calling <code>\${medal[2]}</code> will return "gold".

Setting	Description
	<ul style="list-style-type: none"> Fail: Select this option to fail the assertion if the attribute contains more than one value. <p>For more information on using multivalued variables, including delimiter characters and concatenation options, see <i>Working with Multivalued Context Variables</i> in the <i>Layer 7 Policy Manager User Manual</i>.</p>
[Edit]	Displays the Attribute Variable Mapping dialog to modify an existing pairing.
[Remove]	Removes the selected pairing from the table.
Allow multiple search results	<p>Select this check box to allow multiple search results for the LDAP query. When this option is used the resulting context variables will always be multivalued.</p> <p>Clear this check box to not allow multiple search results.</p>
Maximum results	<p>If allowing multiple search results, enter the maximum number of results permitted. The default "0" (zero) indicates no limit.</p> <p>Tip: The setting [Fail assertion if search yields too many results] lets you control what happens when the maximum is exceeded.</p>
Cache LDAP attribute values	<p>Select this check box to cache the LDAP search results. When a search is performed, the Gateway will use cached results first. This can improve performance in environments where the data on the LDAP server changes infrequently.</p> <p>Clear this check box to not cache any LDAP search results. The LDAP directory is queried for every search.</p>
Cache size	<p>If LDAP search results are being cached, specify the maximum number of LDAP search results to be cached. A size of '0' (zero) means an unlimited cache size.</p> <p>The default is 100 for new installations of the Gateway and 0 when upgrading from a system prior to version 5.3.</p>
Cache maximum age	If LDAP search results are being cached, specify how long to cache an item before the information is discarded.
Fail assertion if search yields no results	<p>Select this check box to have the assertion fail if no search results are returned.</p> <p>Clear this check box if the assertion should always succeed, regardless of the search outcome.</p>
Fail assertion if search yields too many results	<p>Select this check box to fail the assertion if the number of search results exceeds the specified maximum.</p> <p>Clear this check box to never fail the assertion regardless of the number of search results.</p>

4. Click **[OK]** when done.

Require Encrypted UsernameToken Profile Credentials Assertion

The *Require Encrypted UsernameToken Profile Credentials* assertion requires an encrypted Username Token element to be present and that it be encrypted with the same key that was used to sign the timestamp or other parts of the message. This provides message level security without requiring a client certificate. The client creates a new symmetric key and encrypts it for the server. The encrypted symmetric key prevents the UsernameToken from being intercepted and attached to another message.

Note: This assertion only ensures that client credentials are encrypted using the same key that was used elsewhere in the message. To enforce the signing or encryption of other parts of a message, you need to include one or more of the following assertions in the policy: [Require SSL or TLS Transport](#), [Sign Element](#), or [Encrypt Element](#). If response security is configured, the response security will attempt to use (by reference) the session key used by the client in the request.

The Require Encrypted UsernameToken Profile Credentials assertion requires message security features contained in WS-Security version 1.1 or later.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click the **<target>: Require Encrypted UsernameToken Profile Credentials** in the policy window and select **Require Encrypted UsernameToken Profile Credentials Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

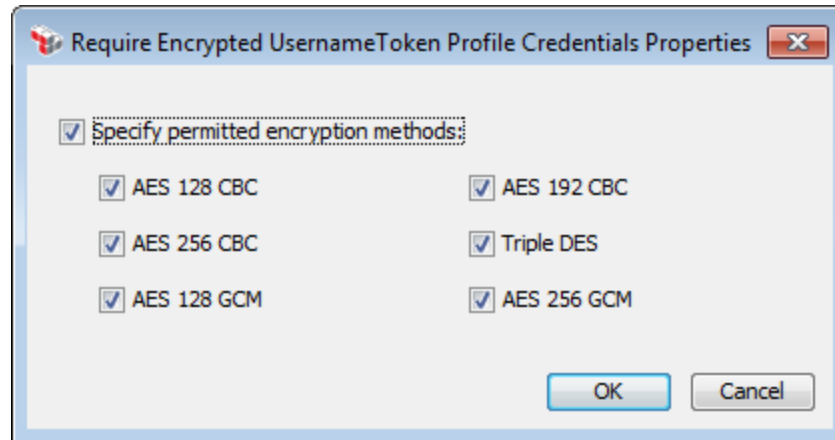


Figure 59: Require Encrypted UsernameToken Profile Credentials Properties

3. By default, all encryption methods are permitted. To choose specific methods to permit in the target message, select the **Specify permitted encryption methods** check box and select the appropriate check boxes next to:

AES 128 CBC
 AES 192 CBC
 AES 256 CBC
 Triple DES
 AES 128 GCM
 AES 256 GCM

Note: If your security provider does not support the "AES-GCM" encryption options, encryption/decryption attempts may fail at runtime if these options are selected.

4. Click **[OK]**.

Require FTP Credentials Assertion

The *Require FTP Credentials* assertion allows you to require FTP authentication—user name, plain text password. This assertion is a credential source that saves the user name and password from the FTP session for later authentication and authorization using the "Authenticate User or Group Assertion" on page 170.

Using the Assertion

- Add the assertion as described in "Adding an Assertion" on page 112.
 The assertion is added to the policy window; no further configuration is required.

Require HTTP Basic Credentials Assertion

The *Require HTTP Basic Credentials* assertion allows you to require basic HTTP authentication—user name, plain text password, and the authentication realm—as a string in the web service or XML application request headers. This assertion is a credential source that saves the user name and password from the HTTP headers for later authentication and authorization via the "Authenticate User or Group Assertion" on page 170 or the "Authenticate Against Identity Provider Assertion" on page 163. This assertion should be used in conjunction with the "Require SSL or TLS Transport Assertion" on page 267

Note the following limitations when authenticating via HTTP Basic:

- The HTTP Basic specification defines the encoding of the username and password as ISO-8859-1. As a result, it is possible to define users in the Internal Identity Provider using arbitrary encoding (for example, multi-byte characters), but these users will not be authenticated successfully over HTTP Basic. **Tip:** Consider using the "Require WS-Security UsernameToken Profile Credentials Assertion" on page 248 instead for authentication in this scenario. The WSS standard accepts arbitrary encoding.
- The Require HTTP Basic Credentials assertion does not support user names containing the ":" (colon) character.
- The Require HTTP Basic Credentials assertion should not be used in NTLM Authentication scenarios where the [Require NTLM Authentication Credentials](#) assertion is also present. Doing so may cause severe performance issues on the Gateway.

Using the Assertion

- Add the assertion as described in "Adding an Assertion" on page 112.

The assertion is added to the policy window; no further configuration is required.

Require HTTP Cookie Assertion

The *Require HTTP Cookie* assertion checks that a request contains a cookie with the same name as that specified in the assertion. If the request does not contain a cookie with this name, then the assertion fails.

The HTTP Cookie assertion does not check the validity or expiry of a cookie. It only checks for the presence of a cookie. A custom assertion such as the [Access Resource Protected by JSAM](#) assertion should be used to validate the content of the cookie.

Policy Example

The following illustrates how this assertion might be used in a policy:

"At least one assertion must evaluate to true"

Require HTTP Basic Credentials

HTTP Cookie: iPlanetDirectoryPro

Access Resource Protected by JSAM (or another custom assertion that uses cookies)

Route via HTTP(S) to URL

Note: The HTTP Cookie assertion should be positioned *after* the Require HTTP Basic Credentials assertion, within an "At least one..." folder. The sample arrangement above does not imply that the Require HTTP Basic Credentials assertion will always be used if present. This is because the custom assertion that follows will check for both a cookie and user/password credentials. If a valid cookie is found, it is used.

Context Variables Created by This Assertion

The Require HTTP Cookie assertion sets the following context variable when a cookie is found.

<prefix>.<cookieName>

Where:

- <prefix> is defined in the assertion properties (default: **cookie**)
- <cookieName> is the name of the cookie from the cookie header

For example, if the cookie header contains: var1=value1; var2=value2

The following context variables will be set:

- `${cookie.var1}`, which contains the value of cookie "var1"
- `${cookie.var2}`, which contains the value of cookie "var2"

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.

- When adding the assertion, the **HTTP Cookie Properties** automatically appear; when modifying the assertion, right-click **Require HTTP Cookie** in the policy window and select **HTTP Cookie Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

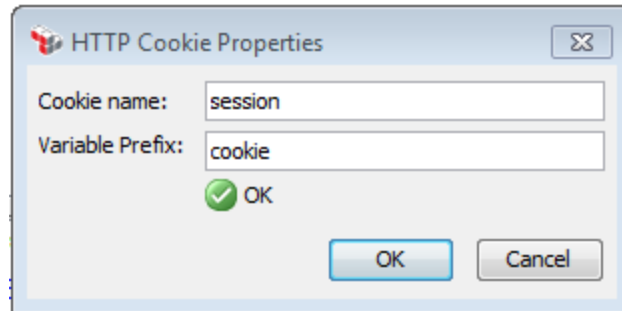


Figure 60: HTTP Cookie Properties

- Configure the properties as follows.

Table 1: HTTP Cookie Properties settings

Settings	Description
Cookie Name	<p>Enter the name of the cookie that is expected to contain the request credentials.</p> <p>Tip: If a cookie with this name is found, the cookie value is placed in the context variable: <code>\${cookie.<cookieName>}</code> (based on the default prefix).</p>
Variable Prefix	<p>Optionally, change the prefix that will be added to the context variable created by this assertion. The prefix will prevent the context variable from being overwritten if the assertion appears more than once in a policy. The default prefix is "cookie."</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

- Click **[OK]** when done.

Require NTLM Authentication Credentials Assertion

The *Require NTLM Authentication Credentials* assertion allows a single point of authentication via NTLM, in addition to the ability to receive authorization data used to control resource access. This data includes the full user name, home directory path, user account permissions and Group membership.

The connection is authenticated via an NTLM or Negotiate protocol and the CA API Gateway will challenge the requester until a security context has been established. Once the first authentication is established, the connection will continue to be authenticated until either the connection drops or the authentication times out.

Once NTLM Authentication has been established, you can use the [Authenticate Against Identity Provider](#) or [Authenticate User or Group](#) assertions to provide further authorization of the user.

Keep in mind of the following points while using the Require NTLM Authentication Credentials assertion:

- The LDAP Identity Provider used in these assertions must match the one used in the Require NTLM Authentication Credentials assertion. Changes to this assertion do not affect established connections.
- You can have only one active (executed) Require NTLM Authentication Credentials assertion in the policy. The others may be present but should not be executed at the same time. The others may be present but should not be executed at the same time, otherwise connection issues may occur.
- NTLM Proxy authentication is not supported in this version. Use pass-through NTLM instead.
- Users with single or multi-byte non-English characters in their names are not supported for NTLM authentication. (As per RFC-4120 , Kerberos Principal names cannot contain non-ASCII characters.)

Note: You may encounter NTLM connection issues when using the Chrome browser. CA recommends using the Internet Explorer or Mozilla Firefox browsers.

Prerequisites:

1. To be able to perform NTLM Authentication, a computer account with sufficient privileges must exist to call the Netlogon service on behalf of the client in the authenticating domain.
2. Trust between Active Directory domains must exist in order to perform NTLM pass-through authentication.
3. NTLM Configuration must already be enabled in the LDAP Identity Provider Wizard. For more information, see LDAP Identity Provider Wizard in the *Layer 7 Policy Manager User Manual*.

Context Variables Created by This Assertion

The Require NTLM Authentication Credentials assertion sets the following context variables for inbound NTLM requests, upon successful NTLM authentication. **Note:** The default `<prefix>` is "ntlm" and can be changed in the assertion properties (Figure 61)."

Note: The variables in Table 57 below (except for `sAMAccountName`) are set only if the user account has the corresponding values set in the Active Directory.

Table 57: Inbound context variables created by Require NTLM Authentication assertion

Setting	Description
<code><prefix>.sAMAccountName</code>	This is the pre-Windows 2000 user name, which is the only required variable prefix.
<code><prefix>.fullName</code>	Contains the full user name of the account (first and last name).
<code><prefix>.homeDirectory</code>	Contains the home directory path from the account profile.
<code><prefix>.homeDirectoryDrive</code>	Contains the home directory drive from the account profile.
<code><prefix>.userAccountFlags</code>	Sets the user flags for permissions.
<code><prefix>.session.key</code>	Contains the session key from the Netlogon server.
<code><prefix>.sid</code>	Contains the SID of the primary group.
<code><prefix>.sidGroups [index]</code>	This is a list of any additional SIDs of which the account is a member.
<code><prefix>.logonDomainName</code>	The domain to which the user is logged.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, **Require NTLM Authentication Credentials** automatically appears. When modifying the assertion, right-click **Require NTLM Authentication Credentials** in the policy window and select **NTLM Authentication Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

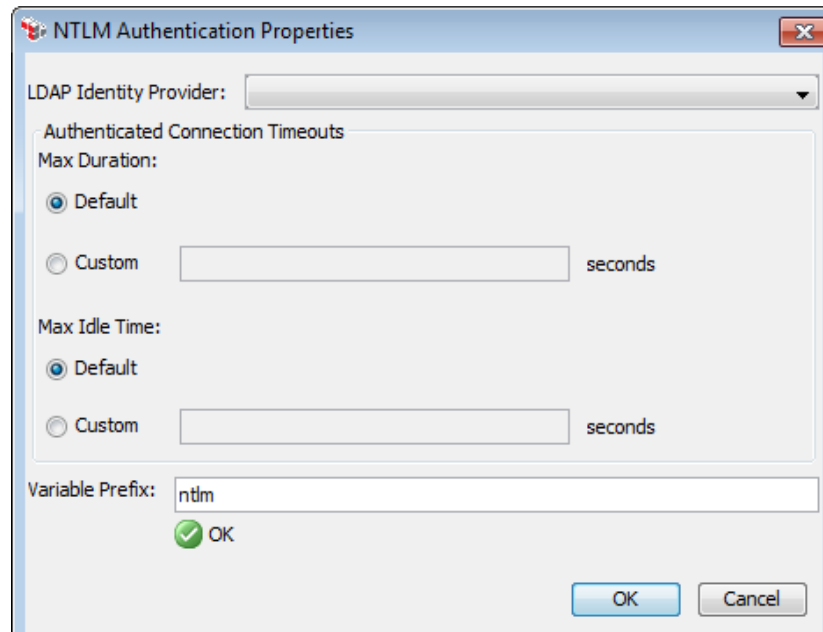


Figure 61: NTLM Authentication Properties

3. An authenticated connection can have one of the following maximum types of durations. Configure the table as follows.

Table 58: NTLM Authentication settings

Setting	Description
Default	The timeout is unlimited.
Custom	This is a timeout specific to the assertion, which can range from 0 to 2147483647.
0	This is the same as "Default", in which there is an unlimited time duration.
Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion . This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>The default prefix is ntlm.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

4. Click **[OK]** when done.

Note: Once the maximum timeout period has been reached, the Gateway will request the client to re-authenticate.

Creating a Computer Account for NTLM Authentication

In order to use NTLM Authentication, a computer account must be first be created.

The following steps are performed on the Active Directory server.

➤ To create a computer account in the Active Directory Server:

1. Start the **Server Manager** in the Active Directory.
2. Open *Active Directory Domain Services* under *Roles*.

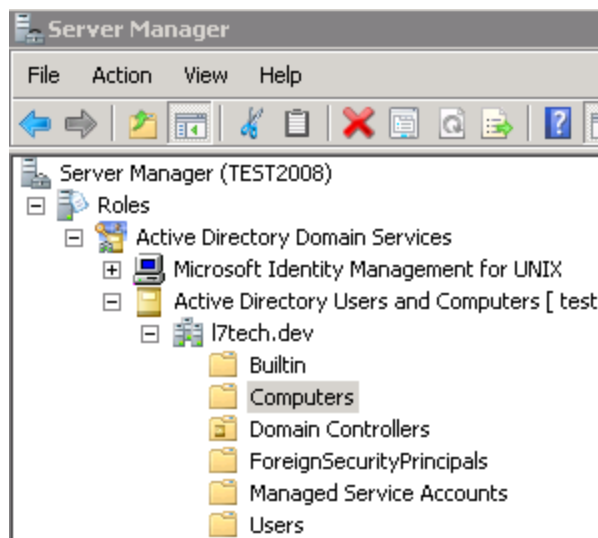


Figure 62: Server Manager

3. Expand the *Active Directory Users and Computers* node.
4. Expand the specific directory (in the sample figure above, it is "l7tech.dev").
5. Right-click the *Computers* node to create a new computer account.
6. Choose the group called "Domain Computers" in the **[Member of]** tab if it is not set by default. This is required before creating a computer account.
7. Populate the fields in the **[General]** tab. The **Computer name** field is required.

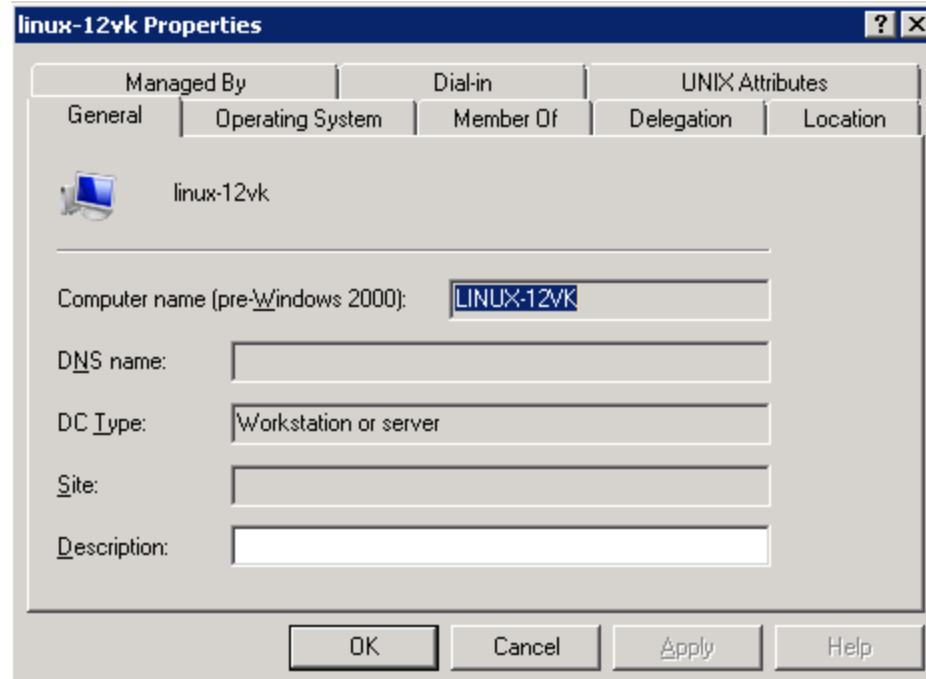


Figure 63: Linux Properties - General tab

- To configure the delegation for the newly created computer account:
1. Open the newly created computer account and select the **[Delegation]** tab.
 2. For the delegation, choose "Trust this computer for delegation to specified services only".
 3. For the trust, choose the **Use any authentication protocol** option.

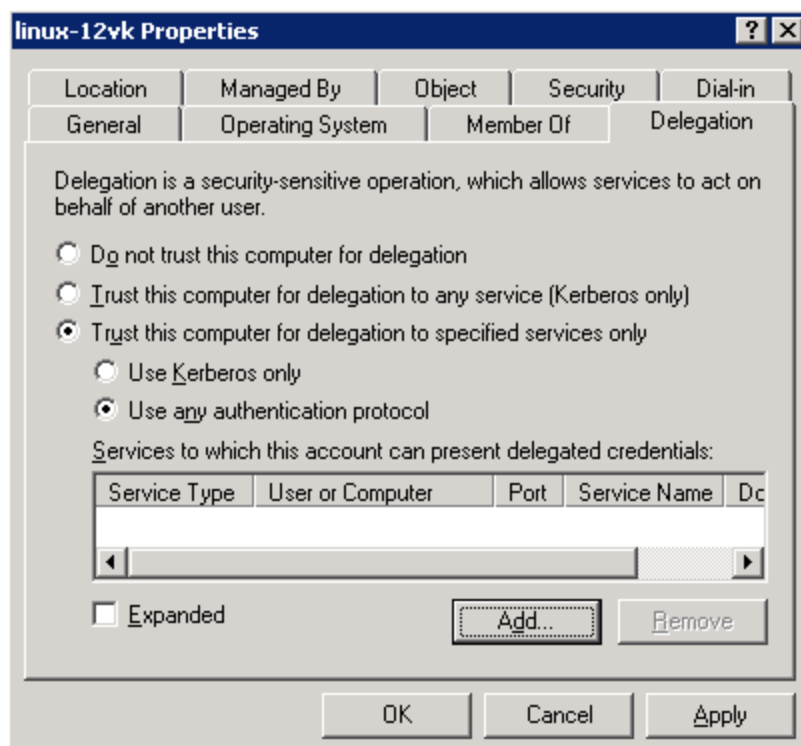


Figure 64: Configuring the [Delegation] tab for a new computer account

4. Click **[Add]**. The "Add Services" dialog appears.

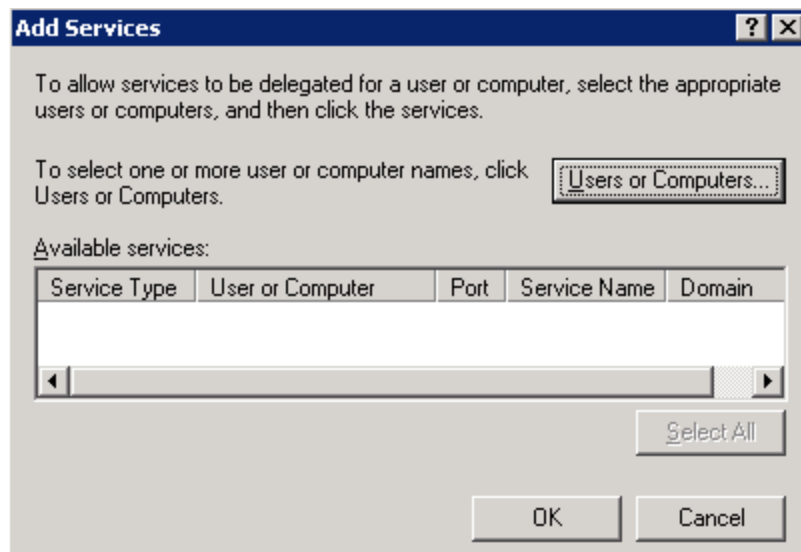


Figure 65: Add Services dialog

5. Click **[Users or Computers]**.
6. From the "Search results" list, select the server in which the netlogon service is running, and click **[OK]**.

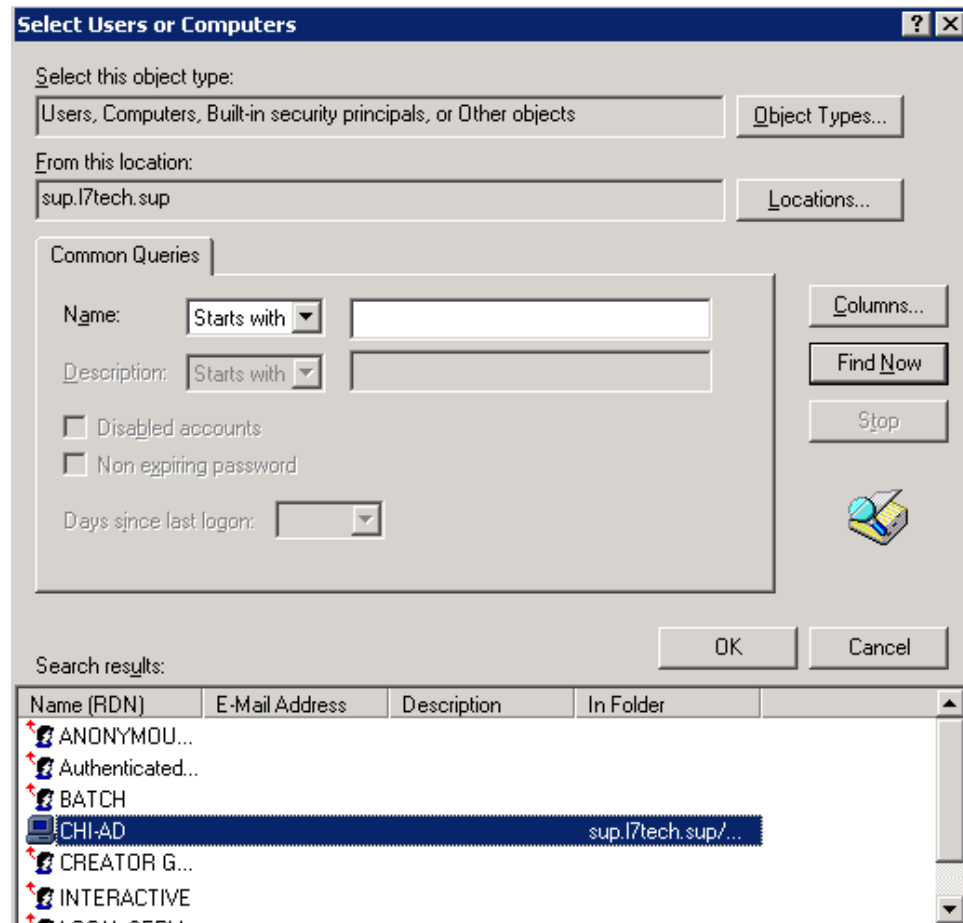


Figure 66: Select Users or Computers dialog

7. In Add Services, select **netlogon** under the Service Type column, and click **[OK]**.

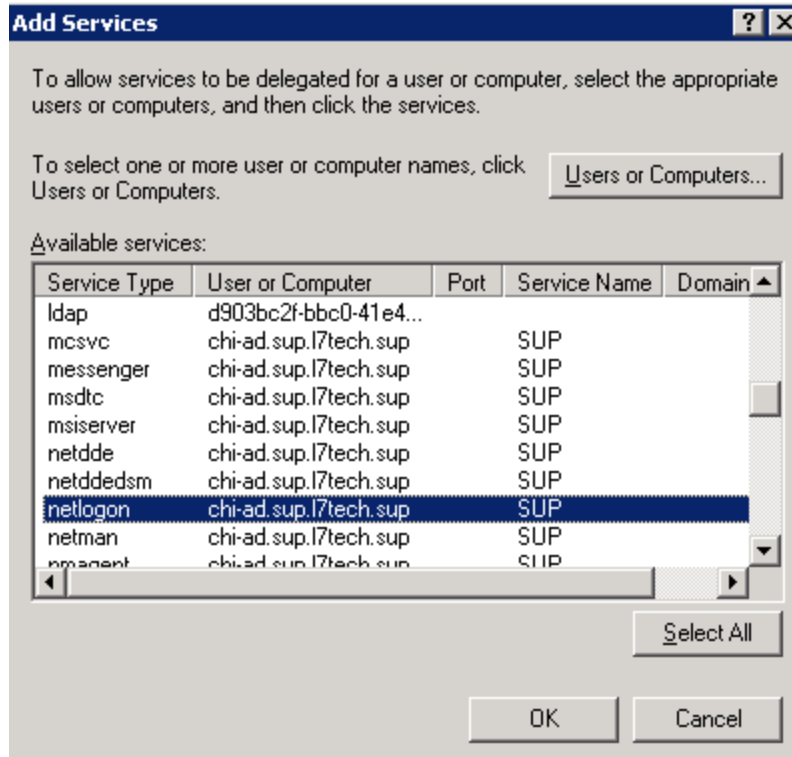


Figure 67: Add Services dialog containing available services

The **[Delegation]** tab displays the netlogon service available for the computer account.

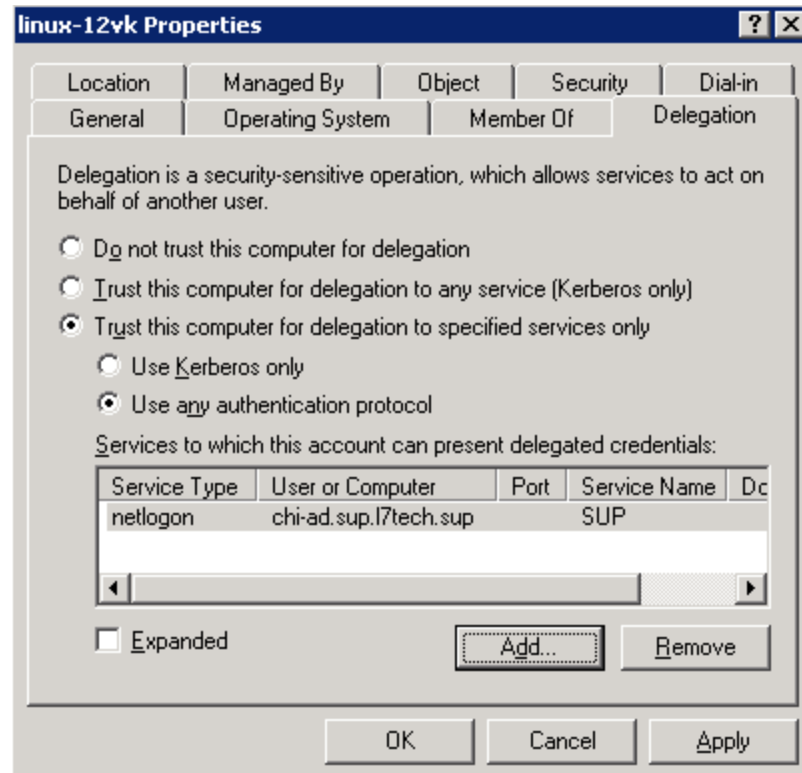


Figure 68: The netlogon service is displayed in the Delegation tab

The final step is to create a *set_password.vbs* script using the following template:

```
Dim objComputer

Set objComputer = GetObject("WinNT://<domain>/<computer account name>$")
objComputer.SetPassword "<password>"

Wscript.Quit
```

When this is complete, execute the *set_password.vbs* script.

Require Remote Domain Identity Assertion

The *Require Remote Domain Identity* assertion enables the Windows Domain Injection feature in the Securespan XML VPN Client.

- When the Securespan XML VPN Client executes this assertion, it will deduce the user name, domain name, and client program name from the operating system and then insert them into the message header. On the Securespan XML VPN Client, this assertion always succeeds.

- When the Gateway executes this assertion, it will examine the headers provided by the Securespan XML VPN Client and then create the corresponding context variables. On the Gateway, this assertion succeeds only if the context variables are set successfully.

For more information, see *Configuring Windows Domain Injection* in the Securespan XML VPN Client documentation.

Note: If identity injection has been disabled on the Securespan XML VPN Client, adding this assertion to a policy will have no effect. Conversely, if identity injection has been enabled full time, it will occur even if this assertion is not used.

Context Variables Created by This Assertion

The Require Remote Domain Identity assertion sets the following context variables. **Note:** The default *<prefix>* is "injected" and can be changed in the assertion properties (Figure 69).

Table 59: Context variables created by Require Remote Domain Identity assertion

Variable	Description
<i><prefix>.user</i>	Contains the user name from the message header.
<i><prefix>.domain</i>	Contains the domain name from message header.
<i><prefix>.program</i>	Contains the client program name from the message header.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Require Remote Domain Identity** in the policy window and select **Remote Domain Identity Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

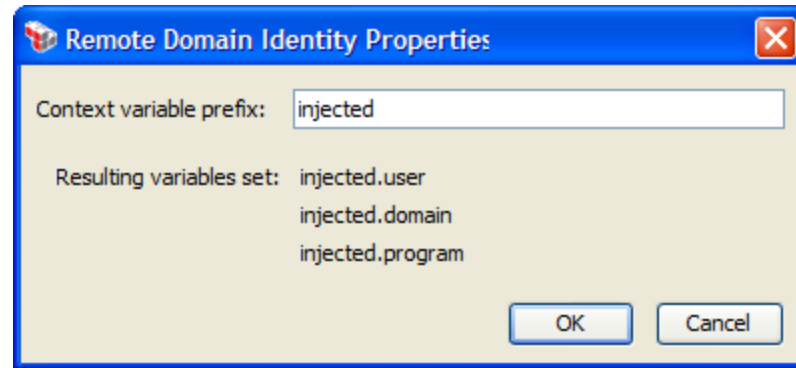


Figure 69: Remote Domain Identity Properties

3. Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.
4. Click **[OK]** when done.

Require SAML Token Profile Assertion

The *Require SAML Token Profile* assertion allows you to require SAML constraints in a policy. SAML (Security Assertions Markup Language) validates a ticket to ensure that it falls within the required constraints. If validation succeeds, then the Gateway passes the message through to the service. If validation fails, then the Gateway returns a SOAP fault.

The Require SAML Token Profile assertion is a credential source that saves subject information for later authorization via the [Authenticate User or Group](#) assertion. This assertion can be used in tandem with the [Protect Against Message Replay](#), [Sign Element](#), and [Encrypt Element](#) assertions. This assertion is also used as a credential source for an identity bridging configuration.

The Require SAML Token Profile assertion supports both the SAML 1.1 and 2.0 standards.

Note: To avoid constraint conflicts, only a single Require SAML Token Profile assertion should be present in a policy.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Context Variables Created by This Assertion

The Require SAML Token Profile assertion sets the following context variable after it is used to validate an [Attribute Statement](#).

saml.attr.<attribute_name>

Where:

- **saml.attr** is a fixed prefix for all context variables created by this assertion
- <attribute_name> is the attribute that was validated, with the following transformations:
 - name is converted to lower case
 - non-alphanumeric characters changed to underscores ('_')
 - if the attribute begins with a number, an 'n' will be prepended
 - all attribute values will be converted to a string if not already a string

Technical tip If the <attribute_name> begins or ends with white space, it cannot be accessed using the context variable described above. You must extract it using an XPath assertion instead. For assistance, contact CA Technical Support.

If an attribute contains more than one value, a multivalued context variable is created.

Examples:

An attribute named "fruit" with a single value "pear" can be accessed with the context variable `${saml.attr.fruit}`, which will yield "pear". If the attribute "fruit" contains multiple values, you can use `${saml.attr.fruit[0]}` to access the first item, `${saml.attr.fruit[1]}` for the second item, etc.

An attribute named "99 beers!" would be accessible as `${saml.attr.n99_beers_}`.

Note: Only attributes named in the assertion properties and validated are placed into context variables. Any other attributes that may be present in the SAML token are ignored (these may be validated using [schema validation](#) and/or XPath assertions if necessary).

Adding and Configuring the Assertion

1. Add the Require SAML Token Profile assertion to the policy development window as described in [Adding an Assertion](#). The **SAML Token Profile Wizard** appears.
2. Follow the wizard to complete the assertion. For details, see "SAML Token Profile Wizard" on page 231.

Editing the Assertion

1. In the policy development window, right-click **<target>: Require SAML <type> Statement** and then select **SAML Token Profile Wizard**. The wizard is displayed in edit mode.

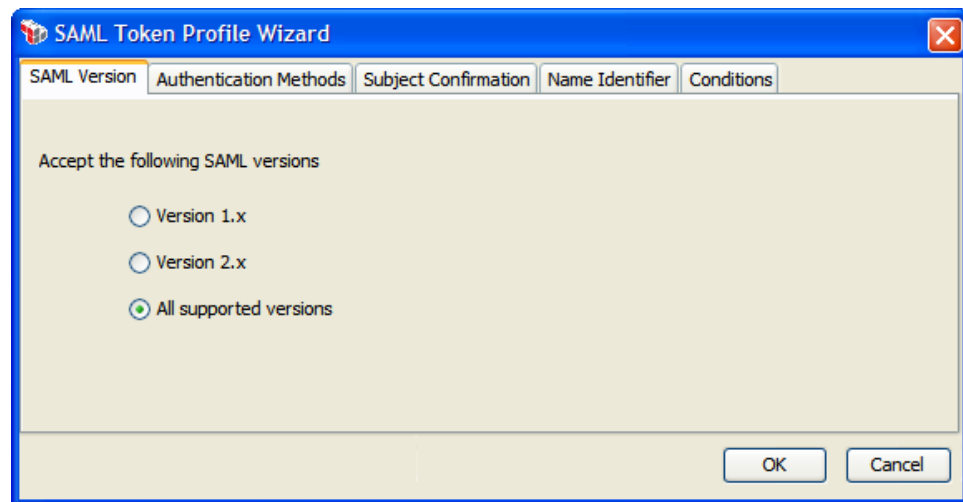


Figure 70: Example of SAML Token Profile Wizard in edit mode

2. In edit mode, each step in the wizard is represented by a tab. Select the appropriate tabs to edit. For more information about each tab, refer to Table 60 for the corresponding step in the SAML Token Profile Wizard.

Table 60: SAML Token Profile Wizard tabs in edit mode

For information on the tab...	See this step in the SAML Token Profile Wizard...
SAML Version	Step 2: SAML Version
SAML Statement Type	Step 3: SAML Statement Type
Authentication Methods	Step 4: Authentication Methods
Authorization Statement	Step 5: Authorization Statement
Attribute Statement	Step 6: Attribute Statement
Subject Confirmation	Step 7: Subject Confirmation
Name Identifier	Step 8: Name Identifier
Conditions	Step 9: Conditions

- Click **[OK]** when done.

SAML Token Profile Wizard

The *SAML Token Profile Wizard* automatically starts when you add a [Require SAML Token Profile](#) assertion to a policy.

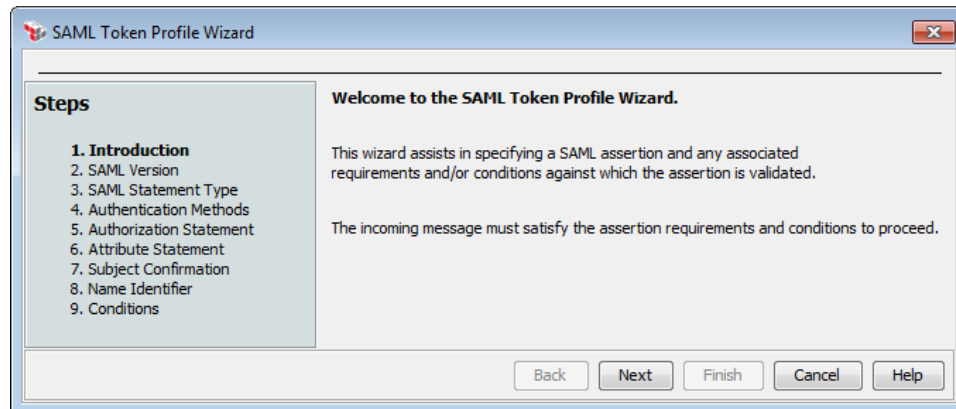


Figure 71: SAML Token Profile Wizard

For more information about wizards, see "Wizard" under Interfaces in the *Layer 7 Policy Manager User Manual*.

Table 61: Using the SAML Token Profile Wizard



Wizard Step	Description
Step 1: Introduction	Introduces the wizard.
Step 2: SAML Version	Specify which SAML versions will be accepted by the Gateway: version 1.1, version 2.0, or any supported version.
Step 3: SAML Statement Type	Select the type of SAML statement to configure: <ul style="list-style-type: none"> Authentication Statement: Proceed to Step 4. Authorization Decision Statement: Proceed to Step 5. Attribute Statement: Proceed to Step 6.
Step 4: Authentication Methods	Use the chooser list to select the authentication methods that will be enforced by the Require SAML Token Profile assertion. You must choose at least one method. Hints: <ul style="list-style-type: none"> Hold down the [Ctrl] or [Shift] keys to select multiple items at once. Click [All] to choose every available authentication method. Click [None] to quickly clear the Selected list and start again.


Wizard Step	Description
	<ul style="list-style-type: none"> • Select the Unspecified method to allow authentication by an unspecified method. • The Available list only displays the methods that are applicable to the SAML version chosen in Step 2 of the wizard. <p>In the Custom field, optionally enter any URI custom authentication methods, separated by spaces. You may reference context variables (either single- or multi-valued variables with space-separated URI values).</p> <p>Note: The SSL/TLS Certificate Based Client Authentication method is not related to the Require SSL or TLS Transport assertion. This method refers to the original authentication, not to the current request which may or may not have used SSL. The SAML-supported authentication methods are outlined in the SAML 1.1 and 2.0 specification documents provided at http://www.oasis-open.org</p> <p>Proceed to Step 7: Subject Confirmation.</p>
Step 5: Authorization Statement	<p>Specify the resource that the SAML statement must describe, the resource action, and the action namespace.</p> <ul style="list-style-type: none"> • Resource: Enter a value for the resource that the SAML statement must describe (for example, "http://acme.org"). • Action: Enter an action value for the resource (for example, "GET"). • Action Namespace: Optionally enter a corresponding action namespace value (for example, "acmeNamespace"). <p>Proceed to Step 7: Subject Confirmation.</p>
Step 6: Attribute Statement	<p>Define one or more SAML attributes that must be described by the SAML statement.</p> <ol style="list-style-type: none"> 1. Click [Add] and then complete the Edit SAML Attribute Constraints dialog: <ul style="list-style-type: none"> • Attribute Name: Enter the name of the attribute. • Attribute Namespace: Optionally enter a namespace for the attribute. This applies only to SAML 1.1. • Attribute Name Format: Optionally specify a URI reference that describes the format of the attribute name. Only attributes that declare this format will be accepted. This applies only to SAML 2.0. <ul style="list-style-type: none"> • Unspecified: If no name format is provided, the default value of <code>urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified</code> is used. • URI Reference: This option uses the URI <code>urn:oasis:names:tc:SAML:2.0:attrname-format:uri</code>. • Basic: This option uses the URI:

Wizard Step	Description
	<p><i>urn:oasis:names:tc:SAML:2.0:attrname-format:basic.</i></p> <ul style="list-style-type: none"> • Other: Select this option to define your own attribute name format in the box below. • Attribute Value: To require an exact variable match, select Specific Value and then enter a set value. To require that a particular attribute be present, but allow it to have any non-empty value rather than requiring a specific match, select the Allow any non-empty value option. <p>When a non-empty attribute value is required, you can separately validate the attribute contents using XPath expressions, transient variables, and the Compare Expression assertion.</p> <p>To modify an attribute statement, select it and click [Edit]. To delete an attribute statement, select it and click [Delete].</p> <p>2. Click [OK] to enter the attribute into the table. Repeat to configure additional attributes.</p> <p>To modify an existing Attribute Statement, select it from the list and then click [Edit].</p> <p>To remove an Attribute Statement, select it from the list and then click [Remove].</p> <p>Tip: The attribute values validated by the Attribute Statement are available in context variables. For more information, see "Context Variables Created by This Assertion" in "Require SAML Token Profile Assertion" on page 228.</p>
Step 7: Subject Confirmation	<p>Select one or more subject confirmation methods that should be accepted by the Gateway and indicate whether the message signature is required as the proof material:</p> <p>Holder-of-Key</p> <p>This allows SAML tokens that use the <i>Holder-of-Key</i> subject confirmation method (with the standard URI <i>urn:oasis:names:tc:SAML:1.0:cm:holder-of-key</i> or <i>urn:oasis:names:tc:SAML:2.0:cm:holder-of-key</i>, depending on the selected SAML version in Step 2 of the wizard). For such assertions, the Gateway will require that the subject demonstrate possession of the private key corresponding to the public key in the Subject certificate.</p> <p>The Holder-of-Key subject confirmation method currently requires that the request ticket's "SubjectConfirmation" element contain a "KeyInfo" element that contains a complete copy of the Subject's X.509 certificate. Any other form of Holder-of-Key ticket will be rejected by the Gateway.</p> <p>The request Subject may use one of two methods to prove that they hold this key:</p> <ul style="list-style-type: none"> • The request includes at least one element covered by a valid

Wizard Step	Description
	<p>WSS message signature, and the signing certificate is the Subject certificate. Or,</p> <ul style="list-style-type: none"> The request arrived over SSL/TLS with client certificate authentication, and the client certificate exactly matches the Subject certificate. <p>When the Holder-of-Key subject confirmation method is selected, you have access to the [Require Message Signature] check box:</p> <ul style="list-style-type: none"> Select this check box to require proof-of-possession using a WSS message signature. Either the message body or security header timestamp must be signed to prove possession. <ul style="list-style-type: none"> To require that the message body is signed, use the Require Signed Element assertion. To require that the timestamp is signed, use the Require Timestamp in Message assertion. Clear this check box to not require that either the message body or header timestamp be signed. In this case, the Require SSL or TLS Transport assertion must be present in the policy. <p>Sender Vouches</p> <p>This allows SAML tokens that use the <i>Sender Vouches</i> subject confirmation method (with the standard URI <i>urn:oasis:names:tc:SAML:1.0:cm:sender-vouches</i> or <i>urn:oasis:names:tc:SAML:2.0:cm:sender-vouches</i>, depending on the selected SAML version in Step 2 of the wizard). For such assertions, the Gateway will require that the sender, presumably different from the subject, vouches for the verification of the subject.</p> <p>The Sender Vouches subject confirmation method is typically used only in a SAML identity bridging policy.</p> <p>Three conditions must be met in order to use the Sender Vouches confirmation method:</p> <ul style="list-style-type: none"> An existing trust relationship with the sender ("Attesting Entity") must be configured in the Gateway. To do this, import the sender's certificate, configured as a "SAML Attesting Entity" certificate, into the Trust Store. For more information, see Managing Certificates. The SAML ticket used by the SAML token must be bound to the request message by one of the following methods: <ul style="list-style-type: none"> Send the request over SSL using the sender certificate as the SSL client certificate, OR If SSL is not used, then the SAML ticket needs to be bound to the message with a WSS signature. One complication here is that the SAML ticket does not necessarily contain or refer to the sender certificate; it usually contains or refers to the subject certificate and, assuming that the ticket is signed, contains or refers to the certificate of the ticket

Wizard Step	Description
	<p>issuer. In this method, therefore, the WSS signature must cover both the SAML token and the relevant portions of the rest of the message that use the sender certificate as the signing certificate.</p> <ul style="list-style-type: none"> The format of the request message must conform to the OASIS Web Services Security standards: SAML Token Profile 1.0 (for SAML 1.1) or SAML Token Profile 1.1 (for SAML 2.0). The Gateway does not support references to SAML tokens that are not included with the request message. <p>The OASIS Web Services Security: SAML Token Profile 1.0 standards document is available online at: www.oasis-open.org/committees/download.php/1048/WSS-SAML-06.pdf.</p> <p>When the Sender Vouches subject confirmation method is selected, you have access to the Require Message Signature check box:</p> <ul style="list-style-type: none"> Select this check box to require proof-of-possession using a WSS message signature. Either the message body or security header timestamp must be signed to prove possession. <ul style="list-style-type: none"> To require that the message body is signed, use the Require Signed Element assertion. To require that the timestamp is signed, use the Require Timestamp in Message assertion. Clear this check box to not require that either the message body or header timestamp be signed. In this case, the Require SSL or TLS Transport assertion must be present in the policy. <p>Bearer</p> <p>This allows SAML tokens that use the <i>Bearer Token</i> subject confirmation method (with the standard URI <code>urn:oasis:names:tc:SAML:1.0:cm:bearer</code> or <code>urn:oasis:names:tc:SAML:2.0:cm:bearer</code>, depending on the selected SAML version in Step 2 of the wizard). Like HTTP cookies, such assertions will always be assumed to belong to whatever message contains them, and the subject will be assumed to be the sender of the message.</p> <p>The Bearer Token subject confirmation method does not protect against an attacker modifying the message or stealing a copy of the assertion and attaching it to an unauthorized message. To protect the secrecy of the SAML token when using the Bearer subject confirmation method, be sure to select the SSL-TLS Certificate Based Client Authentication check box in Step 4 of the SAML Token Profile Wizard.</p> <p>None</p> <p>This allows SAML tokens that do not contain a subject confirmation method.</p> <p>Not having a subject confirmation method exposes the system to various threats. To protect the secrecy of the SAML token when a confirmation</p>

Wizard Step	Description
	method is not used, be sure to select the "SSL-TLS Certificate Based Client Authentication" option in Step 3 of the SAML Token Profile Wizard.
Step 7: Subject Confirmation (cont'd)	<p>If SAML version 2.0 is permitted, complete the Subject Confirmation Data fields:</p> <ul style="list-style-type: none"> • Recipient: This property allows the expected recipient to be configured. You may enter the name directly or enter a String context variables. Leave this field blank to allow any recipient.  • Check Address: Select this check box to validate the 'Address' attribute. Currently, the Gateway only supports IPv4 addresses. Note: Address validation is meaningful only when a transport with a client IP address is used. For example, this setting is not compatible with messages routed via JMS. • Check Validity Period: Select this check box to check the time period validity period in the request. The permissible clock skew for validation is defined by the cluster properties <i>samlAssertion.validate.notBeforeOffsetMin</i> and <i>samlAssertion.validate.notOnOrAfterOffsetMin</i>. Note: If there are no validity period constraints in the request message, then there is nothing to check and validation (of the time period constraints) will always succeed.
Step 8: Name Identifier	<p>Specify the name formats that are acceptable to the Gateway; optionally enter a subject name qualifier:</p> <ul style="list-style-type: none"> • Name Qualifier: Optionally enter a subject name identifier (for example, "www.example.com"). You may reference context variables.  • Format: Select one or more subject name formats that should be accepted by the Gateway. Select the Unspecified check box if the subject name format is not known. This will cause the Gateway to attempt to match the subject name identifier specified in the Name Qualifier field against the user login property. If the Name Qualifier field is blank, then the Gateway will not verify the Name Qualifier attribute value. <p>You can only select name formats applicable to the SAML version chosen in Step 2 of the wizard.</p>
Step 9: Conditions	<p>In this step, you can optionally specify any conditions to be observed.</p> <ul style="list-style-type: none"> • Check Assertion Validity Period: Select this check box to verify that the SAML token is still within its validity period, using the current Gateway time. Clear this check box to not check the validity period within the token.

Wizard Step	Description
	<ul style="list-style-type: none"> Maximum Expiry Time: Specify the maximum allowable expiry time period for the SAML token. The Gateway will use the earlier of the expiry date or the specified period. This allows you to restrict the token's expiry date with an earlier date. (If the specified date is later than the token's expiry date, then the token's date takes priority.) Tokens that exceed the expiry time will cause policy consumption to fail and audit message code 6108 will be logged. The default is 0 (zero), which indicates that token expiration is not checked. The maximum allowable expiry time is 100 years. Audience Restriction: Optionally enter an audience restriction constraint into the field. You may reference context variables. 

Require SSH Credentials Assertion

The *Require SSH Credentials* assertion allows you to require a user's SSH credentials in a request. You can require either the user name and plain text password only, or the user name and public key only, or the user name and either the plain text password or the public key.

This assertion is a credential source that saves the user name with the password or public key from the SSH session for later authentication and authorization using the "Authenticate User or Group Assertion" on page 170.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the **Require SSH Credentials Properties** automatically appear; when modifying the assertion, right-click **Require SSH Credentials** in the policy window and select **Require SSH Credentials Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

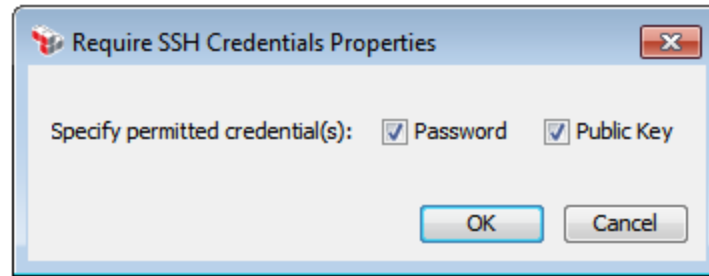


Figure 72: Require SSH Credentials Properties

3. Specify the credentials that are required in the request: **Password**, **Public Key**, or both.
4. Click **[OK]** when done.

Require SSL or TLS Transport Assertion

The *Require SSL or TLS Transport* / *Require SSL or TLS Transport with Client Authentication* assertion allows you to specify the SSL or TLS requirement to ensure transport-level confidentiality and integrity. You can specify whether an SSL/TLS connection is required, optional, or forbidden.

You can optionally require client certificate authentication and can control whether to check the validity period of the client certificate prior to gathering credentials.

Note: When requiring client certificate authentication, the assertion will behave as a credential source that saves the client certificate from the SSL-TLS handshake for later authentication and authorization via the [Authenticate User or Group](#) assertion.

This assertion appears in two different assertion palettes:

- When accessed from the [Access Control](#) palette, this assertion is labeled "**Require SSL or TLS Transport with Client Authentication**" and has the Require Client Certificate Authentication check box selected by default.
- When access from the [Transport Layer Security](#) palette, this assertion is labeled "**Require SSL or TLS Transport**" and does not have the Require Client Certificate Authentication check box selected by default.

In either instance, you are free to toggle this check box according to your needs.

Using the Assertion

1. Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **[Require|Forbid|Optional] SSL or TLS Transport <with Client Authentication>** in the policy window and select **SSL or TLS Transport Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

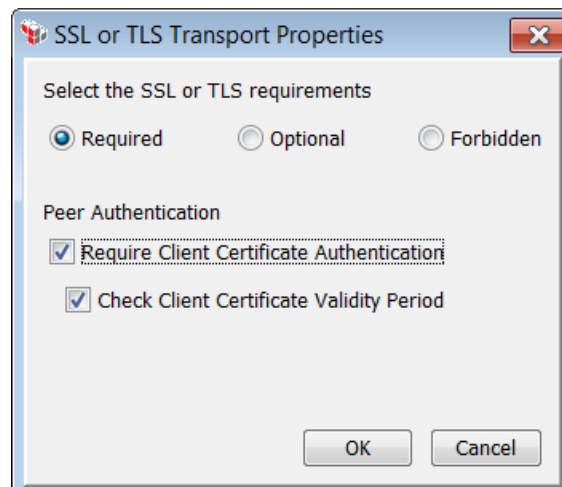


Figure 73: SSL or TLS Transport Properties

3. Configure the properties as follows:

Table 62: SSL or TLS Transport settings

Setting	Description
Select the SSL or TLS requirements	<ul style="list-style-type: none"> • Required: Select this option to disallow requests that do not arrive over an SSL-secured connection. This setting is the default setting for the assertion. When you select Required, the Peer Authentication options are enabled. • Optional: Select this option to configure the Gateway to match the behavior of the incoming request. Requests are not required to arrive over an SSL-secured connection. • Forbidden: Select this option to disallow requests that arrive over an SSL-secured connection. This setting can be used to discourage users of a free service from consuming server SSL resources without paying for an upgraded account.
Require Client Certificate Authentication	<p>Indicates whether the client certificate needs to be authenticated:</p> <ul style="list-style-type: none"> • Select this check box to gather the client certificate to be authenticated later in the policy by an authentication assertion (for example, "Authenticate User or Group Assertion" on page 170).

Setting	Description
	<p>This indicates that a client certificate is required as part of the SSL-TLS handshake. The client certificate is used to authenticate the service requestor.</p> <p>This check box is available only when "Select the SSL or TLS requirements" is set to Required.</p> <p>Note: Selecting the check box does not ensure that the client certificate will be authenticated. The Require SSL or TLS Transport with Client Authentication assertion only behaves as a credential source assertion. An authentication assertion must be present in the policy to authenticate the certificate.</p> <ul style="list-style-type: none"> • Clear this check box to not gather the client certificate. This makes the "Require SSL or TLS Transport with Client Authentication Assertion" (accessed from the Access Control palette) identical to the "Require SSL or TLS Transport Assertion" (accessed from the Transport Layer Security palette).
Check Client Certificate Validity Period	<p>Controls whether the validity period of the client certificate is checked during SSL-secured connections.</p> <ul style="list-style-type: none"> • Select this check box to check the validity period of the client certificate and not gather credentials if the certificate is expired. This option will not populate the <code>\${request.ssl.clientCertificate}</code> variable. This setting is the default. • Clear this check box to not check the client certificate validity period and gather credentials from all client certificates. This options will allow the <code>\${request.ssl.clientCertificate}</code> variable to be populated with expired certificates. <p>Notes: (1) Although expired certificate information may be gathered, such certificates cannot be used to authenticate users. For example, the Authenticate User or Group assertion will fail when an expired certificate is used. (2) Regardless of whether you check the validity period prior to gathering the credentials, validity will still be checked if an actual authentication is attempted (using the Internal Identity Provider, Federated Identity Provider, or LDAP Identity Provider).</p>

4. Click **[OK]** when done.

Require Windows Integrated Authentication Credentials Assertion

The *Require Windows Integrated Authentication Credentials* assertion requires the presence of credentials from a Windows domain in the request.

As this assertion is a credential source, ensure that there no other conflicting credential sources in the policy (for example, the [Require HTTP Basic Credentials](#) assertion).

The Require Windows Integrated Authentication Credentials assertion places the realm of the client (which should be an expected value for the identity provider) into the *kerberos.realm* context variable. This enables policy decisions based on this aspect of the client credential and is useful in situations where the client can be from multiple domains/realms. For example:

EAST.MYCOMPANY.COM
WEST.MYCOMPANY.COM

The realm is displayed when using the [Manage Kerberos Configuration](#) task. Ensure that the realm has been validated by this task before an Kerberos authentication is attempted.

This assertion supports both the Kerberos and NTLM protocols for Windows Integrated Authentication. To allow a service policy to automatically handle both protocols, you should structure your policy so that both the Require Windows Integrated Authentication Credentials and [Require NTLM Authentication Credentials](#) assertions are present in the policy (in that order):

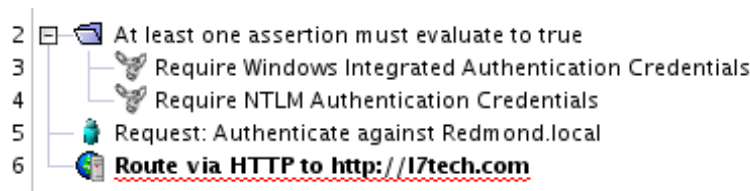


Figure 74: Policy to handle both Kerberos and NTLM protocols

Notes: (1) The policy fragment in Figure 74 does not support delegated credentials use case. It is intended to support authentication of the user credentials using available authentication assertions only. (2) Before using the Require Windows Integrated Authentication Credentials assertion, be sure the *Windows Domain Login Configuration* in the *Layer 7 Installation and Maintenance Manual* has been performed.

Using the Assertion

1. Add the assertion to the policy development window. For more information, see "Adding an Assertion" on page 112.

The assertion is added to the policy window; no further configuration is required.

2. Move the assertion to the place in the policy list where enforcement should occur.
3. Optionally use the [Authenticate User or Group](#) assertion to limit access to specific users from an LDAP Identity Provider. (**Note:** If you do this, be sure the Active Directory server has been configured as an LDAP provider.)

Require WS-Secure Conversation Assertion

The *Require WS-Secure Conversation* assertion allows you to require that request and response messages be secured using a secure conversation session. Specifically, messages must:

- Include a "SecurityContextToken" referencing an already-established WS-Secure Conversation session
- Include at least one element signed with the shared secret from this session as proof of possession of the session shared secret

The Require WS-Secure Conversation assertion is a credential source that saves the user that owns the session for later authorization via the "Authenticate User or Group Assertion" on page 170. This assertion can be used in tandem with the [Protect Against Message Replay](#), [Sign Element](#), and [Encrypt Element](#) assertions.

Some more information about using WS-Secure Conversation on the Gateway:

- The Require WS-Secure Conversation assertion, by itself, does not require that the request message contain a timestamp, and does not check the validity of any time stamp that might be present. To protect against stale or replayed messages, use the Require WS-Secure Conversation assertion with the "Protect Against Message Replay Assertion" on page 680.
- This assertion may behave unexpectedly if there are two users in different identity providers, with both recognizing the same certificate credentials.
- To enable persistence for WS-Secure Conversation sessions, set the cluster property `wss.secureConversation.clusterSessions` to "true". This will allow WSSC sessions to be shared between cluster nodes.

- Federated virtual users are not compatible with secure conversation. For more information on virtual users, see Federated Identity Provider Users and Groups in the *Layer 7 Policy Manager User Manual*.

Context Variable Created by This Assertion

When the Require WS-Secure Conversation assertion is used, it creates the following context variable that contains the secure conversation context in the inbound request message:

inboundSC.session

To access the session ID, use `${inboundSC.session.id}`.

Using the Assertion

- Add the assertion to the policy development window as described in [Adding an Assertion](#).

The assertion is added to the policy window; no further configuration is required.

Require WS-Security Kerberos Token Profile Credentials Assertion

The *Require WS-Security Kerberos Token Profile Credentials* assertion requires that the request message contains a valid WSS1.1 Kerberos Token (specifically, a GSS wrapped Kerberos v5 AP-REQ, as defined in the GSSAPI specification).

This assertion places the realm of the client in the *kerberos.realm* context variable. This enables policy decisions based on this aspect of the client credential and is useful in situations where the client can be from multiple domains/realm. For example:

EAST.MYCOMPANY.COM
WEST.MYCOMPANY.COM

For more information on the Kerberos specification, see <http://docs.oasis-open.org/wss/v1.1/>. From there, you can download the *wss-v1.1-spec-pr-KerberosTokenProfile-01* document in either HTML or PDF format.

Note: When authenticating users with Kerberos, the realm must be validated before authentication is performed. Ensure that the *kerberos.realm* context variable is an expected value for the identity provider.

Note: The Gateway must be correctly configured to use the Require WS-Security Kerberos Token Profile Credentials assertion. For more information, see *Windows Domain Login Configuration* in the *Layer 7 Installation and Maintenance Manual* and *Creating Trusted Gateway Accounts* in the *Securespan XML VPN Client* documentation.

Using the Assertion

1. Add the assertion to the policy development window. For more information, see "Adding an Assertion" on page 112.

The assertion is added to the policy window; no further configuration is required.

2. Move the assertion to the place in the policy list where Kerberos authentication should occur.
3. Optionally use the [Authenticate User or Group](#) assertion to provide access to the LDAP Identity Provider. (**Note:** If you do this, be sure the Active Directory server has been configured as an LDAP provider.)

Require WS-Security Password Digest Credentials Assertion

The *Require WS-Security Password Digest Credentials* assertion allows you to require that a WSS Digest token is present with a matching username and password. You can optionally check whether a timestamp or nonce is present, but this assertion does not confirm whether the timestamp has expired nor does it enforce that the nonce is not reused.

This assertion will succeed if the processed security header of the target message contains at least one WSS Digest Token with a matching username and password.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.

- Right-click **<target>: Require WS-Security Password Credentials** in the policy window and select **Require WS-Security Signature Properties** or double-click the assertion in the policy window.

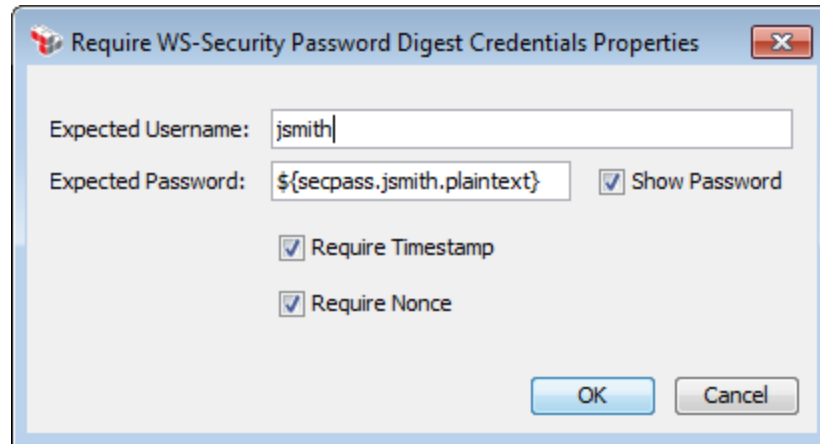





Figure 75: Require WS-Security Password Digest Credentials Properties

- Configure the properties as follows:

Table 63: WS-Security Signature settings

Setting	Description
Expected Username	Specify the expected username in the WSS Digest Token. You may reference context variables. 
Expected Password	Enter the expected password. You may type a plaintext password, however it is highly recommended that you reference the password using the <code>\${secpass.*.plaintext}</code> context variable instead. 
Show Password	Select this check box to display the password as it is being typed. Clear this check box to obfuscate the typed password.
Require Timestamp	Select this check box to require that a timestamp be present. Note: This assertion does not confirm whether the timestamp has expired.
Require Nonce	Select this check box to require that a nonce is present. Note: This assertion does not enforce whether the nonce is not reused.

- Click **[OK]** when done. 

Require WS-Security Signature Credentials Assertion

The *Require WS-Security Signature Credentials* assertion allows you to require that the web service or XML application target message:

- Includes an X.509 BinarySecurityToken containing a client certificate
- Has at least one element signed by that client certificate's private key as a proof of possession of the private key for the client certificate.

The Require WS-Security Signature Credentials assertion is a credential source that saves the certificate from the X.509 BinarySecurityToken for later authorization via the [Authenticate User or Group](#) or [Authenticate Against Identity Provider](#) assertions. This assertion can be used in tandem with the [Protect Against Message Replay](#), [Sign Element](#), and [Encrypt Element](#) assertions.

The Require WS-Security Signature Credentials assertion supports version 1.0 of the WS-Security standard. The Gateway creates and uses X.509 v3 certificates.

W A R N I N G

The Require WS-Security Signature Credentials assertion, by itself, does not require that the request message contain a timestamp, and does not check the validity of any timestamp that might be present. To protect against stale or replayed messages, use the Require WS-Security Signature Credentials assertion with the "Protect Against Message Replay Assertion" on page 680.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Require WS-Security Signature Credentials** in the policy window and select **WS-Security Signature Properties** or double-click the assertion in the policy window.

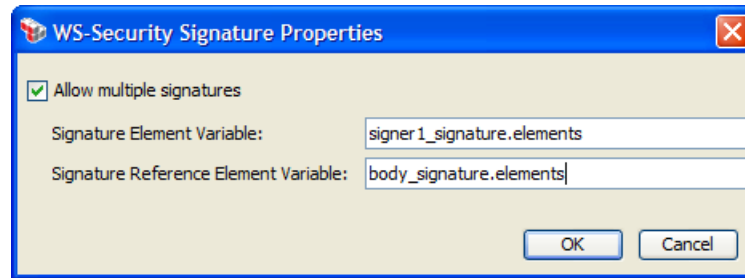




Figure 76: WS-Security Signature Properties

3. Configure the properties as follows:

Table 64: WS-Security Signature settings

Setting	Description
Allow multiple signatures	<p>Select this check box to permit multiple signatures in a policy.</p> <p>Clear the check box to disallow multiple signatures. If this check box is not selected and multiple signatures are present, then the assertion will fail.</p> <p>For more information, see "Working with Multiple Signatures" on page 17.</p>
Signature Element Variable 	<p>To pick a particular signature to use with an authentication, ensure that an XPath assertion (for example, Sign Element, Encrypt Element) has been used to set a context variable to restrict the processed signatures. Then enter the context variable in the Signature Element Variable field.</p> <p>The following is a simple example of an XPath expression containing signature information in the header:</p> <pre>/soapenv:Envelope/soapenv:Header/wsse:Security/ds:Signature[1]</pre> <p>Note: The ".element" variable is not compatible with the Require WS-Security Signature Credentials assertion; use the ".elements" variable instead. For more information about the XPath context variables, see the Evaluate Request XPath and Evaluate Response XPath assertions.</p>
Signature Reference Element Variable 	<p>Enter a context variable that will be used to select the signature by (one or more) elements that it signs. This variable may be used in addition to the Signature Element Variable.</p> <p>The Signature Element Variable identifies the set of acceptable signatures (which is all signatures in the message if the variable is not set). The Signature Reference Element Variable further restricts that set of signatures to ones that have signed the desired elements (if the variable is set previously using XPath assertions).</p> <p>Note: Specifying a Signature Reference Element Variable is not validating the signature reference—it is only for signature selection. The Require Signed Element assertion is still required to verify that the correct message parts are signed.</p>

4. Click **[OK]** when done.

Require WS-Security UsernameToken Profile Credentials Assertion

The *Require WS-Security UsernameToken Profile Credentials* assertion allows you to require basic WS-Security UsernameToken authentication of user name, plain text password, and the authentication realm in a section of the XML message for the web service or XML application. This assertion is a credential source that saves the user name and password from the WSS UsernameToken for later authentication and authorization via the "Authenticate User or Group Assertion" on page 170.

This assertion supports version 1.0 of the WS-Security standard. The Gateway creates and uses X.509 v3 certificates.

Tip: Since the *Require WS-Security UsernameToken Profile Credentials* assertion requires a plain text password to be inserted into the service message, you should also select the [Require SSL or TLS Transport](#) assertion when using basic WS Token authentication.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Using the Assertion

- Add the assertion to the policy development window as described in "Adding an Assertion" on page 112.

The assertion is added to the policy window; no further configuration is required.

Require XPath Credentials Assertion

The *Require XPath Credentials* assertion looks for a login (user name) and password in the current request using a pair of XPath expressions. If the target credentials are found in the message, then the Gateway sets the current request's credentials using the contents of the elements described by the XPath expressions and optionally removes the original elements from the request.

Note: In order to use the *Require XPath Credentials* assertion, both a user name and password must be configured for the identity or identities in the policy. For more information, see the "Authenticate User or Group Assertion" on page 170.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the **XPath Credentials Properties** automatically appear; when modifying the assertion, right-click **<target>: Require XPath Credentials...** in the policy window and select **XPath Credentials Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

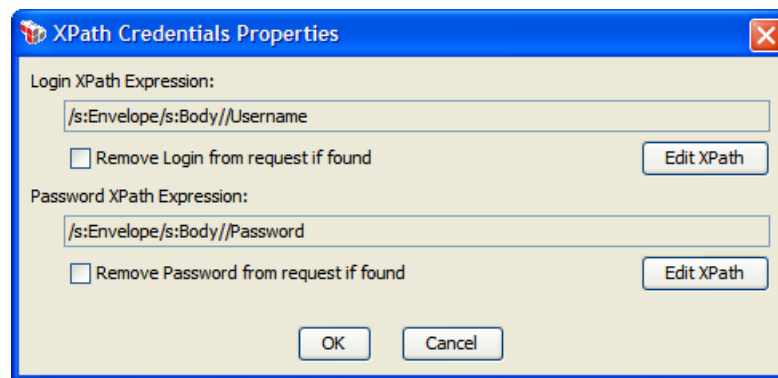




Figure 77: XPath Credentials Properties

- Configure the properties as follows:

Table 65: XPath Credentials settings

Setting	Description
Login XPath Expression 	<p>The XPath 1.0 expression that will locate the element containing the login ID or user name. The default expressions is:</p> <p><code>/s:Envelope/s:Body//Username</code> (SOAP) <code>//Username</code> (non-SOAP)</p> <p>Click [Edit XPath] if you need to select another expression. For more information, see "Selecting an XPath" on page 154.</p> <p>Tip: Before constructing XPath expressions for the login/user name and password elements, consult the service's namespace map to view and choose the appropriate namespace prefixes. To access the map, click [Edit Namespaces] while selecting an XPath to see the default</p>

Setting	Description
	namespaces and prefixes.
Remove Login from request if found	<p>Select this check box to have the Gateway remove the element containing the login/user name value from the request message. Use this option when credentials must be authenticated by the Gateway but not communicated to the protected service.</p> <p>Tip: A request message will never expose login information if the message uses context variables to hold a user's credentials. Thus, login information will not be exposed regardless of whether the Remove Login from request if found check box is selected.</p>
Password XPath Expression 	<p>The XPath 1.0 expression that will locate the element containing the password. The default is:</p> <p><code>/s:Envelope/s:Body//Password</code> (SOAP)</p> <p><code>//Password</code> (non-SOAP)</p> <p>Click [Edit XPath] to construct this expression. For more information, see "Selecting an XPath" on page 154.</p>
Remove Password from request if found	<p>Select this check box to have the Gateway remove the element containing the password value from the request message, but save the credentials in memory.</p>

Tip: For greater flexibility, you may reference context variables in an XPath expression. For more information, see Context Variables for XPaths in the *Layer 7 Policy Manager User Manual*.

- If necessary, click **[Namespaces]** to edit the [namespace map](#).
- Click **[OK]** when done.

Retrieve Credentials from Context Variable Assertion

Using the *Retrieve Credentials from Context Variable* assertion, you can use an X.509 certificate contained in a specified context variable as if had arrived as X.509 credentials (for example, via an SSL client certificate or from a WS-Security signature). These credentials can then be used for authentication purposes in the [Authenticate User or Group](#) or [Authenticate Against Identity Provider](#) assertions.

This assertion could be used with the context variables created by the "(Non-SOAP) Verify XML Element Assertion" on page 391.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

W A R N I N G

Only use certificate credentials from an entity that has proven that it possesses the corresponding private key (for example, from a digital signature or a TLS client certificate). Do not use certificates from unverified sources.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>**: **Retrieve Credentials from Context Variable: <variable>** in the policy window and select **Credentials from Context Variable Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

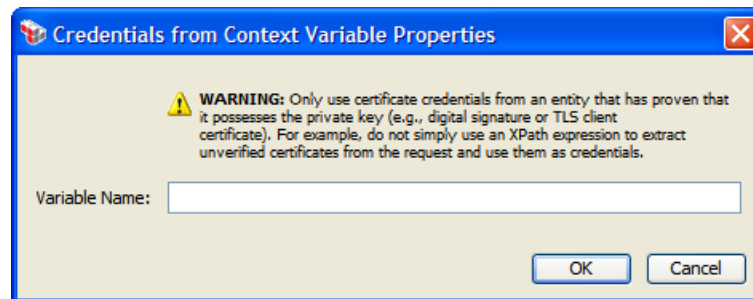


Figure 78: Credentials from Context Variable Properties

3. Enter a context variable containing an X.509 certificate. The Gateway will use this certificate as though it had arrived from conventional credential sources.
4. Click [OK].

Retrieve Kerberos Authentication Credentials Assertion

The *Retrieve Kerberos Authentication Credentials* assertion enables the Gateway to support the following extensions to the Kerberos Protocol:

- **Protocol Transition (S4U2Self):** This is used for clients that require access to Active Directory resources, but are unable to acquire a Kerberos token. The

Protocol Transition delegation method allows the Gateway to request a Kerberos service ticket on behalf of the client. The client may be using any other authentication methods, such as: basic, or certificate-based authentication, or SAML tokens.

- **Constrained Delegation (S4U2Proxy):** Using this method, the client sends a service ticket that will be presented along with the server's TGT (Ticket Granting Ticket) to Active Directory. This is used to request a service ticket using constrained delegation to another service. Only services in a previously configured list can receive a service ticket. Constrained delegation ensures that only authorized authenticated servers are permitted to perform constrained delegation to the next server.

For a summary of the configuration required to support each delegation method, see ["Using the Protocol Transition Delegation Method"](#) and ["Using the Constrained Proxy Delegation Method"](#) below.

Prerequisites:

- Kerberos and the Active Directory should be configured and operational
- A valid keytab file has been generated on the Active Directory server
- CA API Gateway must have a valid keytab uploaded, if keytab-based authentication is used
- Java 7 is required if using the browser client version of the Policy Manager

Using the Protocol Transition Delegation Method

The following is a summary of the workflow for using the "Protocol Transition" delegation method. **Note:** These instructions assume familiarity with Active Directory. If you require assistance, please contact your AD administrator.

1. Configure the Gateway account on the Active Directory:
 - a. Log in to the Active Directory and open the properties for the Gateway account.
 - b. In the Properties dialog: access [Delegation] tab > choose **Trust this user for delegation to specified services only** > choose **Use any authentication protocol** below it.
 - c. Click **[Add]** below the list > click **[Users or Computers]** in the Add Services dialog > click **[Advanced]** in the Select Users or Computers dialog.

- d. Click [**Find Now**] in the Select Users or Computers dialog and then select the server that requires Kerberos authentication.
 - e. Click [**OK**] to dismiss the dialog boxes until you return to the Add Services dialog.
 - f. Select the "**http**" service and then click [**OK**].
 - g. Click [**OK**] to close the Gateway account Properties dialog.
2. Create a new service and construct a policy that includes:
 - One or more credential source assertions (for example, [Require HTTP Basic Credentials](#))
 - An identity assertion (for example, [Authenticate Against Identity Provider](#))
 - Retrieve Kerberos Authentication Credentials assertion
 - [Route via HTTP\(S\)](#) assertion
 3. Configure the Retrieve Kerberos Authentication Credentials assertion as follows:
 - **Realm:** Realm of service
 - **Target SPN:** Service Principal Name of the destination service protected by Kerberos. The Kerberos ticket obtained by the Gateway from KDC is passed to that service.
 - **Gateway Credentials:** Can be either option. If Gateway credentials is "Use Gateway Keytab", then the credentials stored in the keytab file will be used, otherwise you must provide credentials in the assertions.
 - **Delegation Method:** Choose **Protocol Transition**
 - **Authenticated User:** Use either last authenticated user or provide an authenticated user name and user realm.
 - **User Realm:** The realm of the user. If the user's realm differs from the service realm, the Gateway automatically performs a Kerberos cross-realm referral authentication. For more information about this process, refer to: <http://msdn.microsoft.com/en-us/library/cc246109.aspx>.
 4. Configure the Route via HTTP(S) assertion:
 - In the [Target] tab, set the route URL.
 - In the [Security] tab, choose the Service Authentication method **Use Windows Integrated** and then choose **Use Delegated Credentials**.

5. Call the service from a client that is not a part of the authenticating domain or does not have a trusted relationship with the domain.

Using the Constrained Proxy Delegation Method

The following is a summary of the workflow for using the "Constrained Proxy" delegation method. **Note:** These instructions assume familiarity with Active Directory. If you require assistance, please contact your AD administrator.

1. Configure the Gateway account on the Active Directory:
 - a. Log in to the Active Directory and open the properties for the Gateway account.
 - b. In the Properties dialog: access [Delegation] tab > choose **Trust this user for delegation to specified services only** > choose **Use Kerberos only** below it.
 - c. Click **[Add]** below the list > click **[Users or Computers]** in the Add Services dialog > click **[Advanced]** in the Select Users or Computers dialog.
 - d. Click **[Find Now]** in the Select Users or Computers dialog and then select the server that requires Kerberos authentication.
 - e. Click **[OK]** to dismiss the dialog boxes until you return to the Add Services dialog.
 - f. Select the "**http**" service and then click **[OK]**.
 - g. Click **[OK]** to close the Gateway account Properties dialog.
2. Create a new service and construct a policy that includes the following assertions:
 - [Require Windows Integrated Authentication Credentials](#)
 - Retrieve Kerberos Authentication Credentials
 - [Route via HTTP\(S\)](#)
3. Configure the Retrieve Kerberos Authentication Credentials assertion as follows:
 - **Realm:** Realm of authenticated user (provided in the assertion; Gateway will not locate the realm from the KDC)
 - **Target SPN:** Service Principal Name of the destination service protected by Kerberos. The Kerberos ticket obtained by the Gateway from KDC is passed to that service.

- **Gateway Credentials:** Can be either option. If Gateway credentials is "Use Gateway Keytab", then the credentials stored in the keytab file will be used, otherwise you must provide credentials in the assertions.
 - **Delegation Method:** Choose **Constrained Proxy**
4. Configure the Route via HTTP(S) assertion:
 - In the [Target] tab, set the route URL.
 - In the [Security] tab, choose the Service Authentication method **Use Windows Integrated** and then choose **Use Delegated Credentials**.
 5. Call the service from the client that is logged to the authenticating domain or has trusted relationship with the domain.

Kerberos Service Ticket/Session Caching

The Gateway implements Kerberos referral/credentials ticket caching to minimize the number of requests sent to the KDC (Key Distribution Center) and improve transaction response time. The entire referral chain is stored in the cache, as well as the session key. These are reused later when generating a new service ticket. The following cluster properties can be used to configure the cache:

- *kerberos.cache.size*: Sets the maximum size of the cache.
- *kerberos.cache.timeToLive*: Limits the maximum time the Kerberos tickets are store. If any ticket in the chain expire before the maximum period is reached, the entire chain is discarded and the Gateway will request new referral tickets and session keys from the KDC again.

For more information about these and other Kerberos-related cluster properties, see "Kerberos Cluster Properties" in the *Layer 7 Policy Manager User Manual*.

Note the following limitations to the Kerberos caching:

- Cached data is not persisted to the data source.
- Cached data is not synchronized to all cluster node.

Using the Assertion


1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.

- When adding the assertion, the **Kerberos Authentication Credentials Properties** dialog automatically appears; when modifying the assertion, right-click **Retrieve Kerberos Authentication Credentials** in the policy window and select **Kerberos Authentication Credentials Properties** or double-click the assertion in the policy window. The assertion properties are displayed.


Figure 79: Kerberos Authentication Credential Properties

- Configure the properties as follows:

Table 66: Kerberos Authentication Credentials settings

Setting	Description
KDC Settings 	<p>Enter the following information about the KDC. You may reference context variables in either field.</p> <ul style="list-style-type: none"> Realm: Enter the location of the KDC. Target SPN: Enter the routing destination service principal name. <p>The Target SPN will be used with the Realm to look up the service principal name from the keytab file, if a multiple principal keytab file is provided. For more information on multiple</p>

Setting	Description
	principal keytab files, see "Using Windows Domain Login" in the <i>Layer 7 Installation and Maintenance Manual</i> .
Gateway Credentials	<p>Specify the credentials for the KDC that will be used to authenticate the Gateway in order to obtain a TGT (Ticket Granting Ticket) on the client's behalf. Choose from the following:</p> <ul style="list-style-type: none"> • Use Gateway Keytab: Use the credentials from the keytab that was uploaded to the Gateway. For more information, see Managing Kerberos Configuration in the <i>Layer 7 Policy Manager User Manual</i>. • Use Configured Credentials: Specify the credentials to use in the following fields: <ul style="list-style-type: none"> • Name: Enter the username. • Password: From the drop-down list, select the Password to use to log in. If the password you require is not listed, click [Managed Stored Passwords] to add it to the list of store passwords. For more information, see Managing Stored Passwords in the <i>Layer 7 Policy Manager User Manual</i>. <p>Tip: You cannot type the password directly here; it must be defined in the Gateway's secure password storage.</p>
Delegation Method	<p>Choose the delegation method to use:</p> <ul style="list-style-type: none"> • Protocol Transition: Choose this option to use the user login credentials from the policy enforcement context to request a Kerberos service ticket from KDC (Key Distribution Center) for the Gateway on behalf of the authenticated user. This ticket will be passed to the destination service protected by Kerberos via the routing assertion. The Gateway account must be configured to enable delegation to specified services only using any authentication protocol. <p>Note: To use this method, the user must have been authenticated via one of the credential source assertions such as Require HTTP Basic Credentials.</p> • Constrained Proxy (Kerberos Only): Choose this option if the client forwarded Kerberos service ticket to the Gateway to act on behalf of the client when the Gateway has limited access to the services protected by Kerberos. The Gateway will present this ticket to KDC in exchange to a new ticket for the destination service. The Gateway account must be configured to enable delegation to specified services only using Kerberos authentication protocol. <p>Note: Currently, only the Route via HTTP(S) assertion supports Kerberos constrained delegation.</p>
Authenticated User	When the delegation method is "Protocol Transition", identify the user

Setting	Description
 ("Protocol Transition" delegation method only)	<p>who will be acquiring the Kerberos ticket. You may reference context variables.</p> <p>This panel is disabled when delegation method is "Constrained Proxy".</p> <ul style="list-style-type: none"> • Last Authenticated User: Use the most recently authenticated user. • Specify User Name: Specify any user specified in the User CN text field. You can enter any of the following: <ul style="list-style-type: none"> • a user CN name • a context variable that contains the user CN name • either of the predefined context variables: <code>\${request.authenticateduser}</code> or <code>\${request.authenticatedusers[<index>]}</code> • User Realm: Enter the realm of the authenticated user. If left blank, this assertion will use the service realm as the user realm. <p>Tip: When the user's realm differs from the service realm, the Gateway automatically performs a Kerberos cross-realm referral authentication, obtaining the necessary referral ticket (s) in the background. For more information, refer to: http://msdn.microsoft.com/en-us/library/cc246109.aspx.</p>

4. Click **[OK]** when done.

Retrieve SAML Browser Artifact Assertion

The *Retrieve SAML Browser Artifact* assertion uses the credentials in a request message to obtain a SAML Browser Artifact from a SAML Single Sign-On (SSO) endpoint. The SSO endpoint authenticates a requestor using either Basic Authentication or HTML Form POST Authentication. If authentication succeeds, the Gateway parses the redirect header and saves the "SAMLart" parameter in memory for future assertions in the same policy to use.

The Retrieve SAML Browser Artifact assertion is useful for "mixed-mode" SAML interactions in which an initial request containing a user's credentials establishes a SSO session that can be used in subsequent browser-based requests from the same user. Multiple instances of this assertion can be used in a policy if required.

The saved SAML artifact value can be used in the [Evaluate Regular Expression](#) assertion by entering the variable "\${samlBrowserArtifact.artifact}" in the Replacement field in the Evaluate Regular Expression Properties. This is useful when resources require different SAMLart parameters.

Note: The Retrieve SAML Browser Artifact assertion should be placed after the credential source assertion (such as the [Require HTTP Basic Credentials](#) assertion) and before the assertion that uses the obtained context parameters (such as the [Evaluate Regular Expression](#) assertion).

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **SAML Browser Artifact Properties** automatically appear; when modifying the assertion, right-click **Retrieve SAML Browser Artifact** in the policy window and select **SAML Browser Artifact Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

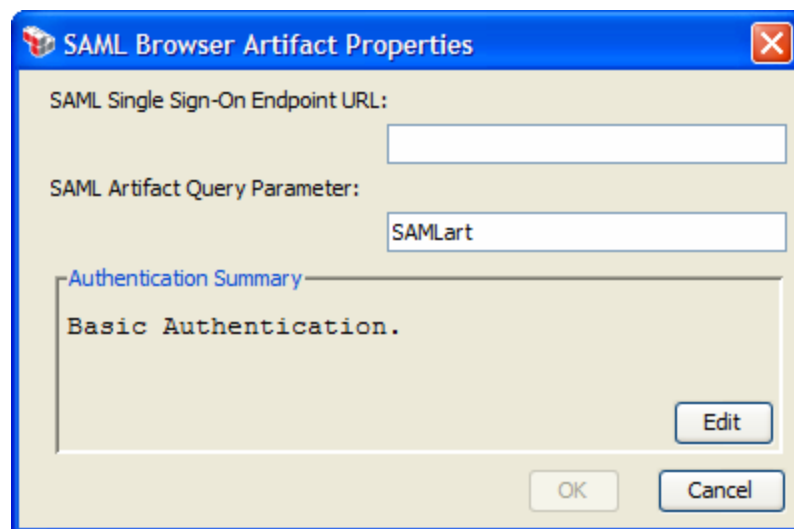


Figure 80: SAML Browser Artifact Properties

3. Configure the properties as follows:

Table 67: SAML Browser Artifact settings

Setting	Description
SAML Single Sign-On Endpoint URL	<p>Enter the URL of the SAML identity provider endpoint. This URL must include both the Single Sign-On (SSO) endpoint and the corresponding service endpoint.</p> <p>If the SSO system returns a '302' status code after processing, then the assertion succeeds and will proceed to process the service endpoint</p>

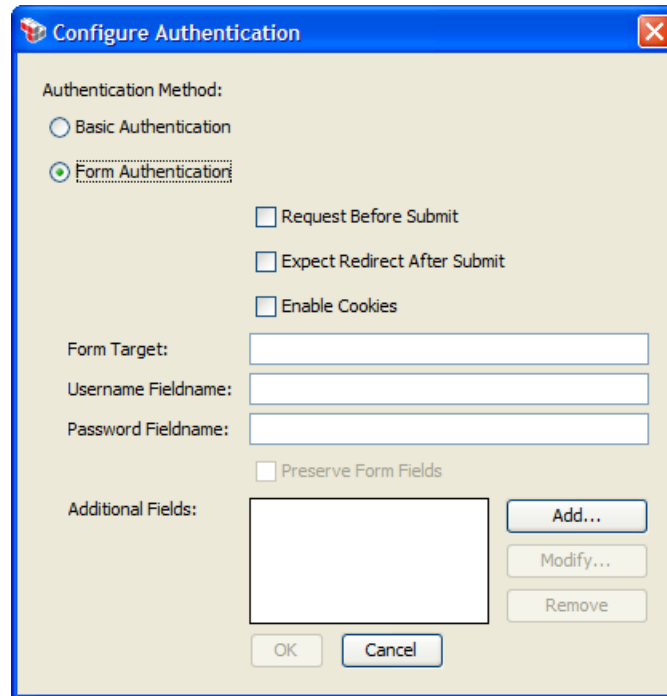
Setting	Description
	<p>URL. If the SSO system returns a non-302 status code, the assertion will fail, possibly resulting in a failure of the policy if:</p> <ul style="list-style-type: none"> the target redirect URL cannot be parsed, or if the result does not match the SAML Artifact Query Parameter specified below, <p>Note: The URL specified here is only for authentication, not message routing. Even if the URL is the same as the endpoint of the service, the Route via HTTP(S) assertion is still required to route service messages.</p>
SAML Artifact Query Parameter	<p>This field is populated with the default value "SAMLart". SAMLart is the type of cookie returned from the SSO system, which is then sent on to the target URL. If the return cookie is not the same type as the value in the SAML Artifact Query Parameter, then the assertion will fail.</p> <p>The SAMLart setting should suffice for most usage scenarios of the Retrieve SAML Browser Artifact assertion. You should change the default value only if the administrator of the Single Sign-On system has chosen a different parameter name. The value in the SAML Artifact Query Parameter field is case sensitive.</p>

- Examine the information in the **Authentication Summary** box. There are two types of authentication methods:

Table 68: Authentication methods

Method	Description
Basic Authentication	<p>The Basic Authentication method uses a Require HTTP Basic Credentials, Require WS-Security UsernameToken Profile Credentials, or Retrieve XPath Credentials assertion to extract credentials from an incoming request message. Credentials are passed to the Single Sign-On (SSO) endpoint in an HTTP message header.</p> <p>This is the default authentication method.</p>
Form Authentication	<p>Like Basic Authentication, Form Authentication uses a credential source assertion to extract credentials from an incoming request message, but uses an HTML form to pass the credentials to the SSO endpoint. Form parameters can be auto-detected or manually configured.</p>

- Do one of the following:
 - To use the default Basic Authentication, click **[OK]**. The assertion is added to the policy development window.
 - To change to Form Authentication, click **[Edit]**. The Configure Authentication dialog appears.



The image shows a 'Configure Authentication' dialog box. It has a title bar with a close button. Inside, there's a section 'Authentication Method:' with two radio buttons: 'Basic Authentication' and 'Form Authentication'. 'Form Authentication' is selected. Below this are three checkboxes: 'Request Before Submit', 'Expect Redirect After Submit', and 'Enable Cookies'. Then, there are three text input fields labeled 'Form Target:', 'Username Fieldname:', and 'Password Fieldname:'. Below these is a checkbox 'Preserve Form Fields'. At the bottom left is a list box labeled 'Additional Fields:'. To its right are three buttons: 'Add...', 'Modify...', and 'Remove'. At the very bottom are 'OK' and 'Cancel' buttons.

Figure 81: Configure Authentication

6. Select **Form Authentication** and then choose a method to enter form parameters:

Table 69: Authentication methods

Method	Description
Automatically detect form parameters	<p>To automatically detect the user name and password in the HTML form from the SSO endpoint:</p> <ol style="list-style-type: none"> 1. Select the [Request Before Submit] check box. 2. If the HTML form contents are dynamic, also select the Preserve Form Field check box to extract and include the parameters in the HTML page in the Single Sign-On (SSO) endpoint authentication form. 3. Click [OK] to enter the form authentication details.
Manually configure form parameters	<p>Manually configuring form parameters improves the performance of the Retrieve SAML Browser Artifact assertion. When parameters are fully specified, there is no need to parse the authentication HTML form from the Single Sign-On endpoint and, in some cases, the HTML form may not need to be requested, optimizing the authentication workflow.</p> <p>To manually configure the form parameters:</p> <ol style="list-style-type: none"> 1. To request the SSO endpoint HTML page containing the login form, select the Request Before Submit check box. If the HTML form contents are dynamic, also select the Preserve Form Field check box to extract and include the parameters

Method	Description
	<p>in the HTML page in the Single Sign-On (SSO) endpoint authentication form.</p> <ol style="list-style-type: none"> If the initial redirect after the form POST is a re-direction to a resource (as opposed to the redirect with the SAML Artifact), check the Enable Redirect After Submit check box. To store the SAML Artifact authentication HTTP cookies, check the Enable Cookies check box. When selected, the SAML Artifact cookies will be re-used in future request messages with the same credentials. <p>Note: Cookies are always supported during form processing, even when the Enable Cookies check box is not selected. Any cookie sent in the initial form request (when the Request Before Submit check box is selected) will be passed back during form submission and when an initial redirect is followed (when the [Enable Redirect After Submit] check box is selected).</p> <ol style="list-style-type: none"> In the Form Target field, enter the URL to which the authentication form should be submitted. This URL must correspond with the form's action, such as "http://sso.example.com/login". In the User Name Field Name field, enter the name of the form parameter that specifies the user. For example, "User Name". In the Password Field Name field, enter the name of the form parameter that specifies the password. For example, "Password". When the Request Before Submit check box is selected in step 1 above, select the Preserve Form Field check box to extract and include the HTML form parameters in the authentication form. <p>Tip: Instead of copying the parameters from the HTML form, specify one or more additional fields for the authentication form as outlined in step 8 below. Doing so saves time by forgoing the form parsing process.</p> <ol style="list-style-type: none"> Optionally configure additional fields for the authentication form as follows: <ol style="list-style-type: none"> Click [Add] and then enter a Field Name (e.g., "Source") and Field Value (e.g., "Form") for the form parameter. Click [OK] to add the new field to the [Additional Fields] box in the Configure Authentication dialog. Repeat to configure additional fields as required. <p>To modify a field, select the field name and click [Modify]. To remove a field, select the field name and click [Remove].</p> Click [OK] to add the form authentication details to the Configure SAML Browser/Artifact dialog.

7. Click **[OK]** when done.

Use WS-Federation Credential Assertion

The *Use WS-Federation Credential* assertion submits credentials from the current request to the local ADFS Server. This assertion has two modes of operation:

- **Token Request:** A login and password authenticated token request is submitted to the local ADFS Server. On success a SAML token is added to the current request's SOAP security header.

In "Token Request" mode, the Use WS-Federation Credential assertion takes credentials gathered by a preceding credential source assertion, such as the transport-level [Require HTTP Basic Credentials](#) assertion or message-level [Require WS-Security UsernameToken Profile Credentials](#) assertion, and requests a token from the local ADFS Server. In "Token Exchange" mode, the WS-Federation Passive Credential assertion uses a SAML token from the request.

- **Token Exchange:** A SAML token authenticated token request is submitted to the local ADFS Server. On success, a SAML token is added to the current request's SOAP security header.

In "Token Exchange" mode, the WS-Federation Passive Credential assertion uses a SAML token from the request. If the token request/exchange is successful, a SAML token will replace the current request's credentials. If the message's original credentials are XML-based, then the XML element containing those credentials will be removed from the message.

For more information on configuring the Gateway to use WS-Federation credentials, see *Configuring WS-Federation Credential Exchange* in the Securespan XML VPN Client documentation.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **WS-Federation Request Properties** automatically appear; when modifying the assertion, right-click **[Obtain|Exchange|Authenticate] Credentials using WS-Federation Request to...**

in the policy window and select **WS-Federation Request Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 82: WS-Federation Request Properties

3. Configure the properties as follows:

Table 70: WS-Federation Request settings

Setting	Description
Action	From the drop-down list, select whether to perform a Token Request or Token Exchange . Refer to the introduction to this topic for the differences.
Token Service URL	Enter the complete URL of the WS-Federation server. The server must be running and configured to accept requests containing the values configured below.
Reply URL	Optionally enter the address of the federated service.
Realm	Enter the SOAP payload namespace URI of the requesting realm. This should match the Realm entered for the Gateway account. The Realm is only for token request actions.
Authenticate with service	If the protected service requires authentication, select this check box to have the Gateway authenticate with the protected service.

Setting	Description
Context	The context information that should be passed in with the request.
Include freshness timestamp	Select this check box to include a timestamp. The timestamp is available only for token request actions.

4. Click **[OK]** when done.

Chapter 5: Transport Layer Security Assertions

Note: This category may also include custom-created encapsulated assertions. For more information, see "Working with Encapsulated Assertions" on page 126.

In the Policy Manager, the following assertion is available in the Transport Layer Security (TLS) category of the [Assertions] tab:

Require SSL or TLS Transport Assertion 267

This assertion establishes the transport-level encryption requirement for a service.

Require SSL or TLS Transport Assertion

The *Require SSL or TLS Transport /Require SSL or TLS Transport with Client Authentication* assertion allows you to specify the SSL or TLS requirement to ensure transport-level confidentiality and integrity. You can specify whether an SSL/TLS connection is required, optional, or forbidden.

You can optionally require client certificate authentication and can control whether to check the validity period of the client certificate prior to gathering credentials.

Note: When requiring client certificate authentication, the assertion will behave as a credential source that saves the client certificate from the SSL-TLS handshake for later authentication and authorization via the [Authenticate User or Group](#) assertion.

This assertion appears in two different assertion palettes:

- When accessed from the [Access Control](#) palette, this assertion is labeled "**Require SSL or TLS Transport with Client Authentication**" and has the Require Client Certificate Authentication check box selected by default.
- When access from the [Transport Layer Security](#) palette, this assertion is labeled "**Require SSL or TLS Transport**" and does not have the Require Client Certificate Authentication check box selected by default.

In either instance, you are free to toggle this check box according to your needs.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **[Require|Forbid|Optional] SSL or TLS Transport <with Client Authentication>** in the policy window and select **SSL or TLS Transport Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

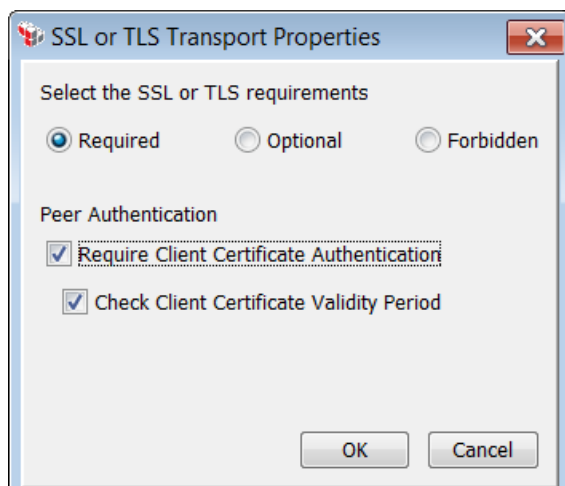


Figure 83: SSL or TLS Transport Properties

3. Configure the properties as follows:

Table 71: SSL or TLS Transport settings

Setting	Description
Select the SSL or TLS requirements	<ul style="list-style-type: none"> • Required: Select this option to disallow requests that do not arrive over an SSL-secured connection. This setting is the default setting for the assertion. When you select Required, the Peer Authentication options are enabled. • Optional: Select this option to configure the Gateway to match the behavior of the incoming request. Requests are not required to arrive over an SSL-secured connection. • Forbidden: Select this option to disallow requests that arrive over an SSL-secured connection. This setting can be used to discourage users of a free service from consuming server SSL resources without paying for an upgraded account.

Setting	Description
Require Client Certificate Authentication	<p>Indicates whether the client certificate needs to be authenticated:</p> <ul style="list-style-type: none"> Select this check box to gather the client certificate to be authenticated later in the policy by an authentication assertion (for example, "Authenticate User or Group Assertion" on page 170). <p>This indicates that a client certificate is required as part of the SSL-TLS handshake. The client certificate is used to authenticate the service requestor.</p> <p>This check box is available only when "Select the SSL or TLS requirements" is set to Required.</p> <p>Note: Selecting the check box does not ensure that the client certificate will be authenticated. The Require SSL or TLS Transport with Client Authentication assertion only behaves as a credential source assertion. An authentication assertion must be present in the policy to authenticate the certificate.</p> <ul style="list-style-type: none"> Clear this check box to not gather the client certificate. This makes the "Require SSL or TLS Transport with Client Authentication Assertion" (accessed from the Access Control palette) identical to the "Require SSL or TLS Transport Assertion" (accessed from the Transport Layer Security palette).
Check Client Certificate Validity Period	<p>Controls whether the validity period of the client certificate is checked during SSL-secured connections.</p> <ul style="list-style-type: none"> Select this check box to check the validity period of the client certificate and not gather credentials if the certificate is expired. This option will not populate the <code>\${request.ssl.clientCertificate}</code> variable. This setting is the default. Clear this check box to not check the client certificate validity period and gather credentials from all client certificates. This options will allow the <code>\${request.ssl.clientCertificate}</code> variable to be populated with expired certificates. <p>Notes: (1) Although expired certificate information may be gathered, such certificates cannot be used to authenticate users. For example, the Authenticate User or Group assertion will fail when an expired certificate is used. (2) Regardless of whether you check the validity period prior to gathering the credentials, validity will still be checked if an actual authentication is attempted (using the Internal Identity Provider, Federated Identity Provider, or LDAP Identity Provider).</p>

- Click **[OK]** when done.

Chapter 6: XML Security Assertions

Notes: (1) Depending on which Gateway product you have installed, not all the assertions shown below may be available. See Features by Product in the *Layer 7 Policy Manager User Manual* for a list of which features are available for each product. (2) This category may also include custom-created encapsulated assertions. For more information, see "Working with Encapsulated Assertions" on page 126.

In the Policy Manager, the following assertions are available in the XML Security category of the [Assertions] tab:

Add or Remove WS-Security Assertion	273
Add Security Token Assertion	277
Configuring the Private Key for SAML Assertions	277
Applying WS-Security	278
Adding a WS-S UsernameToken	279
Adding a WS-SC SecurityContextToken	281
Adding a SAML Assertion	281
Adding a WS-S EncryptedKey	282
Add Timestamp Assertion	283
Build RST SOAP Request Assertion	285
Context Variables Created by This Assertion	285
Build RSTR SOAP Response Assertion	288
Context Variables Created by This Assertion	289
Build SAML Protocol Request Assertion	291
SAML Protocol Request Wizard	292
Build SAML Protocol Response Assertion	299
Configuring the [General] Tab	301
Configuring the [Issuer] Tab (SAML 2.0 only)	305
Configuring the [Advanced] tab	306
Cancel Security Context Assertion	306
Configure WS-Security Decoration Assertion	309
Applying WS-Security	309
Configuring the [General] Tab	311
Configuring the [Signing] Tab	312
Configuring the [Encryption] Tab	313
Configuring the [Advanced] Tab	315
Create SAML Token Assertion	315
Context Variables Created by This Assertion	316
SAML Token Creation Wizard	317

Create Security Context Token Assertion	328
Context Variable Created by This Assertion	328
Create XACML Request Assertion	330
Configuring the Subject Node	332
Configuring the Resource Node	333
Configuring the Action Node	333
Configuring the Environment Node	334
Configuring the Attribute Node	334
Configuring the Multiple Attributes Node	340
Configuring the Resource Content Node	344
Encrypt Element Assertion	346
Establish Outbound Secure Conversation Assertion	348
Context Variables Created by This Assertion	349
Evaluate SAML Protocol Response Assertion	353
Evaluate XACML Policy Assertion	356
Generate OAuth Signature Base String Assertion	360
Context Variables Created by This Assertion	360
Generate Security Hash Assertion	365
Look Up Certificate Assertion	367
Look Up Outbound Secure Conversation Session Assertion	370
Context Variables Created by This Assertion	370
(Non-SOAP) Check Results from XML Verification Assertion	372
(Non-SOAP) Decrypt XML Element Assertion	374
Context Variables Created by This Assertion	374
(Non-SOAP) Encrypt XML Element Assertion	376
(Non-SOAP) Sign XML Element Assertion	377
(Non-SOAP) Validate SAML Token Assertion	380
Step 1: Introduction	381
Step 2: SAML Version	381
Step 3: SAML Statement Type	382
Step 4: Authentication Methods	383
Step 5: Authorization Statement	384
Step 6: Attribute Statement	385
Step 7: Subject Confirmation	386
Step 8: Name Identifier	389
Step 9: Conditions	390
Step 10: Embedded Signature	391
(Non-SOAP) Verify XML Element Assertion	391
Context Variables Created by This Assertion	391
Process RSTR Response Assertion	395
Context Variables Created by This Assertion	395
Protect Against Message Replay Assertion	397
Require Encrypted Element Assertion	400
Require Signed Element Assertion	402

Context Variables Created by This Assertion	402
Require Timestamp Assertion	405
Sign Element Assertion	407
Use WS-Security 1.1 Assertion	409

The XML Security assertions define the message-level encryption and signature requirements for service XML messages and enable protection against replay attacks.

You can use multiple XML Security assertions in a single policy. These assertions can only be used in a web service policy.

The default [WSS recipient](#) in the [Sign Element](#) and [Encrypt Element](#) assertions can be changed if necessary.

Note: Many of the XML Security assertions must be preceded by specific assertions. Be sure to refer to the Policy Validation Messages.

Add or Remove WS-Security Assertion

The *Add or Remove WS-Security* assertion is used to apply pending WS-Security decorations to a message or to remove security headers. You can control how to handle the WS-Security headers and the WS-Security options.

This assertion should be placed after the following WS-Security assertions in a policy if the target message is the request message or a context variable:

- [Add Security Token](#)
- [Add Timestamp](#)
- [Configure WS-Security Decoration](#)
- [Encrypt Element](#)
- [Sign Element](#)

Tip: Though it is not necessary to use the Add or Remove WS-Security assertion to apply pending decoration to the default *response* message, it will not cause harm and may be advantageous in some instances (for example, if you want to override the encryption recipient).

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

1. Do one of the following:



- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: [Apply|Clear] WS-Security** in the policy window and choose **WS-Security Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 84: WS-Security Properties

3. Configure the properties as follows:

Table 72: WS-Security Properties settings

Setting	Description
<i>WS-Security header options</i>	
Remove and recreate matching security headers (if found)	<p>Select this check box to remove the existing header before applying WS-Security, if there is an existing security header in the message that matches one of the target headers. This setting is the default behavior.</p> <p>Clear this check box to retain the existing header(s) while applying WS-Security.</p>
Remove all unmatched security headers	<p>Choose this option to remove any existing WS-Security headers that do not match any of the target headers.</p> <p>Clear this check box to retain all existing WS-Security headers.</p> <p>Tip: This option can be used to remove <i>all</i> security headers when the Apply WS-Security check box is not selected.</p>
Use MustUnderstand attribute	<p>By default, the resulting security header will have a <i>mustUnderstand</i> attribute.</p> <ul style="list-style-type: none"> Select this check box to use the recommended <i>mustUnderstand</i> attribute from the resulting WS-Security header. Clear this check box to omit the recommended <i>mustUnderstand</i> attribute from the Security header.
For the default recipient:	<p>Indicate how to handle the default recipient for the security header:</p> <ul style="list-style-type: none"> Omit actor attribute: Do not use an actor. This setting is the default. Use Layer 7 actor: Use the Layer 7 actor as the default actor.
Apply WS-Security	<p>Select this check box to apply any pending WS-Security decorations, as specified in the WS-Security options displayed below. Note: When [Apply WS-Security] is used, any decorations that were applied will be cleared automatically.</p> <p>Clear this check box to not apply pending WS-Security decorations to the header. The WS-Security options are disabled. Note: This is not the same as clearing the decorations, which is done using the [Clear WS-Security] option below.</p>
Clear WS-Security	<p>Select this check box to clear any pending WS-Security decorations and also any WS-Security decorations that would be applied automatically after the policy complete.</p> <p>Clear this check box to allow all WS-Security decoration requirements to be processed normally.</p> <p>About automatic WS-Security decoration</p> <p>Automatic WS-Security decorations are those that are not</p>

Setting	Description
	<p>currently "pending", but which will be applied automatically after the policy completes.</p> <p>For example, the Require WS-SecureConversation assertion always signs the response and adds a timestamp. If this is not desirable—for example, the Gateway is configured to pass-through secure conversation—choose [Clear WS-Security] to prevent automatic decoration.</p>
<p>Tip: When both [Apply WS-Security] and [Clear WS-Security] are selected, the assertion will apply all pending WS-Security and clear any automatic WS-Security decorations that would be applied automatically after the policy completes.</p>	
<i>WS-Security options</i>	
Version	<p>Choose the version of WS-Security to use: 1.0 or 1.1. The default <Not Specified> setting will use WSS 1.0, unless WSS 1.1 is detected or explicitly configured in the policy.</p>
Select the default recipient certificate...	<p>Choose which certificate to use with XML encryption:</p> <ul style="list-style-type: none"> • Use default certificate: Use the default certificate for the recipient. Note: The certificate is for the default recipient. To override this default recipient, see "Changing the WSS Assertion Recipient" on page 146. • Use selected certificate for default recipient: Choose this option to browse for the certificate to use. Click [Select] and then locate the certificate to use. Examine the certificate details displayed to ensure that it is the correct certificate. • Lookup default recipient by name: Choose this option to use the certificate of the specified default recipient. You may reference a context variable that will resolve to the recipient at run time. If more than one certificate matches the name, then the first valid certificate is used.  • Use Certificate from Context Variable: Choose this option to use the context variable specified in the adjacent box. Note: This context variable must contain a type X.509 certificate. 

- Click **[OK]** when done.

Add Security Token Assertion

The *Add Security Token* assertion is used to signify that one of the following security tokens should be added to the SOAP security header in the target message:

WS-S UsernameToken
WS-SC SecurityContextToken
SAML Assertion (Token)
WS-S EncryptedKey

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Note: The Add Security Token assertion only adds the token to the list of pending decoration requirements for the message. The token is not actually added until the "Add or Remove WS-Security Assertion" on page 273 is executed.

Configuring the Private Key for SAML Assertions

When adding a "SAML Assertion" as the Security Token Type (Figure 87), ensure that the Add Security Token assertion is configured with the correct private key based on the SAML Assertion type:

Table 73: Configured private key for various SAML Assertion types

SAML Assertion type	Configured Private Key
Holder-of-Key	Must be the subject's key
Sender Vouches	Must be the sender's key
Bearer	Can be either the default private key for the Gateway or some other custom key
None	Can be either the default private key for the Gateway or some other custom key

To learn more about selecting a private key for this assertion, see *Selecting a Custom Private Key* in the *Layer 7 Policy Manager User Manual*.

For more information about the SAML Assertion types, see "SAML Token Profile Wizard" on page 231.

Applying WS-Security

If this assertion targets a message other than the response, you must add the [Add or Remove WS-Security](#) assertion after the Add Security Token assertion in the policy in order for the token to be applied:

Request: Add Security Token
Request: Apply WS-Security

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Tip: When WS-Security is involved, be sure to specify the appropriate WSS header handling option in the routing assertion's properties. In most instances, the setting *"Don't modify the request Security header"* is usually appropriate.

Using the Assertion

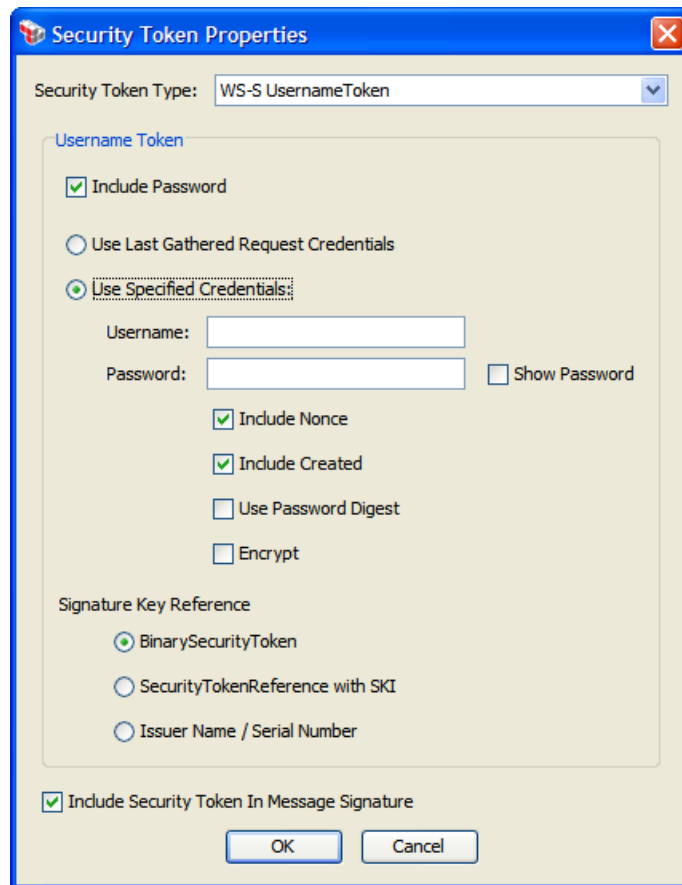
1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Add [Signed] Security Token** in the policy window and choose **Security Token Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

3. Choose a Security Token Type to add and configure as required:

WS-S UsernameToken
WS-SC SecurityContextToken
SAML Assertion
WS-S EncryptedKey

4. Set the **Include Security Token in Message Signature** check box as required:
 - Select this check box if you want the added token to be signed. (This will occur even if the token itself is responsible for the signing.) The assertion name in the policy window will appear as *"Add Signed Security Token"*.
 - Clear this check box to include the token in the Security header but not sign it. Other parts of the message may still be signed if so configured. The assertion name in the policy window will appear as *"Add Security Token"*.
5. Click **[OK]** when done.

Adding a WS-S UsernameToken



The dialog box titled "Security Token Properties" shows the configuration for a "WS-S UsernameToken". The "Security Token Type" is set to "WS-S UsernameToken". Under the "Username Token" section, the "Include Password" checkbox is checked. Below it, the "Use Specified Credentials" radio button is selected, with text boxes for "Username:" and "Password:". The "Show Password" checkbox is unchecked. Other options like "Include Nonce", "Include Created", "Use Password Digest", and "Encrypt" are unchecked. Under "Signature Key Reference", the "BinarySecurityToken" radio button is selected. The "Include Security Token In Message Signature" checkbox at the bottom is checked. "OK" and "Cancel" buttons are at the bottom right.

Figure 85: Security Token Properties - WS-S UsernameToken

Configure the settings specific to each security token type:

Table 74: Adding a WS-S UsernameToken

Setting	Description
Include Password	<p>Select this check box to include the password in the token.</p> <p>When the Include Password check box is selected, this adds a <i>wsse:Password</i> element to the security token in the target message:</p> <pre><wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">password</wsse:Password></pre> <p>This applies only where a password is provided by the requesting SOAP message (as enforced by the "Require HTTP Basic Credentials Assertion" on page 215) or is entered in the provided text box.</p>
Use Last Gathered Request Credentials	<p>Choose this option to use the credentials from the most recently gathered request.</p>

Setting	Description
Use Specified Credentials	<p>Choose this option to use credentials that you specify here:</p> <ul style="list-style-type: none"> • Username: Enter the user name to use. • Password: Enter the password to use. Available only when the password is included. <ul style="list-style-type: none"> • Choose [Show Password] if you wish the password text to be visible as it is typed in. • Clear [Show Password] to display an obfuscated password, for additional security. • Include Nonce: Select this check box to include a nonce in the token. • Use Password Digest: Select this check box to calculate and display a digest password in the <i>Password</i> element of the UsernameToken. Clear this check box to use the basic password as entered in the Password field for the <i>Password</i> element. Available only when [Include Password] is selected. • Encrypt: Select this check box to encrypt the token.
Signature Key Reference	<p>Choose the method to use to embed the signing certificate:</p> <ul style="list-style-type: none"> • BinarySecurityToken: The certificate is embedded within the message and does not require the recipient to already possess a copy of the signing certificate. This results in larger messages, but is more compatible. This setting is the default. • SecurityTokenReference with SKI: Use SecurityTokenReference containing the SubjectKeyIdentifier (SKI). This produces smaller messages, but at the risk of decreased compatibility. • Issuer Name/Serial Number: Use a SecurityTokenReference containing the certificates issuer distinguished name and serial number. This produces smaller messages, but at the risk of decreased compatibility.

Adding a WS-SC SecurityContextToken

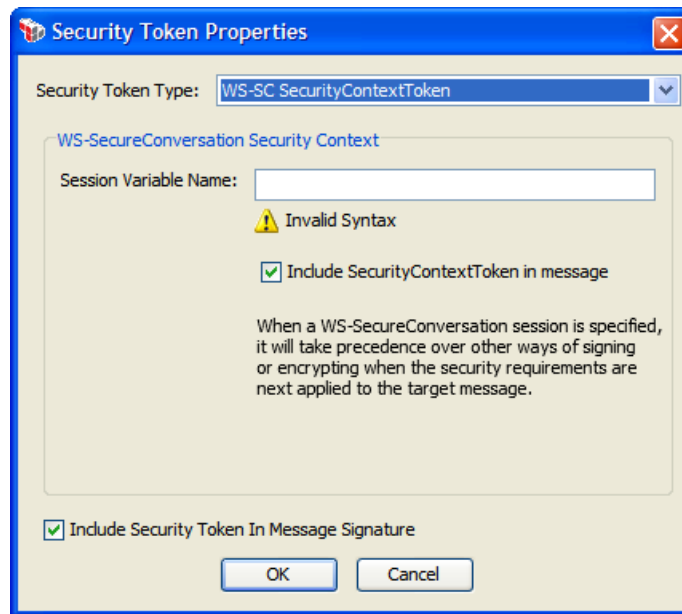



Figure 86: Security Token Properties - WS-SC SecurityContextToken

- **Session Variable Name:** Enter the context variable containing the **WS-SecureConversation Security Context**. This is normally **scLookup.session**, which is defined in the "Look Up Outbound Secure Conversation Session Assertion" on page 370. 

Tip: You can use an indexing option to specify a value from a multivalued context variable. For example, use `foo[1]` to select the second value in the multivalued variable `foo`. For more information, see "Indexing Options during Interpolation" in Working with Multivalued Context Variables in the *Layer 7 Policy Authoring User Manual*.

- **Include SecurityContextToken in message:** The default is to add a SecurityContextToken (SCT) in the message when it is decorated. **Tip:** You may need to clear this check box when decorating responses to a WCF client.

Adding a SAML Assertion

IMPORTANT: When adding a SAML Assertion as the security token, ensure that the Add Security Token assertion is configured with the correct private key. For more information, see "Configuring the Private Key for SAML Assertions" at the beginning of this topic.

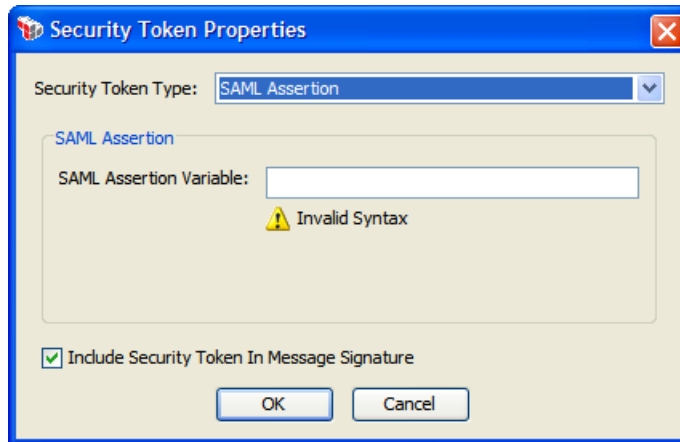



Figure 87: Security Token Properties - SAML Assertion

- **SAML Assertion Variable:** Enter the context variable containing the **SAML Assertion** (Token). This is normally **issuedSamlAssertion**, which is defined in the "Create SAML Token Assertion" on page 315. 

Tip: You can use an indexing option to specify a value from a multivalued context variable. For example, use `foo[1]` to select the second value in the multivalued variable `foo`. For more information, see "Indexing Options during Interpolation" in Working with Multivalued Context Variables in the *Layer 7 Policy Authoring User Manual*.

Adding a WS-S EncryptedKey

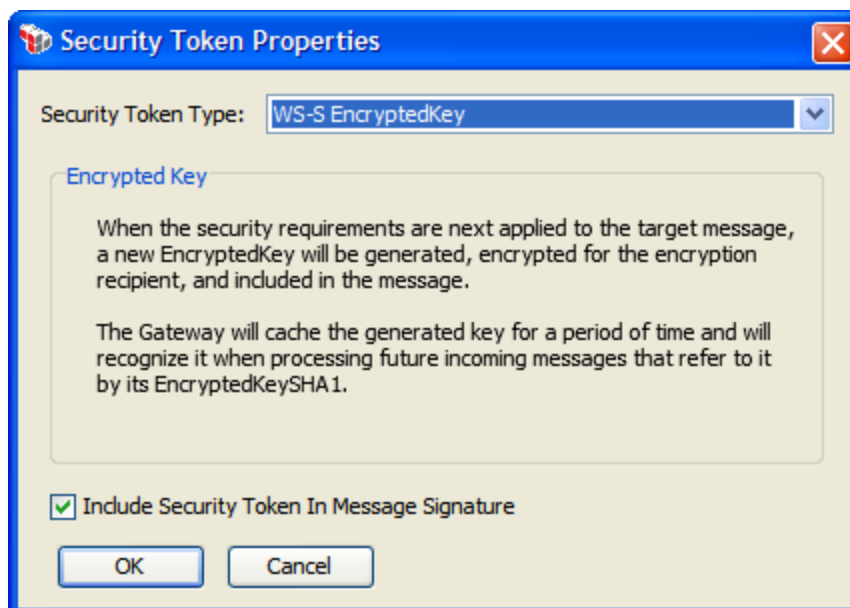


Figure 88: Security Token Properties - WS-S EncryptedKey

No further configuration is required for token type WS-S EncryptedKey. The Gateway will create a new EncryptedKey and include it in the target message when the security requirements are next applied.

The Gateway will cache the generated key for a period of time and will recognize it when processing future incoming messages that refer to it by its EncryptedKeySHA1.

Add Timestamp Assertion

The *Add Timestamp* assertion is used to add a signed `<wsu:Timestamp>` element into the SOAP security header of all target messages. You can configure the expiry time period for the timestamp and you can choose the method used to include the SSL certificate for the Gateway.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about selecting a private key for this assertion, see Selecting a Custom Private Key in the *Layer 7 Policy Manager User Manual*.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Note: The "Add or Remove WS-Security Assertion" on page 273 must follow the Add Timestamp assertion in a policy.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. This assertion contains default settings that are appropriate for most instances. To change any of the settings, right-click **<target>: Add [Signed] Timestamp** in the policy window and select **Timestamp Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

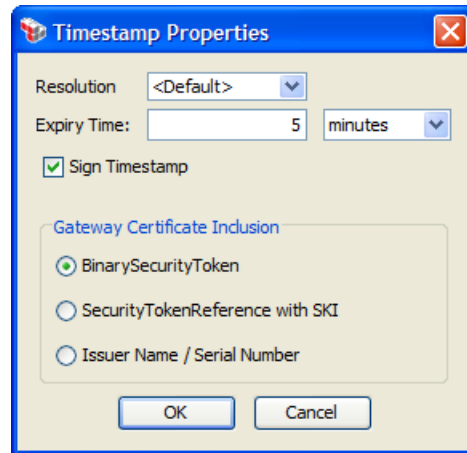


Figure 89: Timestamp Properties

3. Configure the properties as follows:

Table 75: Timestamp settings

Setting	Description
Resolution	To specify a timestamp resolution, select a value from the resolution drop-down list. When the value is ' <Default> ', the Gateway default resolution is used.
Expiry Time	Select the unit of measure from the drop-down list (milliseconds, seconds, minutes, hours), then enter the length of the expiry time for the timestamp. Fractional measurements are permitted. The default is 5 minutes.
Sign Timestamp	<p>Select this check box to digitally sign the timestamp. When signatures are used, "signed" will appear in the assertion name in the policy window ("Add signed Timestamp").</p> <p>Note: The [Sign Timestamp] check box must be enabled if a private key has been selected for this assertion. If the check box is cleared, any private key will be ignored.</p>
Gateway Certificate Inclusion	<p>Select the method to use to include the SSL certificate for the Gateway:</p> <ul style="list-style-type: none"> • BinarySecurityToken: The certificate is embedded within the message and does not require the recipient to already possess a copy of the signing certificate. This results in larger messages, but is more compatible. This setting is the default. • SecurityTokenReference with SKI: Use SecurityTokenReference containing the SubjectKeyIdentifier (SKI). This produces smaller messages, but at the risk of decreased compatibility. • Issuer Name/Serial Number: Use a SecurityTokenReference containing the certificates issuer distinguished name and serial number. This produces smaller messages, but at the risk of decreased compatibility.

4. Click **[OK]** when done.

Build RST SOAP Request Assertion

The *Build RST SOAP Request* assertion is used to create a SOAP message containing a Request Security Token (RST) in the SOAP body. The security token requested from the service is either a Security Context Token (SCT) or a SAML Token.

Context Variables Created by This Assertion

The Build RST SOAP Request assertion sets details about the RST request message in the following context variables. **Note:** The default *<prefix>* is "requestBuilder" and can be changed in the assertion properties (Figure 90).

Table 76: Context variables created by Build RST SOAP Request assertion

Variable	Description
<i><prefix>.rstRequest</i>	Stores the RST Request message generated
<i><prefix>.clientEntropy</i>	Stores the client entropy, if the option [Generate and include client entropy] is selected in the assertion properties

Using the Assertion




1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Build RST SOAP [Cancel|Issue|Validate] Request** in the policy window and choose **RST SOAP Request Builder Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 90: RST SOAP Request Builder Properties

3. Configure the properties as follows.

Table 77: RST SOAP Request Builder settings

Setting	Description
SOAP Version	Choose the SOAP version to be used in the RST SOAP message: 1.1 or 1.2.
WS-Trust Namespace	Choose the WS-Trust namespace to be used in a <i>RequestSecurityToken</i> element: <ul style="list-style-type: none"> http://docs.oasis-open.org/ws-sx/ws-trust/200512 (v1.3 and v1.4) http://schemas.xmlsoap.org/ws/2005/02/trust (v1.2) http://schemas.xmlsoap.org/ws/2004/04/trust (pre-v1.2)
Token Type	Choose the token type to be used in the message: <p><Not Included> (no token is requested)</p> <p>SAML2 Assertion</p> <p>SAML Assertion</p> <p>WS-SC SecurityContextToken</p>
Request Type	Choose the type of request to build: <p>Cancel</p> <p>Issue (default)</p> <p>Validate</p>

Setting	Description
<wst:Issuer> Address 	Optionally specify the issuer of the security token that is presented in the RST SOAP request message. The Issuer element's type is an endpoint reference as defined in WS-Addressing. You may reference context variables.
<wsp:AppliesTo> Address 	Optionally specify the URL of the <Address> in a <wsp:AppliesTo> element, which is a scope specified by the requestor for the issued token. You may reference context variables.
Target Token Variable 	<p>If the Request Type is either Cancel or Validate, optionally specify a context variable of type String that will be used for the target element (the <i>CancelTarget</i> or <i>ValidateTarget</i> elements, respectively). This context variable should either contain:</p> <ul style="list-style-type: none"> an Element—for example, one that was selected using an XPath; this Element should be a <i>SecurityTokenReference</i> or a security token. a Security Context Token (for example, one created by the "Establish Outbound Secure Conversation Assertion" on page 348). A <i>SecurityTokenReference</i> will be generated for the token. <p>Tip: You can use an indexing option to specify a value from a multivalued context variable. For example, use <i>foo[1]</i> to choose the second value in the multivalued variable <i>foo</i>. For more information, see "Indexing Options during Interpolation" in Working with Multivalued Context Variables in the <i>Layer 7 Policy Authoring User Manual</i>.</p>
Key Size (bits)	Optionally specify the key size in bits.
Token Lifetime	<p>Optionally, select this check box to specify a time range for the returned security token.</p> <p>Note: The issuer is not obligated to honor this range and may return a more (or less) restrictive interval.</p>
Use System Default	<p>When specifying a Token Lifetime, select this check box to use the system default, as defined by the <i>outbound.secureConversation.defaultSessionDuration</i> cluster property. The default value for this property is 2 hours.</p>
Generate and include client entropy	<p>Optionally select this check box to generate client entropy and include it in the RST request. The generated entropy will be saved into the context variable <prefix>.clientEntropy (see Table 76).</p>
Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>The default prefix is requestBuilder.</p>

Setting	Description
	For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i> .

- Click **[OK]**.

Build RSTR SOAP Response Assertion

Once a security token is issued by the [Create Security Context Token](#) assertion, the *Build RSTR SOAP Response* Assertion is used to create a SOAP response message containing a *RequestSecurityTokenResponse* (RSTR) element. This element does the following:

- For [token issuance](#), the RSTR will wrap the token issued.
- For [token cancellation](#), the RSTR will contain a *RequestedTokenCanceled* element.

This assertion can create two types of responses:

- A response that indicates a security token is being issued.
- A response that indicates a security token is being canceled.

The following is an example of the RSTR element in a response message:

```
<wst:RequestSecurityTokenResponse Context="..." xmlns:wst="...">
  <wst:TokenType>...</wst:TokenType>
  <wst:RequestedSecurityToken>
    // The issued security token appears here
  </wst:RequestedSecurityToken>
  ...
</wst:RequestSecurityTokenResponse>
```

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153. The target message is an inbound RST Request SOAP message that provides the following useful information for building the RSTR:

- All related namespaces such as SOAP Envelope, WS-Trust, WS-Secure Conversation, WS-Addressing, WS-Policy, WS-Security, etc.
- Entropy in the RST request, if applicable
- Key size in the RST request, if applicable.

To learn more about selecting the target identity for this assertion, see "Selecting a Target Identity" on page 152.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Context Variables Created by This Assertion

The Build RSTR SOAP Response assertion sets the following context variables with details of the response. **Note:** The default *<prefix>* is "responseBuilder" and can be changed in the assertion properties (Figure 91).

Table 78: Context variables created by Build RSTR SOAP Response Assertion

Variable	Description
<i><prefix></i> .rstrResponse	Contains the RSTR SOAP response message.
<i><prefix></i> .wsaNamespace	Contains the WS-Addressing namespace that will be used when WS-Addressing is added to the RSTR SOAP response message.
<i><prefix></i> .rstrWsaAction	Contains the RSTR WS-Addressing Action that will be used when WS-Addressing is added to the RSTR SOAP response message.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Build RSTR SOAP Response** in the policy window and choose **RSTR SOAP Response Builder Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

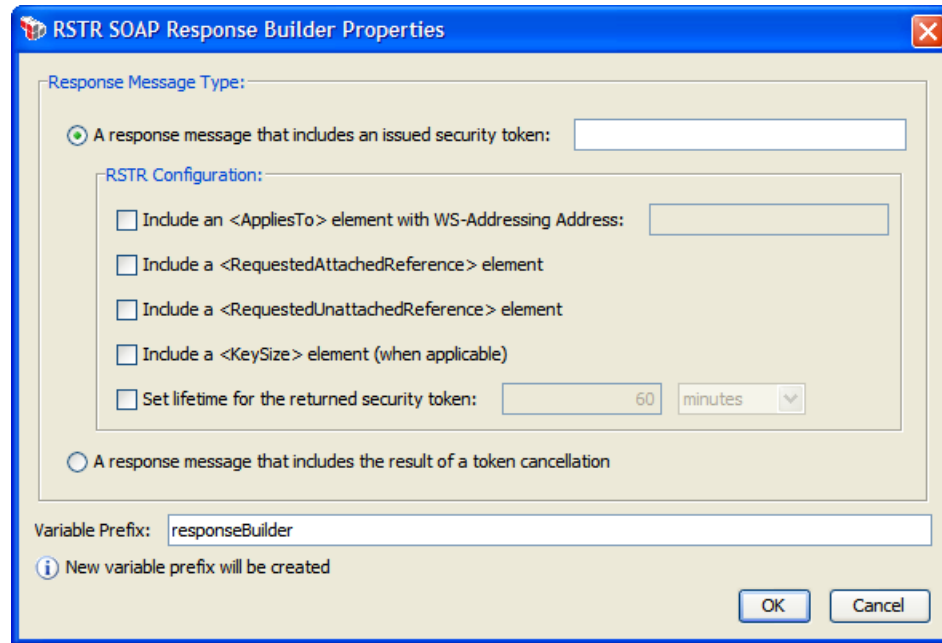


Figure 91: RSTR SOAP Response Builder Properties

3. Specify the **Response Message Type** to create:

- **A response message that includes an issued a security token:** Choose this option to create a response with the `<wst:RequestSecurityTokenResponse>` element, which contains the security token that was issued (either a [SAML Token](#) or a [Security Context Token](#)).

Enter the context variable that contains the issued token. **Tip:** For Security Context Tokens, this will be the `${<prefix>.issuedSCT}` variable. For SAML Tokens, this should be the `${issuedSamlAssertion}` variable.

Optionally choose any of the following **RSTR Configuration** options to include:

- **Include an <AppliesTo> element with WS-Addressing Address:** This optional element is used to specify the address attribute of the endpoint reference.. For example, the service(s) to which this token applies.
- **Include a <RequestedAttachedReference> element:** Since returned tokens are considered opaque to the requestor, this optional element is used to indicate how to reference the returned token when that token does not support references using URI fragments (XML ID).

- **Include a <RequestedUnattachedReference> element:** In certain cases, tokens do not need to be present in the message. This optional element is used to indicate how to reference the token when it is not placed inside the message.
- **Include a <KeySize> element:** The size of the session key for a secure conversation token will be included in the generated message.
- **Set Lifetime for the returned security token:** This lifetime defines the expiry duration of the returned security token. It is not the same as the lifetime defined in the "Create Security Context Token Assertion" on page 328. The default expiry duration is **60** minutes for the returned security token. **Note:** It is recommended that the issuer return this element with issued tokens (in the RSTR) so the requestor knows the actual validity period without needing to parse the returned token.

Note: For more information about the RSTR Configuration elements, please refer to the specifications on WS-Trust and WS-Secure Conversation on www.oasis-open.org.

- **A response message that includes the results of a token cancellation:**
Choose this option to create a response with `<wst:RequestedTokenCanceled/>` in the `<wst:RequestSecurityTokenResponse>` element.
4. Enter a prefix to be added to the context variables [created by this assertion](#). The prefix allows you to uniquely identify the variables if the assertion appears more than once within a policy.

The default variable prefix is **responseBuilder**.

5. Click **[OK]**.

Build SAML Protocol Request Assertion

The *Build SAML Protocol Request* assertion is used to create a SAML request from either a request message, response message, or a Message variable. If the request can be successfully fulfilled, a SAML request is returned containing one or more SAML tokens. The [Evaluate SAML Protocol Response](#) assertion is then used to evaluate the request, response, or Message variable.

The target message for this assertion is set within the wizard, but it may also be changed in the policy window, without using the wizard. For more information, see "Selecting a Target Message" on page 153.

To learn more about selecting a private key for this assertion, see *Selecting a Custom Private Key* in the *Layer 7 Policy Manager User Manual*.

The Build SAML Protocol Request assertion is typically used as follows in a policy:

Build SAML Protocol Request
[Route via HTTP\(S\)](#)
[Evaluate SAML Protocol Response](#)

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Build SAML Protocol Request...** in the policy window and select **SAML Protocol Request Wizard** or double-click the assertion in the policy window.
3. Follow the wizard to complete the assertion. For details, see "SAML Protocol Request Wizard" on page 292.

SAML Protocol Request Wizard

The SAML Protocol Request Wizard automatically starts when you add or edit a [Build SAML Protocol Request](#) assertion in a policy.

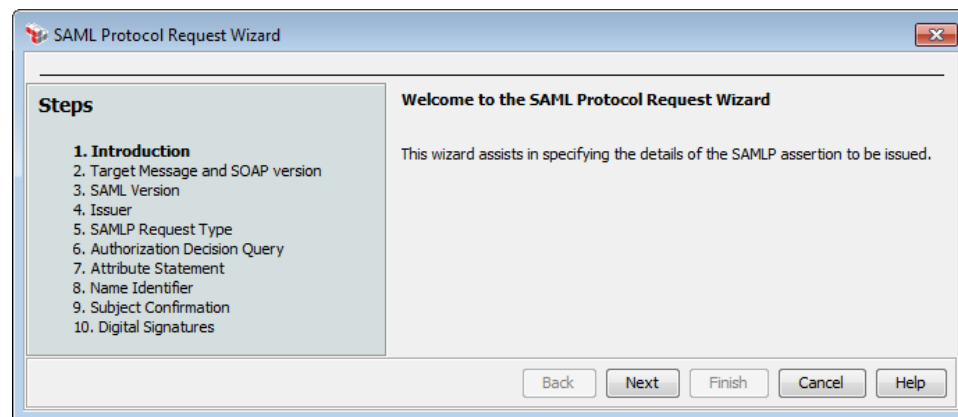



Figure 92: SAML Protocol Request Wizard

For more information about wizards, see "Wizard" under Interfaces in the *Layer 7 Policy Manager User Manual*.

Tip: You can use context variables in many of the text fields in the wizard. These variables are evaluated at runtime as the SAML Protocol request is being constructed.

Table 79: Using the SAML Protocol Request Wizard

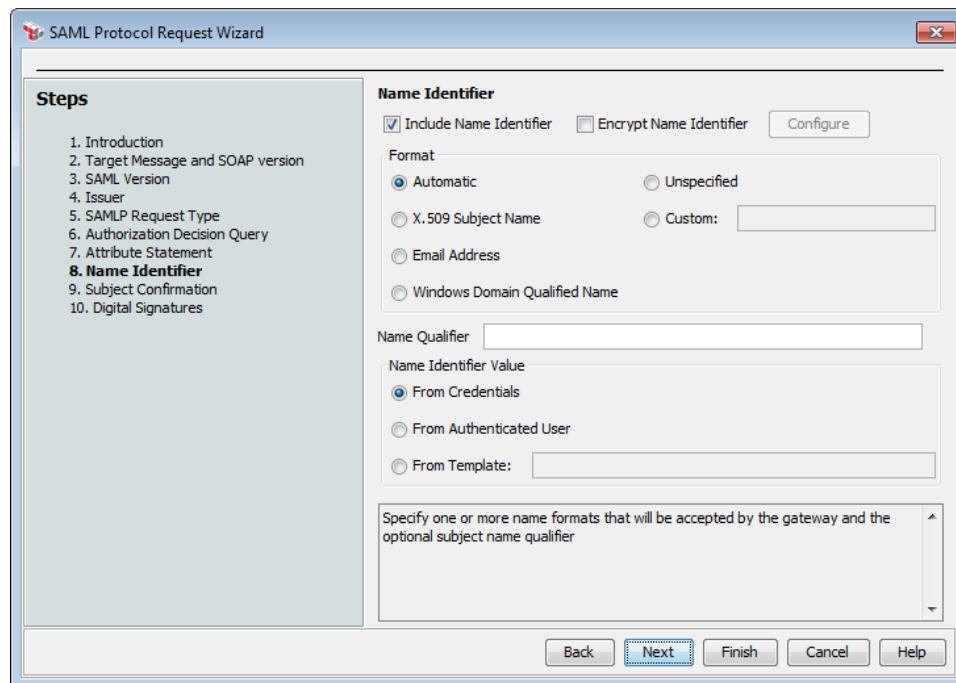
Wizard Step	Descriptions
Step 1: Introduction	Introduces the wizard.
Step 2: Target Message and SOAP Version	<ul style="list-style-type: none"> • Target Message: Choose the target location to set the SAML query: Request, Response, or some Other Message Variable, with the default being <i>samlpRequest.message</i>. For more information on Message variables, see Context Variables in the <i>Layer 7 Policy Manager User Manual</i>. To learn how to change the message target, see "Selecting a Target Message" on page 153. • SOAP Version: Specify the SOAP version to use: 1.1, 1.2, or use version from request.
Step 3: SAML Version	<ul style="list-style-type: none"> • Create a SAML query....: Choose the version of the SAML query request that will be created. • Request Identifier: Choose to have the wizard generate a request identifier or reference a context variable that contains the identifier.  • Optional Request Attributes: Optionally specify a Destination URI or Consent URI.
Step 4: Issuer (SAML 2.0 only)	Configure the Issuer attribute value. For a description of these settings, see " Configuring the [Issuer] Tab " in "Build SAML Protocol Response Assertion" on page 299.
Step 5: SAML Request Type	<p>Specify the SAML query request to be configured:</p> <ul style="list-style-type: none"> • Authentication Request: Select this option to request assertions containing authentication statements to establish a security context at one or more replying parties. Proceed to Step 8 to configure this request type. Note: The Authentication Request option is available only when SAML 2.0 was selected in Step 3. • Authorization Decision Request: Select this option to request whether an assertion subject has permission to access the specified resources. Proceed to Step 6 to configure this request type. • Attribute Query Request: Select this option to make a query that requests the assertion subject associated with the supplied attributes. Proceed to Step 7 to configure this request type.
Step 6: Authorization Query	<p>This step is used if you chose "Authorization Decision Request" in Step 5.</p> <ul style="list-style-type: none"> • Resource: Specify the URI for the resource for which authorization is requested.

Wizard Step	Descriptions
	<ul style="list-style-type: none"> • Action: Specify one or more actions for which authorization is requested. • Action Namespace: Optionally specify a URI reference representing the namespace in which the specified action should be interpreted. • Evidence: Indicate whether the wizard should generate the appropriate evidence block or whether it should obtain the evidence block from a context variable.
Step 7: Attribute Statement	<p>This step is used if you chose "Attribute Query Request" in Step 5.</p> <p>Define the attributes that the SAML statement will include.</p> <ol style="list-style-type: none"> Click [Add] and then complete the Edit SAML Attribute Properties dialog: <ul style="list-style-type: none"> • Attribute Name: Enter the name of the attribute. • Attribute Namespace: Optionally enter a namespace for the attribute. This applies only to SAML 1.x. • Attribute Name Format: Optionally specify a URI reference that describes the format of the attribute name. Only attributes that declare this format will be accepted. This applies only to SAML 2.x. <ul style="list-style-type: none"> • Unspecified: If no name format is provided, the default value of <code>urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified</code> is used. • URI Reference: This option uses the URI <code>urn:oasis:names:tc:SAML:2.0:attrname-format:uri</code> • Basic: This option uses the URI <code>urn:oasis:names:tc:SAML:2.0:attrname-format:uri</code>. • Other: Select this option to define your own attribute name format in the box below. • Friendly Name: Optionally enter a friendly name for the attribute to be used for display purposes. This applies only to SAML 2.x. • Attribute Value: If defining your own attribute name format, enter it here. This applies only to SAML 2.x. Click [OK] to enter the attribute into the table. Repeat to configure additional attributes. <p>To modify an existing Attribute Statement, select it from the list and then click [Edit].</p> <p>To remove an Attribute Statement, select it from the list and then click [Remove].</p>
Step 8: Name Identifier	Enter the details for the Name Identifier.

Wizard Step	Descriptions
	See "Configuring the Name Identifier" below for details.
Step 9: Subject Confirmation	Configure the subject confirmation method in this step. See "Configuring the Subject Confirmation" below for details.
Step 10: Digital Signature	<p>Select the Sign Request check box to include a digital signature in the request. Clear the check box to not include a digital signature.</p> <p>A digital signature is not always required in SAML. The following are some examples where the signature may not be required:</p> <ul style="list-style-type: none"> When a signature is "inherited"—an unsigned assertion gains protection from a signature in the containing protocol response message. The SAML requestor has obtained an assertion from the SAML authority directly, through a secure channel. In this case, the SAML authority has been verified using means other than a digital signature.

Configuring the Name Identifier

This wizard step configures the details for the Name Identifier in the SAML Protocol Request.



The screenshot shows the 'SAML Protocol Request Wizard' window, specifically Step 8: Name Identifier. On the left, a 'Steps' pane lists the wizard's progression, with '8. Name Identifier' highlighted. The main area is titled 'Name Identifier' and contains the following options:

- ☒ Include Name Identifier ☐ Encrypt Name Identifier [Configure](#)
- Format**
 - ☒ Automatic ☐ Unspecified
 - ☐ X.509 Subject Name ☐ Custom:
 - ☐ Email Address
 - ☐ Windows Domain Qualified Name
- Name Qualifier**
- Name Identifier Value**
 - ☒ From Credentials
 - ☐ From Authenticated User
 - ☐ From Template:
- A text area at the bottom: 'Specify one or more name formats that will be accepted by the gateway and the optional subject name qualifier'.

At the bottom of the window are buttons for 'Back', 'Next' (highlighted), 'Finish', 'Cancel', and 'Help'.

Figure 93: SAML Protocol Request Wizard - Step 8: Name Identifier (SAML 2.x version shown)

1. Select the **Include Name Identifier** check box to include the Name Identifier in the SAML token.

Clear the check box to not include the Name Identifier. This disables all the remaining settings in the wizard step; click **[Next]** to proceed to the next step in the wizard.


2. Select the **Encrypt Name Identifier** check box to encrypt the Name Identifier. This causes a `<saml:EncryptedID>` to be placed in the `<saml:Subject>` element.

Clear the check box to not encrypt the Name Identifier. This will place a `<saml:NameID>` in the `<saml:Subject>` element.

3. If encrypting the Name Identifier, click **[Configure]** and complete the EncryptedID Encryption Properties. For more information, see Configuring Encryption Settings in the *Layer 7 Policy Manager User Manual*.

If not encrypting the Name Identifier, skip to step 4.


4. Choose the **Format** of the Name Identifier:

- **Automatic:** The Name Identifier Format URI will be selected based on the type of credentials used to authenticate the user.
- **X.509 Subject Name:** The Name Identifier Format URI is the X.509 Subject Name.
- **Email Address:** The Name Identifier Format URI is the email address.
- **Windows Domain Qualified Name:** The Name Identifier Format URI is the Windows Domain Qualified Name.
- **Unspecified:** Indicates that the issuer of the SAML token is not warranting that the Name Identifier value meets any particular format expectations.
- **Custom:** Enter a custom Name Identifier Format URI. You may specify a context variable. Ensure that the URI is valid to prevent the assertion from failing. 

5. Optionally enter a **Name Qualifier** template. This value determines the security or administrative domain of the subject. An example of a Name Qualifier might be the Gateway hostname (for example, *gatewayhost.acmecorp.com*). It is not necessary to enter a fully-qualified hostname.

6. For **Name Identifier Value**, indicate where the value of the Name Identifier is to be retrieved:

- **From Credentials:** The value is the user name from the credentials used to authenticate the user.

- **From Authenticated User:** The value is the most appropriate attribute (matching the selected Format) available from the user who was authenticated.
- **From Template:** The value is the result of evaluating the specified template. This will typically be a context variable, perhaps one resulting from an XPath ([Evaluate Request XPath](#) or [Evaluate Response XPath](#)) or from the [Extract Attributes for Authenticated User Assertion](#). 

Configuring the Subject Confirmation

This wizard step configures the subject confirmation method to be used in the SAML Protocol Request.

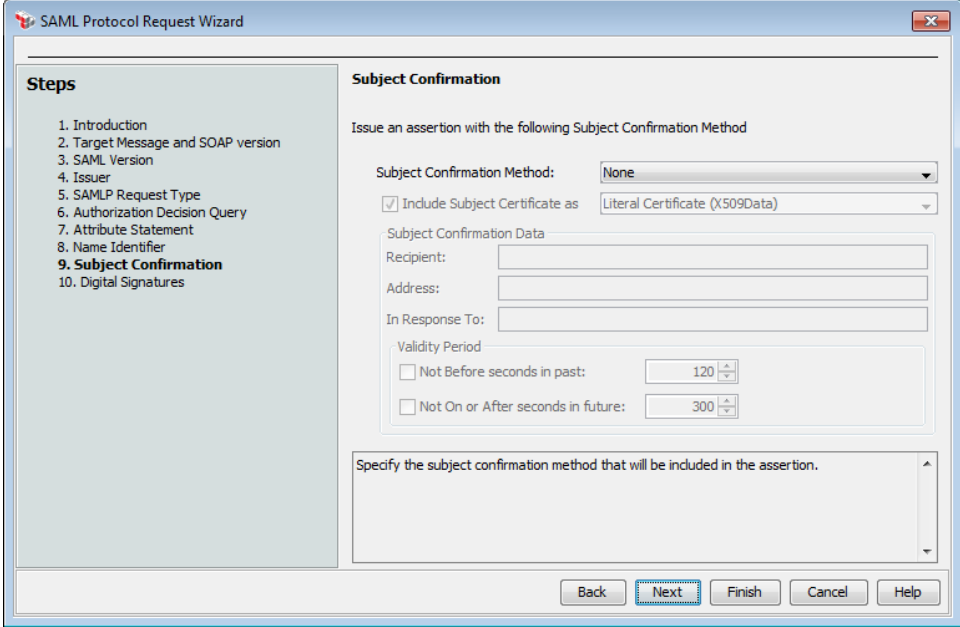


Figure 94: SAML Protocol Request Wizard - Step 9: Subject Confirmation

1. Choose the **Subject Confirmation Method** to be used in the issued SAML token. This allows the SAML-relying party to confirm that the message came from a system entity that corresponds to the subject in the statement or query.

Holder-of-Key

The SAML token will use the *Holder-of-Key* subject confirmation method (with the standard URI `urn:oasis:names:tc:SAML:1.0:cm:holder-of-key` or `urn:oasis:names:tc:SAML:2.0:cm:holder-of-key`, depending on the selected SAML version in Step 3 of the wizard). For such assertions, the Gateway will require that the subject demonstrate possession of the private key corresponding to the public key in the Subject certificate.

The request Subject may use one of two methods to prove that they hold this key:

- The request includes at least one element covered by a valid [WSS message signature](#). The signing certificate will be used as the Subject certificate. Or,
- The request arrived over SSL/TLS with client certificate. The client certificate will be used as the Subject certificate.

Sender Vouches

The SAML token will use the *Sender Vouches* subject confirmation method (with the standard URI *urn:oasis:names:tc:SAML:1.0:cm:sender-vouches* or *urn:oasis:names:tc:SAML:2.0:cm:sender-vouches*, depending on the selected SAML version in Step 2 of the wizard). For such assertions, the Gateway vouches for the verification of the subject.

Bearer

The SAML token will use the *Bearer Token* subject confirmation method (with the standard URI *urn:oasis:names:tc:SAML:1.0:cm:bearer* or *urn:oasis:names:tc:SAML:2.0:cm:bearer*, depending on the selected SAML version in Step 2 of the wizard). Like HTTP cookies, such assertions will always be assumed to belong to whatever message contains them, and the subject will be assumed to be the sender of the message.

None




The SAML token does not have a subject confirmation method.

2. Configure the **Include Subject Certificate** check box as required. This is available on when the **Subject Confirmation Method** is "Holder-of-Key".

Select this check box to specify that the subject's certificate (or a reference to it) will be included in the SAML token. Choose the method by which it should be included or referenced from one of the following options.

- **Literal Certificate (X509Data):** The entire subject certificate is inserted into the SAML token. This increases the size of the assertion significantly, but will mean that the recipient does not have to locate the subject certificates.
- **SecurityTokenReference using SKI:** A Subject Key Identifier (SKI) from the certificate is included in the SAML token. This results in a smaller assertion, but it requires that the recipient look up the subject certificate.
- **SecurityTokenReference using SHA1 Thumbprint:** An SHA1 thumbprint from the certificate is included in the SAML token. Like the SKI option above, this produces a smaller assertion, but it requires that the recipient look up the subject certificate.

Clear this check box to not include the subject's certificate (or reference to it) in the SAML token.

3. If SAML 2.0 is used and the Subject Confirmation Method is not set to "None", optionally complete the **Subject Confirmation Data** section. These fields provide additional information to be used by a specific confirmation method.
 - **Recipient:** Enter a URI that specifies the required entity or location. For example, this attribute might indicate that a resulting SAML token must be delivered to a particular network endpoint in order to prevent an intermediary from redirecting it someplace else. You may reference context variables. 
 - **Address:** Enter the required network address or location. For example, this attribute might be used to bind a resulting SAML token to particular client addresses to prevent an attacker from stealing and presenting the token from another location. You may reference context variables. 
 - **In Response To:** Enter the required message ID. For example, this attribute might be used to correlate the resulting SAML token to the related SAML request. You may reference context variables. 
4. If SAML 2.0 is used and the Subject Confirmation Method is not set to "None", optionally complete define a **Validity Period** for the SAML token:
 - **Not Before seconds in past:** Select this check box and then enter the number of seconds in the past before which the subject cannot be confirmed. The default is **120** seconds.
 - **Not On or After seconds in future:** Select this check box and then enter the number of seconds into the future after which the subject can no longer be confirmed. The default is **300** seconds.

Build SAML Protocol Response Assertion

The *Build SAML Protocol Response* assertion places a SAML token into a SAML Protocol <Response> message and allows various attributes/elements of <Response> to be specified.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

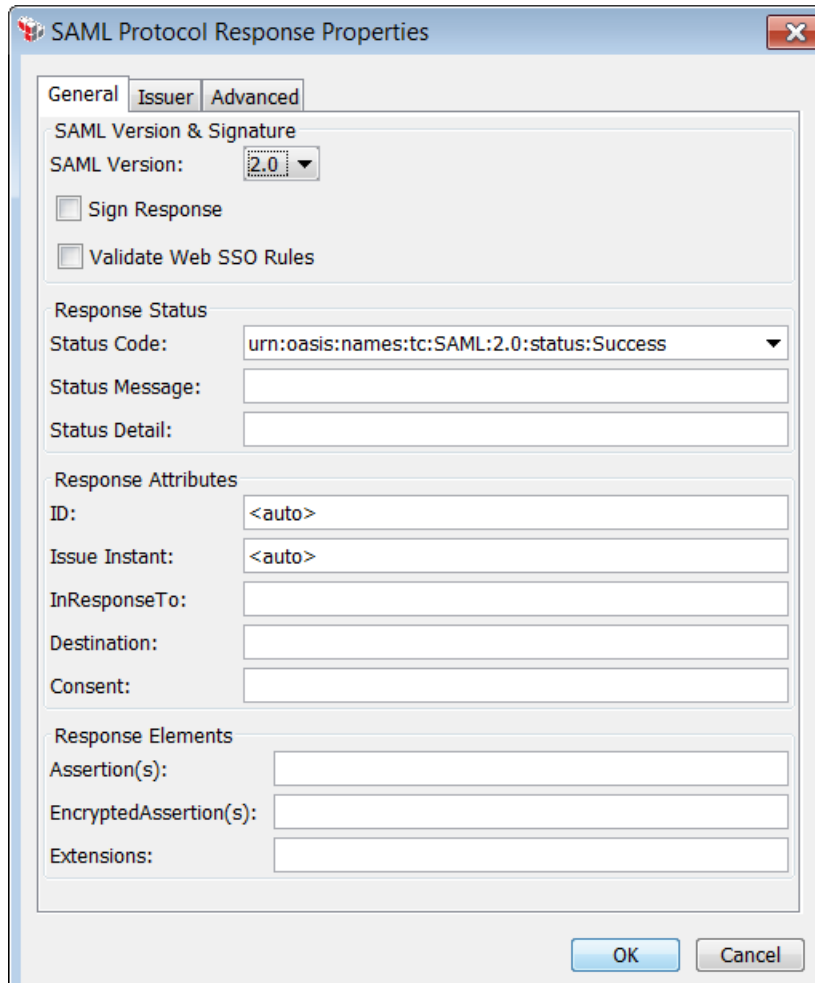
Note: If you select a context variable for the target message, that variable does not need to exist already. The variable will be overwritten if it exists.

To learn more about selecting a private key for this assertion, see Selecting a Custom Private Key in the *Layer 7 Policy Manager User Manual*.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Build SAML Protocol Response** in the policy window and select **SAML Protocol Response Properties** or double-click the assertion in the policy window. The assertion properties are displayed.
3. Configure each tab as necessary. Note the fields differ depending on whether SAML 2.0 or SAML 1.1 is selected. Refer to the appropriate section below for a description of each tab.
4. Click **[OK]** when done.

Configuring the [General] Tab



SAML Protocol Response Properties

General | Issuer | Advanced

SAML Version & Signature

SAML Version: **2.0**

☐ Sign Response

☐ Validate Web SSO Rules

Response Status

Status Code: urn:oasis:names:tc:SAML:2.0:status:Success

Status Message:

Status Detail:

Response Attributes

ID: <auto>

Issue Instant: <auto>

InResponseTo:

Destination:

Consent:

Response Elements

Assertion(s):

EncryptedAssertion(s):

Extensions:

OK Cancel

Figure 95: SAML Protocol Response Properties - [General] tab (SAML 2.0 shown)

Configure this tab as follows:

Table 80: SAML Protocol Response Properties - [General] tab

Setting	Description
SAML Version	Select the SAML version from the drop-down list: 2.0 or 1.1 . Default: 2.0 .
Sign Response	Select this check box if the response should be digitally signed. For more information about selecting a private key for the signature, see <i>Selecting a Custom Private Key</i> in the <i>Layer 7 Policy Manager User Manual</i> .
Validate Web	Select this check box if you want the assertion to validate Web SSO

Setting	Description
SSO Rules	<p>profile rules. If any rule is broken, the assertion fails and a warning is logged. For a description of the rules validated, see "Validating Web SSO Profile Rules" below.</p> <p>Clear this check box if the assertion will be used in situations outside of Web SSO and such validation is not desired (for example, SAML Protocol Attribute Query Responses—see Step 4 in the Evaluate SAML Protocol Response assertion).</p>
<i>Response Status</i>	
Status Code	<p>Specify a response status using either of the following methods:</p> <ul style="list-style-type: none"> Choose the response status from the drop-down list. By default, these responses are used: <ul style="list-style-type: none"> SAML 1.1: Success SAML 2.0: urn:oasis:names:tc:SAML:2.0:status:Success Specify a context variable that will resolve to a valid status code for the SAML version at run time. You will typically use the context variable set by the "Set SAML Response Status Code Assertion" on page 480.
Status Message	Enter a status message to be returned in the response. This message may reference String context variables.
Status Detail	<p>Optionally specify the status detail to be returned in the response. You must use context variables; text entry is not permitted. The variables may be concatenated or separated with a space.</p> <p>The variables may be of type Element, Message (text/xml), or String and may be multivalued.</p>
<i>Response Attributes</i>	
ResponseId (SAML 1.1 only)	<p>Enter the ID for the SAML response. May reference String context variables.</p> <p>Default: <auto>. This indicates that the system will automatically fill the field if no ResponseID is entered.</p>
ID (SAML 2.0 only)	<p>Enter the ID for the SAML response. May reference String context variables.</p> <p>Default: <auto>. This indicates that the system will automatically fill the field if no ID is entered.</p>
Issue Instant	<p>Specify the <i>IssueInstant</i> property to be used in the response. This property contains the date and time when the response was issued. May reference String context variables.</p> <p>Default: <auto>. This indicates that the system will automatically fill the field if no IssueInstant is entered.</p>

Setting	Description
InResponseTo	Optionally specify the <i>InResponseTo</i> value to be used in the response. This is an identifier to the request to which this response may correspond. May reference String context variables.
Recipient (SAML 1.1 only)	Specify the intended recipient for this response. May be a String context variable.
Destination (SAML 2.0 only)	Specify the URI to which the response will be sent.
Consent (SAML 2.0 only)	Specify the Consent property. This indicates whether consent was obtained from a principal in sending the response. May reference String context variables.
<i>Response Elements</i>	
Assertion(s)	<p>Enter one or more context variables containing the SAML tokens to be returned in the response, in the format: <code>\${variableName}</code>. The variables may be concatenated or separated with a space.</p> <p>These variables may be of type Element, Message (text/xml), or String. Variables may be multivalued.</p> <p>Tip: Variables of type Element are created by the Evaluate Request XPath and Evaluate Response XPath assertions in the ".elements" context variable.</p>
EncryptedAssertion(s) (SAML 2.0 only)	<p>Enter one or more context variables containing the encrypted SAML tokens to be returned in the response, in the format: <code>\${variableName}</code>. The variables may be concatenated or separated with a space.</p> <p>These variables may be of type Element, Message (text/xml), or String. Variables may be multivalued.</p>
Extensions (SAML 2.0 only)	<p>Optionally enter one or more context variables, separating them with a space.</p> <p>These variables may be of type Element, Message (text/xml), or String. Variables may be multivalued.</p>

Validating Web SSO Profile Rules

This assertion validates the following profiles rules when the Validate Web SSO Rules check box in the [General] tab is selected.

For SAML 2.0:

Note: If an encrypted token is present in the *samlp:response*, then no rules relating to the enclosed *saml:assertion* (SAML tokens) can be validated, as the Gateway cannot examine the contents of encrypted SAML tokens.

The following Web SSO profile rules are validated:

- If the Idp (SAML Web Browser SSO Profile Identity Provider) wants to return an error, then the *<Response>* must not contain any assertions.
- If the *<Response>* message is signed or if an enclosed assertion is encrypted, then the *<Issuer>* element must be present.
- Response must contain at least one *<Assertion>*, the same rule above for *<Issuer>* applies for each assertion.
- All assertions in the response must be from the same Identity Provider (for example, the same Gateway).
- If multiple assertions are included, then each *<Subject>* element must refer to the same principal.
- Any assertion issued must contain a *<Subject>* element with at least one *<SubjectConfirmation>* element containing a Method of *urn:oasis:names:tc:SAML:2.0:cm:bearer*.
- The bearer *<SubjectConfirmation>* element must contain a *<SubjectConfirmationData>* element that itself **must** contain a Recipient attribute containing the service provider's assertion consumer service URL and a *NotOnOrAfter* attribute that limits the assertion.
 - It must not contain a *NotBefore* attribute.
- The set of bearer assertions must contain at least one *<AuthnStatement>* that reflects who the principal was authenticated.
- Each bearer assertion must contain an *<AudienceRestriction>* including the SP's unique identifier as the *<Audience>* (for example, the web site's URL)
- If no SAML tokens are specified for the response, then the Status Code cannot be "Success", as the response that is generated must be an error.

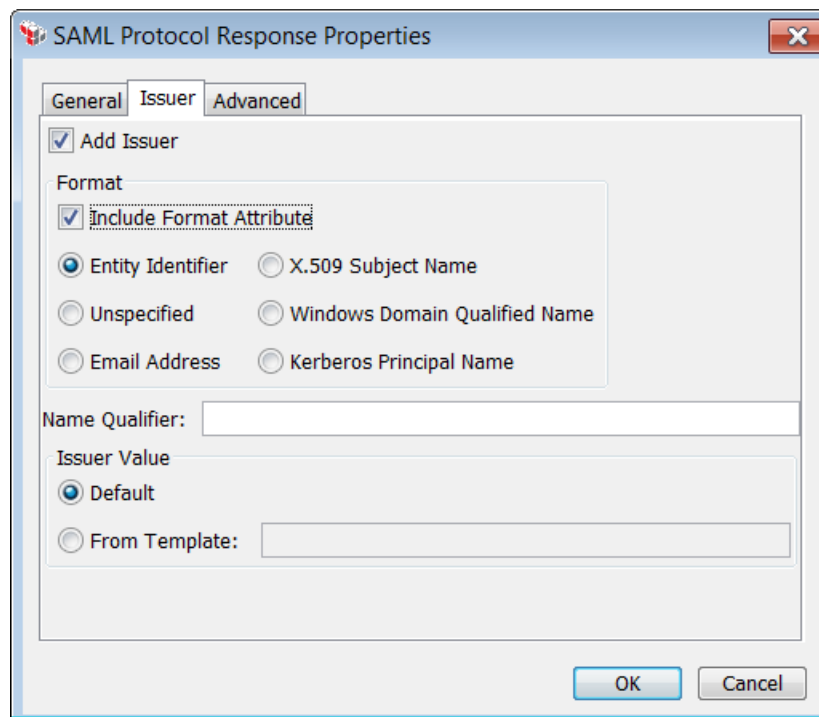
For SAML 1.1:

The following Web SSO rules are validated:

- At least one SSO assertion must be included. An SSO assertion is a SAML token that has a *<saml:Conditions>* element with *NotBefore* and *NotOnOrAfter* attributes present, and also contains at least one or more authentication statements about the subject.
- SAML Response must include the Recipient attribute - *xsd:anyURI*

- Every subject-based statement in the assertion(s) returned to the destination site **must** contain a `<saml:SubjectConfirmation>` element. The `<ConfirmationMethod>` element in the `<SubjectConfirmation>` **must** be set to `urn:oasis:names:tc:SAML:1.0:cm:bearer`.
- If no SAML tokens are specified for the response, then the Status Code cannot be "Success", as the response that is generated must be an error.



Configuring the [Issuer] Tab (SAML 2.0 only)



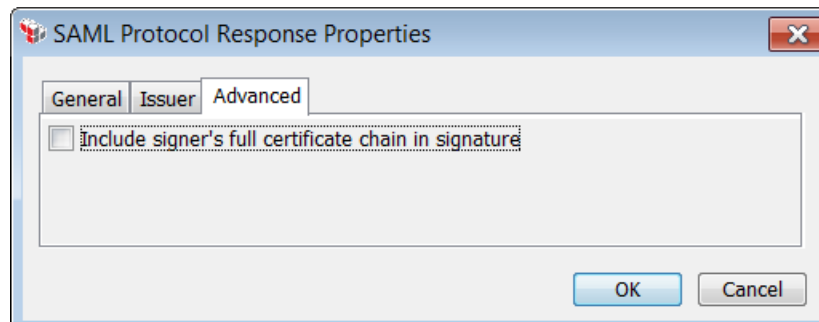
Configure this tab as follows.

Table 81: SAML Protocol Response Properties - [Issuer] tab - SAML 2.0 only

Setting	Description
Add Issuer	Select this check box to add the <i>Issuer</i> element in the SAML Protocol response. This enables the other settings in the tab.
Include Format Attributes	<p>Select this check box to include the <i>Format</i> attribute in the SAML token and then select format of the Issuer attribute:</p> <ul style="list-style-type: none"> <i>Entity Identifier</i> <i>X.509 Subject Name</i> <i>Unspecified</i> <i>Windows Domain Qualified Name</i> <i>Email Address</i> <i>Kerberos Principal Name</i>

Setting	Description
Name Qualifier	Enter the value for the optional <i>NameQualifier</i> attribute. You may reference context variables. 
Issuer Value	Specify how to obtain the value of the Issuer: <ul style="list-style-type: none"> • Default: Select this to use the subject DN from public key that corresponds to the configured private key. • From Template: Select this to override the default by entering a custom value. You may reference context variables. 

Configuring the [Advanced] tab



Select this check box to include the entire certificate chain from the signing private key when signing the response. This includes the root certificate as well as any intermediate certificates.

Tip: The chain that is included will include the full path to the CA certificate only if the corresponding private key certificate chain is complete. If it is a partial chain or if only the subject certificate is available, then selecting the check box will have no effect.

Clear this check box to use only the X.509 certificate data from the signing certificate. The rest of the certificate chain is ignored. This is the default.

Cancel Security Context Assertion

The *Cancel Security Context* assertion is used to cancel a secure conversation session (either inbound or outbound) that is no longer in use.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153. The target message should be an RST SOAP message with the CanceledTarget information.

Canceling Inbound Session

Inbound sessions have a Security Context Token (SCT) created by the "Create Security Context Token Assertion" on page 328. Once canceled, this token is no longer valid for authentication and authorization purposes. The secure conversation session mapped by the identifier defined in the SCT will be destroyed.

After the token is canceled, the "Build RSTR SOAP Response Assertion" on page 288 will create a response message containing a `<wst:RequestedTokenCanceled/>`, similar to the following:

```
<wst:RequestSecurityTokenResponse>
  <wst:RequestedTokenCanceled/>
</wst:RequestSecurityTokenResponse>
```

Canceling Outbound Session

Outbound sessions are established using the [Establish Outbound Secure Conversation](#) assertion. You simply need to specify the URL of the session being canceled.

Note: Canceling an outbound session will also cancel the inbound session, if both sessions are the same. However if the inbound session is not available (for example, it has already been canceled), this assertion will not fail.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Cancel Security Context to <service URL>** in the policy window and select **Security Context Cancellation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

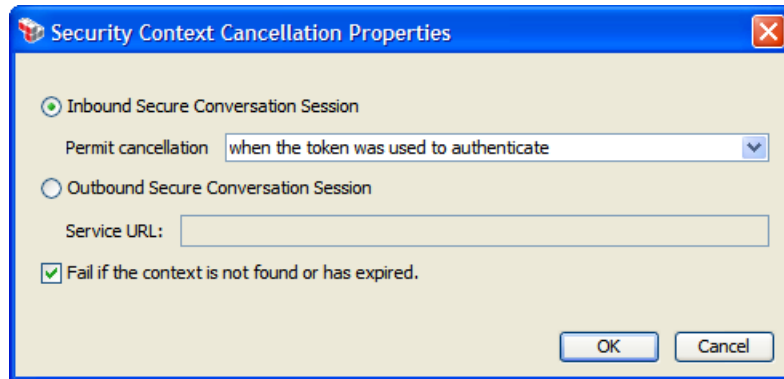


Figure 96: Security Context Cancellation Properties

3. Configure the properties as follows.

Table 82: Security Context Cancellation settings

Setting	Description
Inbound Secure Conversation Session	Select this option to cancel an inbound secure conversation session.
Permit cancellation	<p>If canceling an inbound session, choose when cancellation is possible:</p> <ul style="list-style-type: none"> • always: There is no permission check—the token can always be canceled. When this option is selected, this assertion does not require an authenticated user. • when the token is owned by an authenticated user: Only the authenticated user has the right to cancel the token. • when the token was used to authenticate: Cancellation is possible only if the security token was used to authenticate the session. This setting is the default.
Outbound Secure Conversation Session	Select this option to cancel an outbound secure conversation session. Note that the inbound session will also be canceled if both sessions are the same session.
Service URL	Enter the URL of the service that created the security token.
Fail if the context is not found or has expired	<p>Select this check box to indicate that the assertion will fail if the secure conversation session does not exist or has expired.</p> <p>Clear this check box to allow the assertion to succeed even if the context is not found or is expired.</p>

4. Click **[OK]**.

Configure WS-Security Decoration Assertion

The *Configure WS-Security Decoration* assertion is used to specify or override pending security decorations for a message. This assertion provides a convenient location to set security attributes that were previously configured in other assertions:

Table 83: WS-Security decorations in Configure WS-Security Decoration assertion

Decoration	In tab	Originally set in...
WS-Security Version	[General]	"Add or Remove WS-Security Assertion" on page 273
Signature Digest Algorithm	[Signing]	"Sign Element Assertion" on page 407
Encryption Algorithm(s)	[Encryption]	"Encrypt Element Assertion" on page 346
Signature Key Reference	[Signing]	"Sign Element Assertion" on page 407
Encryption Key Reference	[Signing]	"Encrypt Element Assertion" on page 346
Add Timestamp	[General]	"Add Timestamp Assertion" on page 283
Security Token Signing	[Signing]	"Sign Element Assertion" on page 407
Key Encryption Algorithm	[Encryption]	new option
Use DerivedKey Token	[Advanced]	new option

Tips: (1) Decorations that were originally set elsewhere can continue to be configured in those other assertions as well. (2) The *Configure WS-Security Decoration* assertion can be used to remove the timestamp in the policy: add it after the *Add Security Token* assertion and configure the properties to clear the **Add Timestamp** check boxes in the [\[General\]](#) and [\[Signing\]](#) tabs.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about selecting a private key for this assertion, see *Selecting a Custom Private Key* in the *Layer 7 Policy Manager User Manual*.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Applying WS-Security

If this assertion targets a message other than the response, you must add the [Add or Remove WS-Security](#) assertion after the *Configure WS-Security Decoration* assertion in the policy for the decorations to be applied:

```
Request: Configure WS-Security Decoration
Request: Apply WS-Security
```

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Tip: When WS-Security is involved, be sure to specify the appropriate WSS header handling option in the routing assertion's properties. In most instances, the setting "*Don't modify the request Security header*" is usually appropriate.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Configure WS-Security Decoration** in the policy window and select **Configure WS-Security Decoration Properties** or double-click the assertion in the policy window. The assertion properties are displayed. These properties are organized across the following tabs:

General

Signing

Encryption

Advanced

3. Configure each tab as necessary. Refer to the appropriate section below for a complete description of each tab.

Tip: The "<Unchanged>" setting found in several of the tabs leaves the existing *pending decoration requirement* setting unchanged—this is not necessarily the same as leaving *existing decoration* in the message unchanged. **Example:** Suppose a message is currently signed using SHA-384. One of the pending decorations is a new signature that uses the default digest (which is SHA-1, as specified by the cluster property `wss.decorator.digsig.messagedigest`). When "<Unchanged>" is selected in the [Signing] tab for Signature Digest Algorithm, the new signature will use SHA-1, not SHA-384.

4. Click **[OK]** when done.

Configuring the [General] Tab

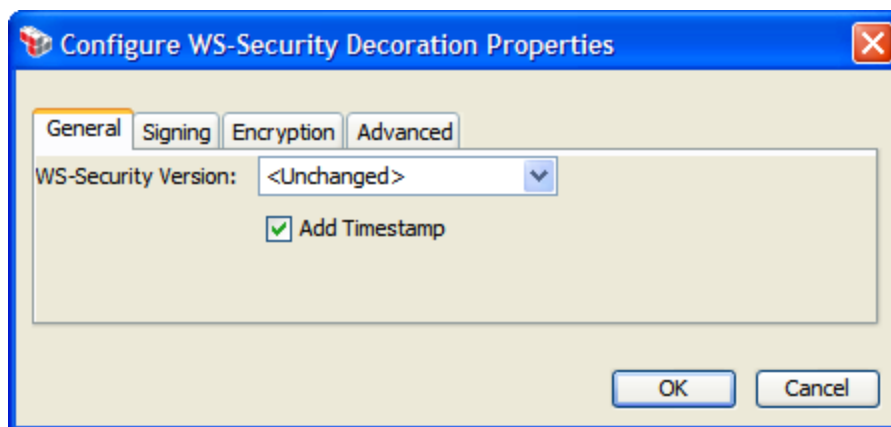


Figure 97: Configure WS-Security Decoration Properties - [General] tab

The **[General]** tab is used to request a specific version of WS-Security or to force a timestamp to be included.

- **WS-Security Version:** Choose the WS-Security version to use: **1.0** or **1.1**. The default "<Unchanged>" setting uses the version of WS-Security in the target message.

The WS-Security version can also be set in the "Add or Remove WS-Security Assertion" on page 273.

- **Add Timestamp:** Use this check box to add or remove a time stamp on the target message.

Tips: (1) Timestamps can also be added (but not removed) using the "Add Timestamp Assertion" on page 283. (2) An existing timestamp will only be removed if the **[Remove and recreate matching security header]** setting in the "Add or Remove WS-Security Assertion" on page 273 is selected (which is the default).

Configuring the [Signing] Tab

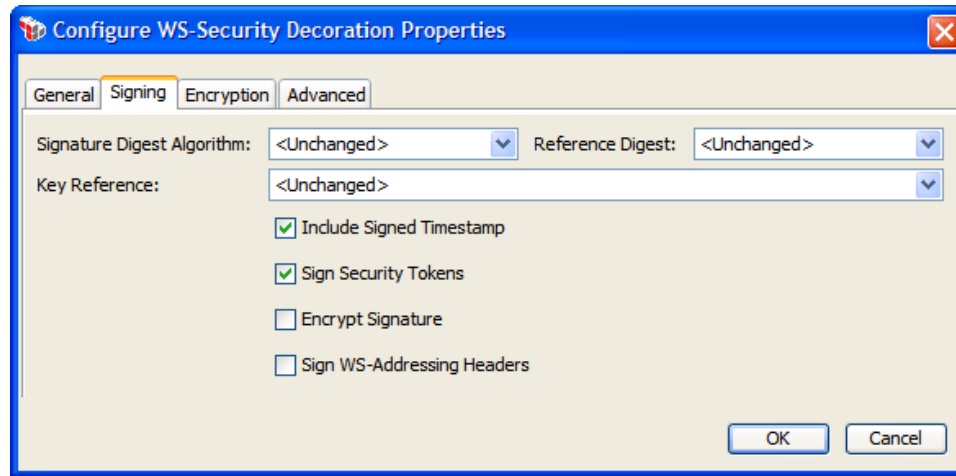


Figure 98: Configure WS-Security Decoration Properties - [Signing] tab

The [Signing] tab is used to set signing-related decorations.

- **Signature Digest Algorithm:** Choose the Signature Digest Algorithm to use: **SHA-1, SHA-256, SHA-384, SHA-512**. The default "<Unchanged>" setting uses the algorithm in the target message's existing decoration requirements. (See the tip under ["Using the Assertion"](#) above for more information.)

The Signature Digest Algorithm can also be set in the "Sign Element Assertion" on page 407.

- **Reference Digest:** Choose the Reference Digest to use: **SHA-1, SHA-256, SHA-384, SHA-512**. The default "<Unchanged>" setting uses the algorithm in the target message's existing decoration requirements. (See the tip under ["Using the Assertion"](#) above for more information.)
- **Key Reference:** Choose the signing key reference mechanism to use: **BinarySecurityToken, SubjectKeyIdentifier, IssuerSerial**. The default "<Unchanged>" setting uses the key reference in the target message's existing decoration requirements. (See the tip under ["Using the Assertion"](#) above for more information.)
- **Include Signed Timestamp:** Select this check box to add a signed timestamp to SOAP header of the target message.

Tips: (1) Signed timestamps can also be added (but not removed) using the "Add Timestamp Assertion" on page 283. (2) If **[Include Signed Timestamp]** is disabled, but **[Add Timestamp]** in the [General] tab is enabled, then an unsigned timestamp will be added to the target message.

- **Sign Security Tokens:** Select this check box to request that signing tokens be included in the message signature.
- **Encrypt Signature:** Select this check box to include the signature in the elements to encrypt, marked as requiring whole-element encryption.
- **Sign WS-Addressing Headers:** Select this check box to sign the WS-Addressing headers and any CA addressing headers present in the message. **Note:** This option must be enabled to ensure compatibility when the SecureSpan XML VPN Client is used.

Configuring the [Encryption] Tab

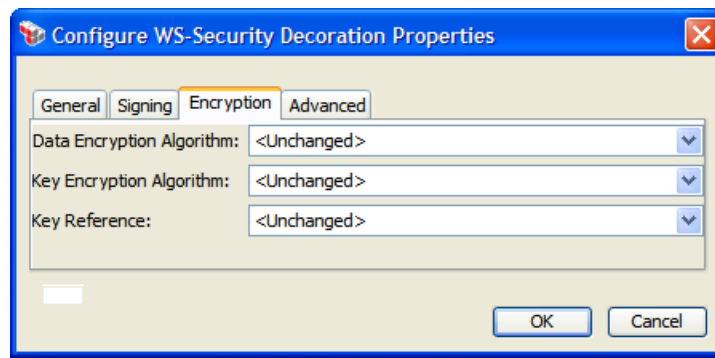


Figure 99: Configure WS-Security Decoration Properties - [Signing] tab

The **[Encryption]** tab is used to configure encryption-related decorations.

- **Data Encryption Algorithm:** Choose a symmetric encryption algorithm to use for data encryption. The default "**<Unchanged>**" setting uses the algorithm in the target message's pending decoration requirements. (See the tip under "[Using the Assertion](#)" above for more information.)

<http://www.w3.org/2001/04/xmlenc#tripledes-cbc>

<http://www.w3.org/2001/04/xmlenc#aes128-cbc>

<http://www.w3.org/2001/04/xmlenc#aes192-cbc>

<http://www.w3.org/2001/04/xmlenc#aes256-cbc>

<http://www.w3.org/2009/xmlenc11#aes128-gcm>

<http://www.w3.org/2009/xmlenc11#aes256-gcm>

The data encryption algorithm can also be set in the "Encrypt Element Assertion" on page 346.

Tip: CA strongly recommends using one of the GCM algorithms when possible (assuming that the expected recipient can handle it). XML messages encrypted using CBC mode could potentially be decrypted by an adversary who can send multiple modified messages to servers that possess the decryption key and are running WS-Security software released before November 2011.

- **Key Encryption Algorithm:**

Key Encryption Algorithms are public key encryption algorithms especially specified for encrypting and decrypting keys when adding an encrypted key to a message during security decoration. Their identifiers appear as Algorithm attributes to EncryptionMethod elements that are children of EncryptedKey. The following is an example of EncryptedKey with a Key Encryption Algorithm:

```
<EncryptedKey Id="uuid...">
  <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
    oaep-mgf1p"/>
  ...
</EncryptedKey>
```

Choose a key encryption algorithm to use. The default "**<Unchanged>**" setting uses the algorithm in the target message's pending decoration requirements. (See the tip under "[Using the Assertion](#)" above for more information.)

http://www.w3.org/2001/04/xmlenc#rsa-1_5 (RSA Version 1.5 Identifier)

<http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p> (RSA-OAEP Identifier)

Tip: CA strongly recommends using RSA-OAEP whenever possible. Use RSA-1.5 only when absolutely necessary for interoperability.

More information about RSA Identifiers

RSA 1.5 is the "RSAES-PKCS1-v1_5" algorithm specified in RFC 3447 (aka PKCS#1):

<http://www.ietf.org/rfc/rfc3447.txt>

RSA OAEP is the "RSAES-OAEP-ENCRYPT" algorithm, also specified in RFC 3447/PKCS#1. This is an implementation of OAEP, the "optimal asymmetric encryption padding" scheme.

RSA OAEP is more resistant to certain cryptographic attacks than RSA 1.5 but may be supported by less third-party software.

- **Key Reference:** Choose a encryption key reference mechanism to use: **IssuerSerial**, **BinarySecurityToken**, **KeyName**, or **SubjectKeyIdentifier**. The default "**<Unchanged>**" setting uses the mechanism in the target message's pending decoration requirements. (See the tip under "[Using the Assertion](#)" above for more information.)

The encryption key reference can also be set in the "Encrypt Element Assertion" on page 346.

Configuring the [Advanced] Tab

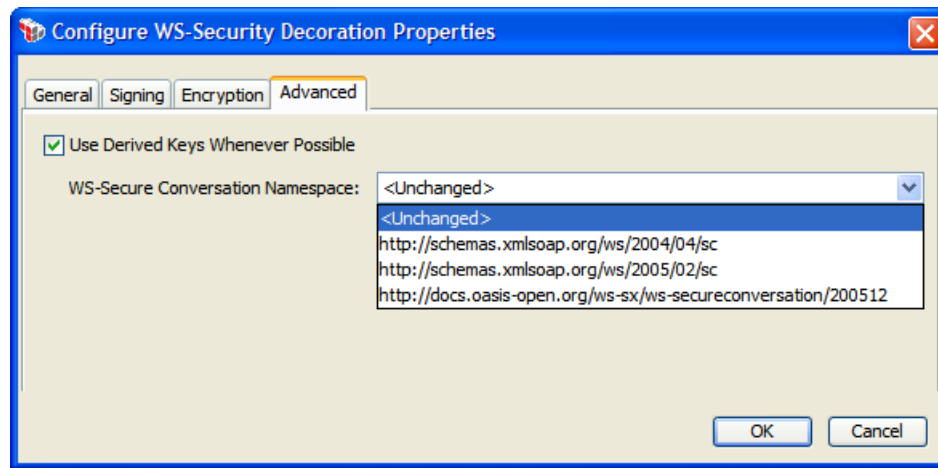


Figure 100: Configure WS-Security Decoration Properties - [Advanced] tab

The **[Advanced]** tab is used to configure derived keys and to optionally select a WS-Secure Conversation Namespace.

- **Use Derived Keys Whenever Possible:** Select this check box to request that derived keys be created and used whenever possible (that is, derived from the original signing or encryption token using a `DerivedKeyToken`). Derived keys can be used for signing or encryption if the signing or encryption method is a WS-SC session, an ephemeral `EncryptedKey`, a Kerberos token, or a SAML token that uses `EncryptedKey` subject confirmation.
- **WS-Secure Conversation Namespace:** If using derived keys, you can choose the version or namespace of secure conversation that will be used with the derived keys. The default value "**<Unchanged>**" will not configure any value, which results in the default/current namespace being used.

Create SAML Token Assertion

The *Create SAML Token Assertion* can create and optionally sign a SAML token. Examples of when this might be useful include:

- You need to create ad-hoc token services (i.e., receive a WS-Trust request, validate it, authenticate and authorize, and then issue a SAML token for the response)
- You are currently using the "Attach SAML Sender-Vouches" option in the [Route via HTTP\(S\)](#) assertion ([Security] tab), but you need a more configurable option.
- You are using a transport like FTP that does not presently include an option to add a SAML sender-vouches token.

The SAML token that is created is stored in the `${issuedSamlAssertion}` context variable. This variable is made available to the "Build RSTR SOAP Response Assertion" on page 288 to create an RSTR response message. For more information, see Working with the Security Token Service in the *Layer 7 Policy Manager User Manual*.

The following is an example of a SAML token in the `${issuedSamlAssertion}` variable:

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
MinorVersion="1" MajorVersion="1"
AssertionID="SamlAssertion-87a72d52cf2716824ccb036c03f17fca"
Issuer="gateway.acmecorp.com"
IssueInstant="2010-08-17T23:01:10.215Z"><saml:Conditions
NotBefore="2010-08-17T22:56:10.000Z"
NotOnOrAfter="2010-08-17T23:06:10.216Z"><saml:AudienceRestrictionCondition>
<saml:Audience>https://saml.salesforce.com</saml:Audience>
</saml:AudienceRestrictionCondition></saml:Conditions><saml:AuthenticationStatement
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
AuthenticationInstant="2010-08-17T23:01:10.215Z"><saml:Subject><saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
NameQualifier="">jsmith@acmecorp.com.sso</saml:NameIdentifier>

<saml:SubjectConfirmation><saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:be
arer

</saml:ConfirmationMethod></saml:SubjectConfirmation></saml:Subject><saml:SubjectLoc
ality
IPAddress="10.0.12.345"/></saml:AuthenticationStatement></saml:Assertion>
```

To learn more about selecting a private key for this assertion, see Selecting a Custom Private Key in the *Layer 7 Policy Manager User Manual*.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Context Variables Created by This Assertion

The Create SAML Token assertion sets the following context variables. **Note:** The default `<prefix>` is "attrStatement" and can be changed in Step 6 of the "SAML Token Creation Wizard" on page 317.

Table 84: Context variables created by Create SAML Token assertion

Context variable	Type	Notes
<code><prefix>.missingAttrNames</code>	String	Stores a list of the missing attributes (comma separated). This variable is empty when no attributes are missing.
<code><prefix>.unknownAttrNames</code>	String	Stores a list of the unknown filter attributes (comma separated). This variable is empty when no attributes are unknown
<code><prefix>.noAttributes</code>	Boolean	Returns "true" when all configured attributes were filtered, otherwise returns "false".

Context variable	Type	Notes
<prefix> filteredAttributes	String	Stores a list of the filtered attributes (comma separated). This variable is empty when no attributes were filtered.
<prefix> .excludedAttributes	String	Stores a list of the excluded attributes (comma separated). This variable is empty when no attributes were excluded . Note: This variable applies to SAML 2.0 only.

Adding and Configuring the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click **Create <token type> SAML Token** in the policy window and select **SAML Token Creation Wizard** or double-click the assertion in the policy window.
- Follow the wizard to complete the assertion. For details, see "SAML Token Creation Wizard" on page 317.

SAML Token Creation Wizard

The *SAML Token Creation Wizard* automatically starts when you add or modify the [Create SAML Token](#) assertion in a policy.

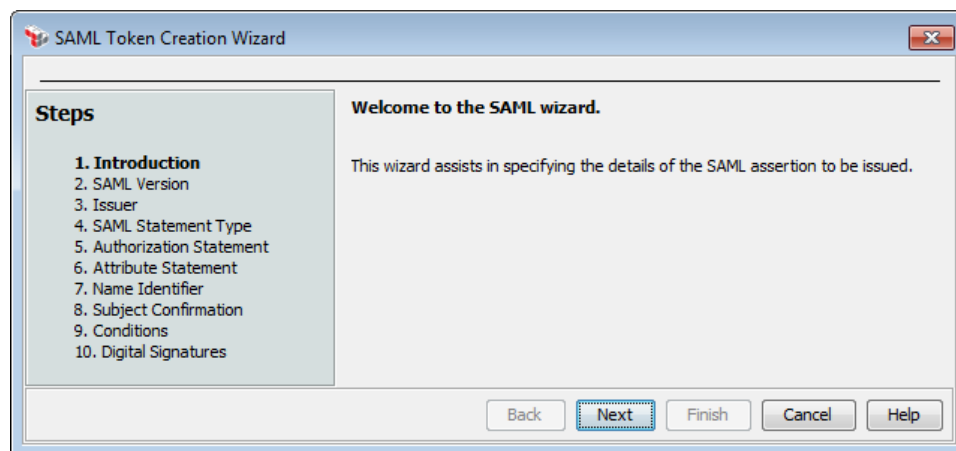







Figure 101: SAML Token Creation Wizard

For more information about wizards, see "Wizard" under Interfaces in the *Layer 7 Policy Manager User Manual*.

Table 85: Using the SAML Token Creation Wizard

Wizard Step	Description
Step 1: Introduction	Introduces the wizard.
Step 2: SAML Version	Specify the version of the SAML token to be issued: 1.1 or 2.0 .
Step 3: Issuer	<p>Configure the Issuer attribute value. The settings differ depending on the SAML version.</p> <p>SAML 1.1</p> <ul style="list-style-type: none"> • Default: Select this to use the subject DN from public key that corresponds to the configured private key. • From Template: Select this to customize the Issuer attribute. <p>You may reference context variables. </p> <p>SAML 2.0</p> <p>For a description of these settings, see "Configuring the [Issuer] Tab" in "Build SAML Protocol Response Assertion" on page 299.</p>
Step 4: SAML Statement Type	<p>Select at least one SAML statement to issue:</p> <ul style="list-style-type: none"> • Authentication Statement: This statement asserts that the subject authenticated with the identity provider at a particular time, using a particular method of authentication. • Authorization Decision Statement: This statement asserts that a subject is permitted to perform a specified action on a specified resource. • Attribute Statement: This statement is used to populate the SAML statement with specified attributes pertaining to the subject. • Include Authentication Context Declaration (SAML 2.0 only): This statement will include an Authentication Context Declaration, if possible. Specifically, this means the generated <i>AuthnStatement</i> or <i>AuthnContext</i> will contain an <i>AuthnContextDecl</i> child. If this check box is not selected, then only a <i>AuthnContextClassRef</i> child is present. <p>Note: The <i>AuthnContextDecl</i> element may not be present for all credential types, even if this option is enabled. This element should be present for password or X.509 credentials.</p> <p>The wizard will lead you through the appropriate steps based on the statements selected.</p>
Step 5: Authorization Statement 	<p>Specify the details for the Authorization Statement:</p> <ul style="list-style-type: none"> • Resource: Enter a value for the resource that the SAML statement must describe (for example, "http://acme.org"). • Action: Enter an action value for the resource (for example, "GET"). • Action Namespace: Optionally enter a corresponding action

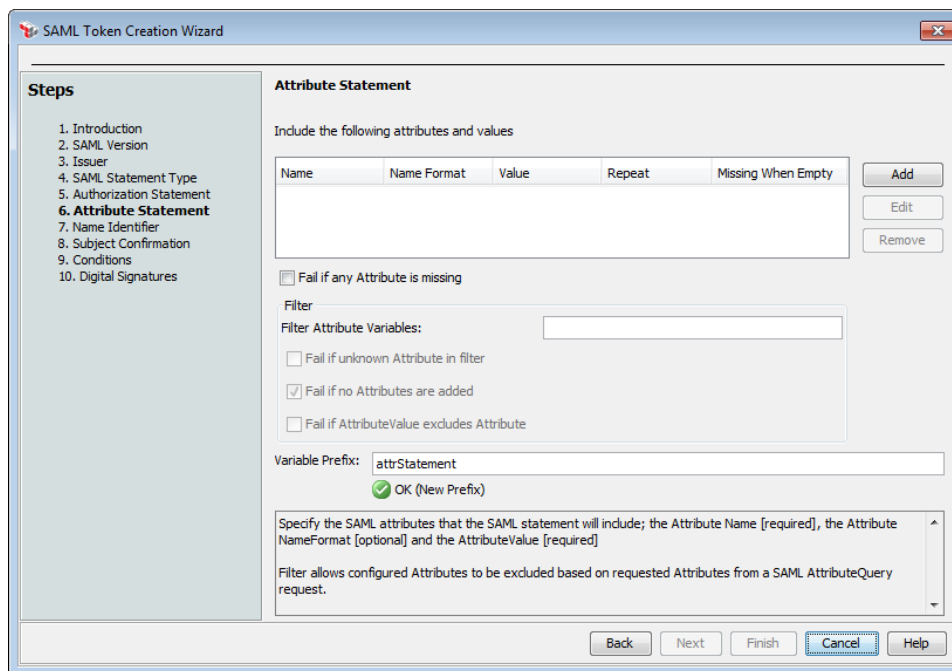
Wizard Step	Description
	namespace value (for example, "http://acme.org/ns/services").
Step 6: Attribute Statement 	<p>Define one or more SAML attributes that will be included in the SAML statement.</p> <p>See "Configuring the Attribute Statement" below for details.</p>
Step 7: Name Identifier 	<p>Enter the details for the SAML Authentication Statement:</p> <ul style="list-style-type: none"> • Name Identifier: Select the check box to include the Name Identifier in the SAML token. If you choose to not include the Name Identifier, clear the check box and then click [Next] to proceed to the next step. • Format: Specify the format of the Name Identifier: <ul style="list-style-type: none"> • Automatic: The Name Identifier Format URI will be selected based on the type of credentials used to authenticate the user. • Unspecified: Indicates that the issuer of the assertion is not warranting that the Name Identifier value meets any particular format expectations. • Any other: The Name Identifier Format URI is selected based on the option chosen. • Name Qualifier: Optionally enter a Name Qualifier template. This value determines the security or administrative domain of the subject. An example of a Name Qualifier might be the Gateway hostname (e.g., <i>gatewayhost.acmecorp.com</i>). It is not necessary to enter a fully-qualified hostname. • Name Identifier Value: Specify where the value of the Name Identifier is to be retrieved: <ul style="list-style-type: none"> • From Credentials: The value is the user name from the credentials used to authenticate the user. • From Authenticated User: The value is the most appropriate attribute (matching the selected Format) available from the user who was authenticated. • From Template: The value is the result of evaluating the specified template. This will typically be a context variable, perhaps one resulting from an XPath (Evaluate Request XPath or Evaluate Response XPath) or from the Extract Attributes for Authenticated User assertion.
Step 8: Subject Confirmation	<p>Configure the subject confirmation method in this step.</p> <p>See "Configuring the Subject Confirmation" below for details.</p>
Step 9: Conditions 	<p>Select one of the following options to restrict the validity period of the issued token to a limited time:</p> <ul style="list-style-type: none"> • Use Default Validity Period Condition: Select this option to

Wizard Step	Description
	<p>use the validity period conditions defined in the following cluster properties. This setting is the default:</p> <p><i>samlAssertion.NotBeforeOffsetMinutes</i> <i>samlAssertion.NotAfterOffsetMinutes</i></p> <ul style="list-style-type: none"> • Customize Validity Period Condition: Select this option to set a custom validity period for this assertion only: <ul style="list-style-type: none"> • Not Before seconds in past: The recipient should reject the token if its current local time is earlier than the token's <i>NotBefore</i> time. This value sets the <i>NotBefore</i> time to the current time on the Gateway minus this number of seconds. <p>This is useful for deployments with known time synchronization issues (for example, two machines that need to communicate with SAML have different system clocks).</p> • Not On Or After seconds in future: The recipient should reject the token if its current local time is later than the token's <i>NotAfter</i> time. This value sets the <i>NotAfter</i> time to the current time on the Gateway plus this number of seconds. • Audience Restriction: Optionally specify any restrictions on the audience for the SAML token. You may specify one or more constraints, separated by a space. The constraints may be static strings or context variables (either single- or multi-valued). The variable values may themselves contain a space-separated list of strings. <p>All strings that resolve to a valid URI will be added as separate <i>saml:Audience</i> elements in the SAML token.</p>
Step 10: Digital Signatures	<p>In this step, you specify the digital signatures that the Gateway should create (if any) after the SAML token is issued.</p> <ul style="list-style-type: none"> • Sign Assertion with an Enveloped Signature: If selected, the Gateway will include an XML Digital Signature within the issued SAML token, allowing it to be used outside the context of the current request or response. This option is mainly useful in situations where the SAML token itself is the focus of the interaction (for example, in token service policies). • Insert Assertion into Security header in request/response: This option is mainly useful for Sender Vouches. If selected, the issued SAML token will be added to the SOAP Security Header in either the Request or Response. In addition to the SAML token, selecting either of the following options will cause a message-level Signature to be created and added to the Security header as well: <ul style="list-style-type: none"> • If Include Assertion in Message-level Signature is selected, the issued SAML token will be included in the Signature. • If Include SOAP Body in Message-level Signature is

Wizard Step	Description
	selected, the SOAP Body element will be included in the Signature.

Configuring the Attribute Statement

This wizard step defines one or more SAML attributes that will be included in the SAML statement.



SAML Token Creation Wizard

Steps

1. Introduction
2. SAML Version
3. Issuer
4. SAML Statement Type
5. Authorization Statement
- 6. Attribute Statement**
7. Name Identifier
8. Subject Confirmation
9. Conditions
10. Digital Signatures

Attribute Statement

Include the following attributes and values

Name	Name Format	Value	Repeat	Missing When Empty

☐ Fail if any Attribute is missing

Filter

Filter Attribute Variables:

☐ Fail if unknown Attribute in filter

☒ Fail if no Attributes are added

☐ Fail if AttributeValue excludes Attribute

Variable Prefix:

☒ OK (New Prefix)

Specify the SAML attributes that the SAML statement will include; the Attribute Name [required], the Attribute NameFormat [optional] and the AttributeValue [required]

Filter allows configured Attributes to be excluded based on requested Attributes from a SAML AttributeQuery request.

Figure 102: SAML Token Creation Wizard - Step 6: Attribute Statement (SAML 2.x version shown)

1. Configure the attributes for the Attribute Statement:
 - To add an attribute, click **[Add]** and then complete Figure 103.
 - To modify an attribute, select it from the list, click **[Edit]** and then complete Figure 103.
 - To remove an attribute, select it from the list, and then click **[Remove]**.

Figure 103: Edit SAML Attribute Properties dialog (SAML 2.x version shown)

Complete the settings as follows:

Table 86: SAML Attribute Properties settings

Setting	Description
Attribute Name	Enter the name of the attribute.
Attribute Namespace (SAML 1.x only)	Optionally enter a namespace for the attribute.
Attribute Name Format (SAML 2.x only)	<p>Optionally specify a URI reference that describes the format of the attribute name. Only attributes that declare this format will be accepted.</p> <ul style="list-style-type: none"> • Unspecified: If no name format is provided, the default value of <code>urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified</code> is used. • URI Reference: This option uses the URI <code>urn:oasis:names:tc:SAML:2.0:attrname-format:uri</code>. • Basic: This option uses the URI: <code>urn:oasis:names:tc:SAML:2.0:attrname-format:basic</code>.


Setting	Description
	<ul style="list-style-type: none"> • Other: Select this option to define your own attribute name format in the box below.
Attribute Value	Optionally enter a value for the attribute.
If Message or Element variable referenced	<p>Configure how to add the contents of the context variable to the Attribute element:</p> <ul style="list-style-type: none"> • Convert to string: The contents will be converted to a string. This setting is the default. • Add as XML fragment: This adds the XML contents of the variable to the <i>saml:AttributeValue</i>.
If variable not found	<p>Configure the behavior when a context variable is not found:</p> <ul style="list-style-type: none"> • Replace variable with empty string: This uses an empty string in place of the variable. This setting is the default. • Replace expression with empty string: This replaces the entire expression with an empty string.
If value resolves to empty string	<p>Configure empty attribute value behaviour:</p> <ul style="list-style-type: none"> • Add empty AttributeValue: This adds an empty <i><saml:AttributeValue /></i>. This setting is the default. • Do not add AttributeValue: This adds the Attribute without any <i><AttributeValue></i> element. • Add null value AttributeValue: This adds a null <i><saml:AttributeValue xsi:nil="true" /></i>.
Attribute Value Comparison (SAML 2.x only)	<p>The requestor of a SAML Protocol Attribute Query service may supply values for a requested Attribute, in this case if the attribute is returned in the response it must not contain any values not equal to the values specified in the query. Configure how incoming AttributeValue elements should be compared for an Attribute:</p> <ul style="list-style-type: none"> • String comparison: The values are compared as strings; no processing is done to the values before comparison. • Canonicalize: The values are canonicalized first. This option should be selected if the values contain XML. <p>Note: When the runtime value for an attribute is multivalued, then only values matching an incoming attribute value will be added.</p>
Missing when empty string	<ul style="list-style-type: none"> • Select this check box to treat a resolved empty string as "missing". This allows the Attribute Statement configuration to fail the assertion if an attribute's value cannot be resolved successfully. • Clear this check box to never interpret a resolved empty string as missing. <p>Tip: If you need to ensure that a referenced variable is successfully resolved at runtime, set "If variable not found" to Replace expression</p>

Setting	Description
	<p>with empty string, then select the Missing when empty string check box is selected. This can be used to fail the assertion. For example, you use LDAP to resolve context variables. However, a variable was not set because either the LDAP attribute does not exist or does not have a value. Using the settings outlined above, the AttributeValue can be declared as "missing" and the Attribute Statement configuration may choose to fail the assertion.</p>
Repeat if Multivalued	<p>Select this check box to expand multivalued context variables into multiple <code><saml:Attribute></code> values.</p> <p><i>Example:</i></p> <p>When [Repeat if Multivalued] is selected, a context variable containing the values ["first", "second"] will result in the following attributes:</p> <pre><saml:Attribute AttributeName="myVar" AttributeNamespace="urn:example.com:attributes"> <saml:AttributeValue>first</saml:AttributeValue> </saml:Attribute> <saml:Attribute AttributeName="myVar" AttributeNamespace="urn:example.com:attributes"> <saml:AttributeValue>second</saml:AttributeValue> </saml:Attribute></pre> <p>Conversely, if the Repeat if Multivalued check box is <i>not</i> selected, the values from the above context variable will be concatenated:</p> <pre><saml:Attribute AttributeName="myVar" AttributeNamespace="urn:example.com:attributes"> <saml:AttributeValue>first, second</saml:AttributeValue> </saml:Attribute></pre> <p>Note: The Repeat if Multivalued check box is unavailable if more than one variable reference is entered into the Attribute Value field or if only a single element is referenced within a multivalued variable. The following are some examples:</p> <ul style="list-style-type: none"> • If a single context variable is entered, the check box can be selected since the variable may be multivalued. • If there is any mixture of variable references and text or other variables, the check box cannot be selected.

- Configure the **Fail if any Attribute is missing** check box as required:
 - Select this check box to fail the assertion if the value of an Attribute is missing. If the assertion fails, this populates the context variable `<prefix>.missingAttrNames` with a list (comma separated) of attribute names.

Exception: When attribute filtering is enabled, this option only fails the assertion when the attribute requested has a missing value.
 - Clear this check box to allow missing Attributes without failing the assertion.
- Configure the **Filter** panel in the Attribute Statement as follows:

Table 87: Filter options in the Attribute StatementTable 87

Setting	Description
Filter Attribute Variables 	<p>Enter a context variable (single or multivalued) of type Element or Message that will contain <i>saml:Attribute</i> value(s). Any other values cause a warning audit but will not fail the assertion. If supplied then only Attributes included in this variable from the list of configured Attributes will be added to the Attribute Statement.</p> <p>Note: For SAML 2.0, the variables must be of type <i>saml:Attribute</i>. For SAML 1.1 they must be of type <i>AttributeDesignator</i>.</p>
Fail if unknown Attribute in filter	<p>Select this check box to fail the assertion if the request contains an unsupported attribute. If the assertion fails, this populates the context variable <i><prefix>.unknownAttrNames</i> with a list (comma separated) of the unknown attribute names.</p> <p>Clear this check box to allow a SAML Token to be issued when there is an unknown attribute requested.</p>
Fail if no Attributes are added	<p>Select this check box to fail the assertion if either the assertion is not configured with any Attributes contained in the Filter Attribute Variables or if the values of the incoming Attributes caused configured Attributes to be filtered. This will populate the context variable <i><prefix>.noAttributes</i> with true. By default, this check box is selected to comply with SAML core.</p> <p>Clear this check box to allow an empty AttributeStatement to be created.</p>
Fail if AttributeValue excludes Attribute <i>(SAML 2.0 only)</i>	<p>Select this check box to fail the assertion if:</p> <ul style="list-style-type: none"> an Attribute in the context variable contains one or more AttributeValue elements <p>AND</p> <ul style="list-style-type: none"> the resolved value(s) of the Attribute in this dialog at runtime does not contain any of the incoming value(s) <p>This populates the context variable <i><prefix>.excludedAttributes</i> with a list (comma separated) of the excluded attributes.</p> <p>Clear this check box to not fail the assertion under the above conditions.</p> <p>Note: This setting does not apply to SAML v1.1, as an AttributeQuery in v1.1 may not include AttributeValue elements.</p>

- Enter a prefix that will be added to the *<prefix>.missingAttrNames* variable and to the variables references in Table 87 above. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy. The default prefix is **attrStatement**.

For an explanation of the validation messages displayed, see Context Variable Validation in the *Layer 7 Policy Manager User Manual*.

Configuring the Subject Confirmation

This wizard step configures the subject confirmation method to be used in the issued SAML token.

Figure 104: SAML Token Creation Wizard - Step 8: Subject Confirmation

1. Choose the **Subject Confirmation Method** to be used for the issued SAML token:

Holder-of-Key

The SAML token will use the *Holder-of-Key* subject confirmation method (with the standard URI *urn:oasis:names:tc:SAML:1.0:cm:holder-of-key* or *urn:oasis:names:tc:SAML:2.0:cm:holder-of-key*, depending on the selected SAML version in Step 2 of the wizard). For such assertions, the Gateway will require that the subject demonstrate possession of the private key corresponding to the public key in the Subject certificate.

The request Subject may use one of two methods to prove that they hold this key:

- The request includes at least one element covered by a valid [WSS message signature](#). The signing certificate will be used as the Subject certificate. Or,
- The request arrived over SSL/TLS with client certificate. The client certificate will be used as the Subject certificate.

Sender Vouches

The SAML token will use the *Sender Vouches* subject confirmation method (with the standard URI *urn:oasis:names:tc:SAML:1.0:cm:sender-vouches* or

urn:oasis:names:tc:SAML:2.0:cm:sender-vouches, depending on the selected SAML version in Step 2 of the wizard). For such assertions, the Gateway vouches for the verification of the subject.

Bearer

The SAML token will use the *Bearer Token* subject confirmation method (with the standard URI *urn:oasis:names:tc:SAML:1.0:cm:bearer* or *urn:oasis:names:tc:SAML:2.0:cm:bearer*, depending on the selected SAML version in Step 2 of the wizard). Like HTTP cookies, such assertions will always be assumed to belong to whatever message contains them, and the subject will be assumed to be the sender of the message.

None


The SAML token does not have a subject confirmation method.

2. Configure the **[Include Subject Certificate as]** check box as required. This is available on when the **Subject Confirmation Method** is "Holder-of-Key".



Select this check box to specify that the subject's certificate (or a reference to it) will be included in the SAML token. Choose the method by which it should be included or referenced from the drop-down list:

- **Literal Certificate (X509Data):** The entire subject certificate is inserted into the SAML token. This increases the size of the assertion significantly, but will mean that the recipient does not have to locate the subject certificates.
- **SecurityTokenReference using SKI:** A Subject Key Identifier (SKI) from the certificate is included in the SAML token. This may result in a smaller token, but it requires that the recipient look up the subject certificate.
- **SecurityTokenReference using SHA1 Thumbprint:** A SHA1 thumbprint from the certificate is included in the SAML token. Like the SKI option above, this may result in a smaller token, but it requires that the recipient look up the subject certificate.

Clear this check box to not include the subject's certificate (or reference to it) in the SAML token.

3. If SAML 2.0 is used and the Subject Confirmation Method is not set to "None", optionally complete the **Subject Confirmation Data** section. These fields provide additional information to be used by a specific confirmation method. You may reference context variables in any of these fields.
 - **Recipient:** Enter a URI that specifies the entity or location to which an attesting entity can present the token. For example, this attribute might indicate that the token must be delivered to a particular network endpoint in order to prevent an intermediary from redirecting it someplace else. You may reference context variables. 

Note: This must be set if configuring a SAML Web SSO profile.

- **Address:** Enter the network address or location from which an attesting entity can present the token. For example, this attribute might be used to bind the token to particular client addresses to prevent an attacker from stealing and presenting the token from another location. You may reference context variables. 
- **In Response To:** Enter the ID of a SAML protocol message in response to which an attesting entity can present the token. For example, this attribute might be used to correlate the token to a SAML request that resulted in its presentation. You may reference context variables. 

Note: This must be set if configuring a SAML Web SSO profile that was started with an *AuthnRequest*.

4. In the **Add Validity Period** section, you can optionally define a validity period for the SAML token:
 - **Not Before seconds in past:** Select this check box and then enter the number of seconds in the past before which the subject cannot be confirmed. The default is **120** seconds.
 - **Not On or After seconds in future:** Select this check box and then enter the number of seconds into the future after which the subject can no longer be confirmed. The default is **300** seconds. **Note:** This must be set if configuring a SAML Web SSO profile.

Create Security Context Token Assertion

The *Create Security Context Token* assertion is used to process an inbound message containing a RequestSecurityToken (RST) request. It will issue a Security Context Token (SCT), establish a secure conversation session, and then save the session. The secure conversation session is mapped by the identifier defined in the SCT.

Context Variable Created by This Assertion

The generated Security Context Token is stored in the `${<prefix>.issuedSCT}` context variable. This variable is made available to the "Build RSTR SOAP Response Assertion" on page 288 to create an RSTR response message. For more information, see *Working with the Security Token Service* in the *Layer 7 Policy Manager User Manual*.

The following is an example of an SCT in the `${issuedSCT}` context variable:

```
<sc:SecurityContextToken wsu:Id="uuid-86acfd31-dcaf-4b4f-9b45-8d79e3c63cba-64"
xmlns:sc="...">
  <sc:Identifier>urn:uuid:...</sc:Identifier>
</sc:SecurityContextToken>
```

This assertion assumes that credentials have been provided and are authenticated in the request.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153. The target message should contain a user's credentials for request authorization and security context creation.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Create Security Context Token** in the policy window and select **Security Context Token Creator Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

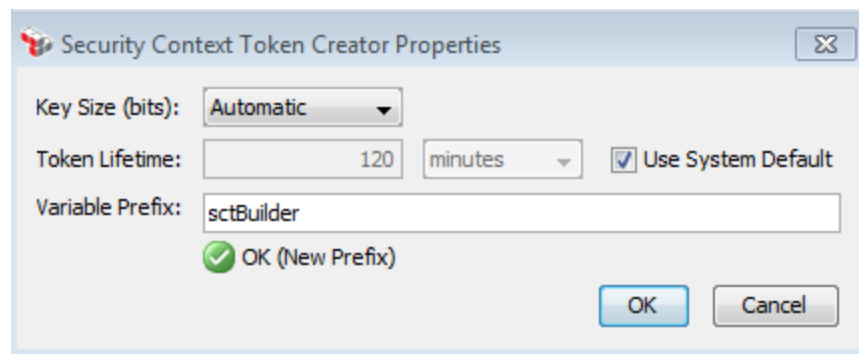


Figure 105: Security Context Token Creator Properties

3. Configure the properties as follows:

Table 88: Security Context Token Creator settings

Setting	Description
Key Size	<p>Select the minimum key size to use. If set to "Automatic", then the key size will be set to the same key size defined in the RST Request SOAP message, which is the target message set by this assertion. If set to "Automatic" and no key size is defined in the RST Request SOAP message, then the default key size 256 is used.</p> <p>If the request value is larger than the configured size, then the value from the request will be used.</p>
Token Lifetime	<p>Specify the length of time since issuing before the token expires. This defines the lifetime of a security context session. This setting is available</p>

Setting	Description
	only if you are not using the system default for token lifetime.
Use System Default	For the token lifetime, use the value defined in the cluster property <code>wss.secureConversation.defaultSessionDuration</code> . The default is 2 hours.
Variable Prefix	<p>Enter a prefix that will be added to the context variable created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>The default variable prefix is sctBuilder.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

4. Click **[OK]**.

Create XACML Request Assertion

The *Create XACML Request* assertion is used to build a valid XACML request and then place it in the specified target (request message, response message, or Message variable). The XACML request can then be used in the [Evaluate XACML Policy](#) assertion or it may be routed to any other PDP (Policy Decision Point) for a decision.

A XACML request is an XML fragment that conforms to the XACML (eXtensible Access Control Markup Language) specification.

A Request contains Attributes in each of the following four categories. By default, the Create XACML Request assertion will add each of these attributes under the root `<Request>` node. You may remove any that are not needed, provided that it's applicable to do so.

- **Subject:** There can be one or more Subject elements, and each can be identified by a category URI.
- **Resource:** XACML 1.0/1.1 must have exactly one Resource element in a request; XACML 2.0 may have more than one Resource element.
- **Action:** There must be exactly one Action element in a request.
- **Environment:** XACML 2.0 must have exactly one Environment element, while earlier versions can have 0 or 1 Environment element.

Note: You should be familiar with the XACML specification before using this assertion to construct a XACML request. For more information, see www.oasis-open.org.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the XACML Request Properties automatically appear; when modifying the assertion, right-click **Create XACML Request** in the policy window and select **XACML Request Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

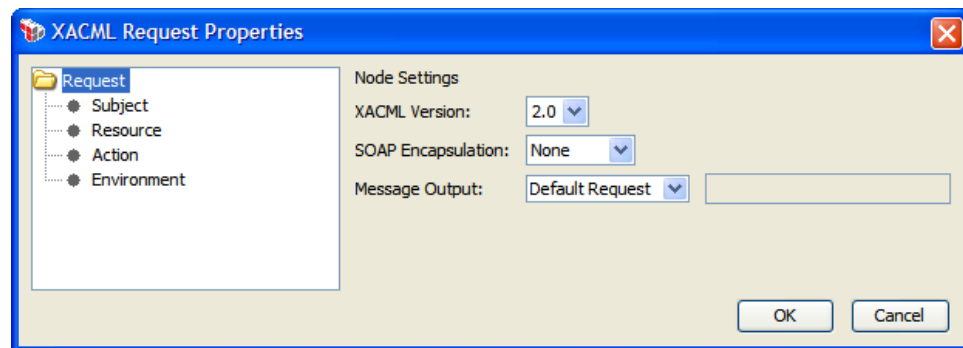



Figure 106: XACML Request Properties

3. Complete the basic settings for the Request:
 - **XACML Version:** Choose the XACML version that the generated XACML request will use. The options are: **1.0**, **1.1**, and **2.0**. (**Note:** The chosen version may affect the availability of certain options when building the request.)
 - **SOAP Encapsulation:** Choose the version of SOAP to use for encapsulation:
 - **SOAP 1.1:** The XACML <Request> element is contained in the Body element of a SOAP 1.1 envelope.
 - **SOAP 1.2:** The XACML <Request> element is contained in the Body element of a SOAP 1.2 envelope.
 - **None:** The XACML request is not enclosed in a SOAP envelope. The [Evaluate XACML Policy](#) assertion will not need to remove the SOAP envelope in order to use it.

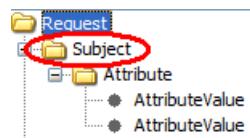
- **Message Output:** Choose where to place the resulting XACML request: In the **Request**, **Response**, or a **Message Variable** (context variable of type Message). If you choose to place it in a Message Variable, the variable does not need to exist beforehand—it will be created by this assertion. Do not enclose the variable with the "\${ }" characters. 
4. Configure each node as appropriate:

Subject
Resource
Action
Environment

Note: Context variables can be used in many settings when configuring a node. If during policy execution a referenced variable does not exist, the service policy will fail and a warning will be logged.

5. Click **[OK]** when done.

Configuring the Subject Node



The **Subject** node corresponds to the <Subject> element in a XACML request. Every XACML request must have at least one Subject node.

- To add a new Subject node, right-click on the Request root node and then select **Add Subject**.
- To remove an existing Subject node, right-click on the Subject node and then select **Remove Subject**. You cannot remove the last Subject node in a Request.

The Subject node has one setting:

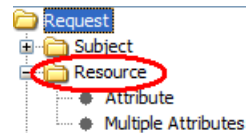
- **Subject Category:** This attribute describes the role that the Subject element plays in making the access request. If more than one Subject has the same Subject Category, then the [Evaluate XACML Policy](#) assertion will treat the contents of those Subject elements as if they were contained in the same Subject element. Select the Subject Category from the drop-down list of standard attributes (as presented by OASIS). This field is optional.

The Subject node can contain the following nodes:

[Attribute](#)

[Multiple Attributes](#)

Configuring the Resource Node



The **Resource** node corresponds to the <Resource> element in a XACML request. A XACML 1.0 or 1.1 request has exactly one Resource node. A XACML 2.0 request may have more than one Resource nodes.

- To add a new Resource node, right-click on the Request root node and then select **Add Resource**.
- To remove an existing Resource node, right-click on the Resource node and then select **Remove Resource**. You cannot remove the last Resource node in a Request.

The Resource node has no settings.

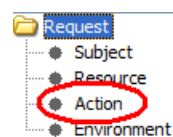
The Resource node can contain the following nodes:

[Resource Content](#)

[Attribute](#)

[Multiple Attributes](#)

Configuring the Action Node



The **Action** node corresponds to the <Action> element in a XACML request. Every XACML request has exactly one Action node.

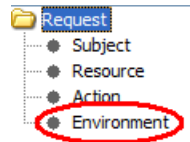
The Action node has no settings.

The Action node can contain the following nodes:

[Attribute](#)

[Multiple Attributes](#)

Configuring the Environment Node



The **Environment** node corresponds to the <Environment> element in a XACML request. In XACML 1.0 or 1.1 requests, the Environment node is optional and may be deleted if not required. In a XACML 2.0 request, there must be exactly one Environment node.

- To remove an Environment element from a XACML 1.0 or 1.1 request, right-click on the Environment node and then select **Remove Environment**.

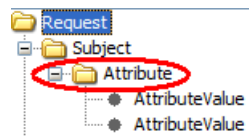
The Environment node has no settings.

The Environment node can contain the following nodes:

[Attribute](#)

[Multiple Attributes](#)

Configuring the Attribute Node



The Attribute node corresponds to the <Attribute> element in a XACML request. This node can be created under all the major nodes:

[Subject](#)

[Resource](#)

[Action](#)

[Environment](#)





One or more Attribute nodes may be created under any of the major nodes.

- To add an Attribute node, right-click a major node and then select **Add Attribute**.
- To remove an Attribute node, right-click the node and then select **Remove Attribute**.

The Attribute node has the following settings:

Tip: Context variables may be entered in all fields in the Attribute node.



- **AttributeID:** The AttributeId. Enter the value for the AttributeId or select from the drop-down list. This field is required. 
- **DataType:** The data type of the contents of the <AttributeValue> element. Enter the Data Type or select from the drop-down list. This field is required. 
- **Issuer:** This attribute specifies the Issuer. This field is optional. 
- **IssueInstant** (XACML 1.0 and 1.1 only): The date and time at which the attribute was issued. The Issue Instant must be one of the following: 
 - blank
 - A context variable that resolves to a valid timestamp, either using one of the built-in variables *gateway.time.local*, *gateway.time.utc*, *request.time.local*, *request.time.utc*, or a user-defined variable that contains a timestamp.
 - A manually entered date/time in the format: *yyyy-MM-dd 'T' HH:mm:ss [Z]*, where 'T' is a separator character and '[Z]' is the required time zone.

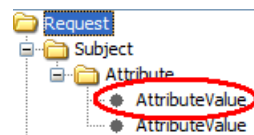
The Attribute node can contain the following number of **AttributeValue** nodes depending on the version of XACML:

XACML 1.0: 0 or 1

XACML 1.1: exactly 1

XACML 2.0: 1 or more

Configuring the AttributeValues Node



The AttributeValue node corresponds to the <AttributeValue> element. This node can only be created under the [Attribute](#) node.




There may be zero, one, or more AttributeValues for each Attribute node:

- To add an AttributeValue node, right-click an Attribute node and then select **Add Attribute Value**. Note that the number of Attribute nodes permitted depends on the version of XACML used (see above).
- To remove an AttributeValue from an Attribute, right-click the AttributeValue node and then select **Remove Attribute Value**.

Figure 107: AttributeValue node

An AttributeValue node has the following settings:

Table 89: AttributeValue node settings

Field	Description
Name 	<p>This is the name of an attribute to be placed into the <AttributeValue> Element, for example:</p> <p style="text-align: center;"><code><AttributeValue ATTR1="value"></code></p> <p>You may enter a context variable (of type String) that contains the name.</p>
Value 	<p>This is the value of an attribute in the <AttributeValue> element; for example:</p> <p style="text-align: center;"><code><AttributeValue att1="VALUE1"></code></p> <p>You may enter a context variable (of type String) that contains the value.</p>
AttributeValue Content 	<p>This is the text content of the <AttributeValue> element; for example:</p> <p style="text-align: center;"><code><AttributeValue>Sample Content</AttributeValue></code></p> <p>Type the content directly into the text box. You may enter a mixture of static text and context variables of type String, Message, or Element. (Variables of type Element are created by the Evaluate Request XPath and Evaluate Response XPath assertions in the ".elements" context variable.)</p> <p>Note: Do not type XML code into the text box. If you do, the XML will not appear correctly in the XACML request. If you wish to use XML in the <AttributeValue>, add the XML fragment to a context variable first.</p>

Field	Description
	For information on how to do this, see the "Set Context Variable Assertion" on page 656.
Repeat for multivalued variables <i>(XACML 2.0 only)</i>	<p>Select this check box to have the assertion automatically create separate <AttributeValue> elements if multivalued context variables are present. This feature is available only in XACML 2.0.</p> <p>For a detailed description of how this works, see "Working with Multivalued Context Variables in AttributeValues" below.</p>
Editing actions	<p>To add an AttributeValue Attribute:</p> <ol style="list-style-type: none"> 1. Click [Add]. 2. Enter the Name and Value. 3. Click [OK]. <p>To edit an AttributeValue Attribute:</p> <ol style="list-style-type: none"> 1. Select the row to edit. 2. Click [Edit]. 3. Modify the appropriate fields. 4. Click [OK]. <p>To remove an AttributeValue Attribute:</p> <ol style="list-style-type: none"> 1. Select the row to delete. 2. Click [Remove].

Working with Multivalue Context Variables in the AttributeValues Node

Context variables can be used in any of the AttributeValue node settings: Name, Value, Content. These variables may be either single-value or multivalued variables. How the system responds depends on the XACML version and whether the **Repeat for multivalued variables** check box is selected. The various outcomes are described in the following table:

Table 90: Effects of multivalued variables in AttributeValues

XACML Version	"Repeat" check box	Context variable encountered	Result
1.0 or 1.1	<not applicable>	Single-value	Creates a single <AttributeValue> element
1.0 or 1.1	<not applicable>	Multivalued	Concatenates the multiple values into a single value and then creates a single <AttributeValue> element. See " Concatenated Values " below for more details.
2.0	Not	Single-value	Creates a single <AttributeValue> element

XACML Version	"Repeat" check box	Context variable encountered	Result
	selected		
2.0	Not selected	Multivalued	Concatenates the multiple values into a single value and then creates a single <AttributeValue> element. See <i>"Concatenated Values"</i> below for more details.
2.0	Selected	Single-value	Creates a single <AttributeValue> element
2.0	Selected	Multivalued	Creates a series of <AttributeValue> elements for the XACML request. The number of elements created is equal to the context variable with the <i>fewest</i> values. See <i>"Multiple <AttributeValue> Elements"</i> below for a detailed description.

Indexed Single Value

When indexing is used to reference a single value within a multivalued context variable (for example, 'variable[0]' for the first value, 'variable[1]' for the second value, etc.), the resulting single value is treated the same as a static value or a single-value context variable (in other words, it will be repeated for each <AttributeValue> element generated).

For more information on indexing, see *Indexing Options* under Working with Multivalued Context Variables in the *Layer 7 Policy Manager User Manual*.

Concatenated Values

When concatenation occurs, the values from a multivalued context variable are combined into a single value, separated by a comma and a space. For example, the variable "\${multiVar}" contains the values *red*, *green*, and *blue*. The concatenated value, which is treated as a single value, will be:

```
red, green, blue
```

For more information, see *Concatenation Options* under Working with Multivalued Context Variables in the *Layer 7 Policy Manager User Manual*.

Multiple <AttributeValue> Elements

When the **Repeat for multivalued variables** check box is selected and a multivalued variable is encountered, a series of <AttributeValue> elements for the XACML request will be created. The number of elements created is equal to the multivalued context variable with the *fewest* values. For example, consider this example:

Name: \${ATTR1} contains the values ID1, ID2, ID3

Value: \${VALUE1} contains the values VALUE1, VALUE2, VALUE3

Content: \${CONTENT} contains the values CONTENT1, CONTENT2

Based on this example, two <AttributeValue> elements will be created because \${CONTENT} only has two values and this becomes the limiting factor:

```
<AttributeValue ID1="VALUE1">CONTENT1</AttributeValue>
<AttributeValue ID2="VALUE2">CONTENT2</AttributeValue>
```

In this example, the values ID3 and VALUE3 will not appear in any <AttributeValue> element. The audit log will record the fact that not all values were used in the building of the XACML request.

Note: If the AttributeValue node's dialog references more than one multi valued context variable, the context variable referenced with the fewest values from all the referenced variables becomes the limiting factor. For example: \${name1} has 5 values, \${value1} has 4 values, and \${content} has 2 values. This will result in only two <AttributeValue> elements being created.

Single-value variables vs. multivalued variables with one value

It is important to note the differences between a *single-value context variable* vs. a *multivalued context variable that contains only a single value*.

- A *single-value variable* will be treated the same as static text: it will be repeated for each <AttributeValue> generated. For example:

Name: \${attr1} contains the values ID1, ID2, ID3

Value: STATIC_VALUE

Content: \${content} contains the values CONTENT1, CONTENT2

Based on this example, two <AttributeValue> elements will be created (note the repetition of "STATIC_VALUE"):

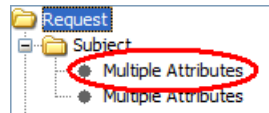
```
<AttributeValue ID1="STATIC_VALUE">CONTENT1</AttributeValue>
<AttributeValue ID2="STATIC_VALUE">CONTENT2</AttributeValue>
```

Note that ID3 is unused in this example.

- A *multivalued context variable containing a single value* will behave as expected: its single value becomes the limiting factor in the number of <AttributeValue> elements created.

For more information, see Working with Multivalued Context Variables in the *Layer 7 Policy Manager User Manual*.

Configuring the Multiple Attributes Node



Multiple Attributes is a special node that is designed to generate multiple <Attribute> elements in the XACML request, either by evaluating XPath expressions and/or multivalued context variables. For details on how multiple <Attribute> elements are dynamically generated, see "[How <Attribute> Elements are Dynamically Generated](#)" below.

One or more Multiple Attributes nodes may be created under any of the major nodes.

- To add an Multiple Attribute node, right-click a major node and then select **Add Multiple Attributes**.
- To remove an Multiple Attribute node, right-click the node and then select **Remove Multiple Attributes**.

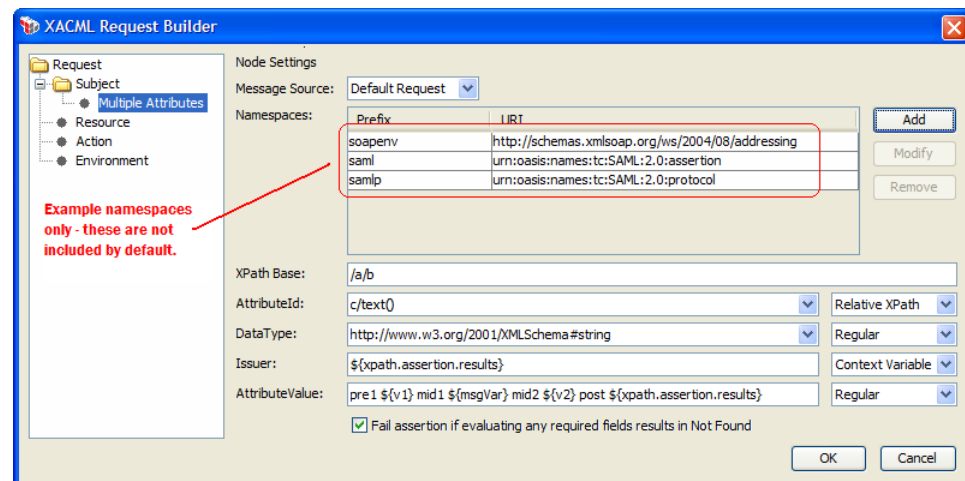




Figure 108: Multiple Attributes node (XACML 2.0 screen)


The Multiple Attribute node has the following settings:

Table 91: Multiple Attribute node settings

Field	Description
Message Source	Choose the message against which the XPath expressions will be evaluated.
Namespace Prefix /URI	These are the namespaces and prefixes that are used in any XPath expression.

Field	Description
XPath Base	<p>If any text field is set to 'Relative XPath', enter the base XPath expression here. When the XPath Base is referenced from a text field set to 'Relative XPath', the expression is evaluated and the number of results found is considered when determining how many <Attribute> elements to create.</p> <p>The Base XPath expression is executed if any field references it, including the AttributeValue field (except this field cannot define iteration).</p> <p>Absolute XPaths are evaluated against the Document and are independent of the base XPath.</p>
[Input type drop-down list] 	<p>The drop-down list next to the AttributeID, DataType, Issuer, IssueInstant, and AttributeValue fields dictates how the input in each field will be interpreted. It is important to select the correct option so that your input is processed correctly:</p> <ul style="list-style-type: none"> Regular: The input is interpreted as regular text. Context variables may be embedded within regular text and will be processed correctly. These context variables may be single value or multiple value. Fields of type 'Regular' will not be considered part of the logic for determining how many <Attribute> elements to create. <p>Notes: (1) For XPath expressions, be sure to select one of the 'XPath' settings. If left on the 'Regular' setting, an XPath expression will be interpreted as text, not as an XPath expression. (2) If you are attempting to access the main/root MIME part of a message using <code>\${variableName.mainpart}</code>, be sure to select the 'Regular' setting. Other settings will not interpret the mainpart correctly. (3) Using <code>".mainpart"</code> results in the variable being evaluated as a String, which will then be included as a text node in the XACML request. By comparison, if the variable is evaluated as a Message or an Element, it will be included as a XML fragment instead.</p> Absolute XPath: The input is interpreted as an absolute XPath expression, which will be evaluated directly against the Message Source document. <p>Notes: (1) When entering an absolute XPath expression in any field other than AttributeValue, you must append <code>"/text()"</code> to the end of the expression. (2) If the XPath expression evaluates to more than one node, only the first one is used. This event will be logged and audited at the INFO level.</p> Relative XPath: The input is interpreted as an XPath expression that is relative to the XPath Base expression specified above. If the XPath expression evaluates to more than one node, only the first one is used. <p>Note: When entering a relative XPath expression in any field other than AttributeValue, you must append <code>"/text()"</code> to the</p>

Field	Description
	<p>end of the expression.</p> <ul style="list-style-type: none"> • Context Variable: The input is a single context variable, with no surrounding text. The context variable may be either a single value or multivalued variable. If the variable is multivalued, then it will become part of the logic for determining how many <Attribute> elements are created. If the value is a single-value variable then it is the same as selecting 'Regular'. All fields accept context variables of type String. The AttributeValue field also accepts variables of type Message and Element. (Variables of type Element are created by the Evaluate Request XPath and Evaluate Response XPath assertions in the ".elements" context variable.) <p>Multivalued context variables will generate multiple XACML elements.</p> <p>Note: If you are attempting to access the main/root MIME part of a message using <code>\${variableName.mainpart}</code>, <u>do not</u> use the 'Context Variable' setting. Use the 'Regular' setting instead.</p>
AttributeID	The expression to use to find the value for the AttributeID attribute of the <Attribute> element. Either select an expression from the drop-down list or enter an expression in the field.
DataType	The expression to use to find the values for the DataType attribute of the <Attribute> element. Either select a value from the drop-down list or enter a value in the field.
Issuer	Enter the expression to use to find the values for the Issuer attribute of the <Attribute> element.
IssueInstant  (XACML 1.0 and 1.1 only)	<p>Enter the expression to use to find the values for the IssueInstant attribute of the <Attribute> element. This attribute specifies when the issuer issued this attribute. The Issue Instant may be one of the following:</p> <ul style="list-style-type: none"> • blank • A context variable that resolves to a valid timestamp, either using one of the built-in variables <i>gateway.time.local</i>, <i>gateway.time.utc</i>, <i>request.time.local</i>, <i>request.time.utc</i>, or a user-defined variable that contains a timestamp. • A manually entered date/time in the format: <code>yyyy-MM-dd'T'HH:mm:ss[Z]</code>, where 'T' is a separator character and '[Z]' is the optional time zone. • An XPath expression that returns the date/time in the above format. <p>Note: A context variable or XPath expression is not evaluated until run time, thus the policy validator will not warn you during design time about incorrect date formats. If a value for IssueInstant is provided and it is invalid, it will cause the assertion to fail.</p>

Field	Description
AttributeValue 	<p>Enter the expression to use to find the values for the <AttributeValue> element. Unlike the preceding three fields, this field can also accept context variables of type Message and Element.</p> <p>Tip: Context variables of type Element are created by the XPath assertion in the .elements variable.</p> <p>For XACML 1.0 or 1.1, only one <AttributeValue> element will be created. For XACML 2.0, multiple <AttributeValue> elements will be generated under either of these conditions:</p> <ul style="list-style-type: none"> • The 'AttributeValue' field is of type 'Context Variable' and a multivalued variable is specified. • The 'AttributeValue' field is of type 'Absolute XPath' or 'Relative XPath' that evaluates to a multi-node results. <p>If the AttributeValue could not be found, the <Attribute> element will be created with an empty <AttributeValue> element.</p>
Fail assertion if evaluating any required fields results in not found	<p>Select this check box if you want the assertion to fail if any of the following required elements could not be found during policy execution:</p> <p><i>AttributeID</i> <i>DataType</i> <i>IssueInstant</i> (if applicable and supplied)</p> <p>Clear this check box to ignore required elements that could not be found and continue generating <Attribute> elements. The assertion does not fail and the unfound elements are logged and audited at the INFO level and no <Attribute> element will be created for the current iteration.</p>
Editing actions	<p>To add a Namespace:</p> <ol style="list-style-type: none"> 1. Click [Add]. 2. Enter the Prefix and URI. 3. Click [OK]. <p>To edit a Namespace:</p> <ol style="list-style-type: none"> 1. Select the row to edit. 2. Click [Modify]. 3. Modify the appropriate fields. 4. Click [OK]. <p>To remove a Namespace:</p> <ol style="list-style-type: none"> 1. Select the row to delete. 2. Click [Remove]. The row is deleted without further confirmation.

How <Attribute> Elements are Dynamically Generated

The Multiple Attributes node will generate multiple <Attribute> elements under either of these conditions:

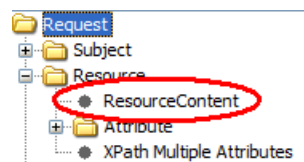
- If any of the **AttributeID**, **DataType**, **IssueInstant**, or **Issuer** fields is of type 'Context Variable' and a multivalued context variable is entered.
- If any of the **AttributeID**, **DataType**, **IssueInstant**, **Issuer**, or **AttributeValue** fields is of type 'Relative XPath' and the **XPath Base** evaluates to a multi-node result.

The number of <Attribute> elements created is based on the multivalued context variable with the *fewest* values and the number of XPath multi-node results (whichever is lower). To see an example of how multivalued context variables can be the constraining factor, see "Multiple <AttributeValue> Elements" under "[Working with Multivalued Context Variables in the AttributeValues Node](#)" above. Just remember that for the <Attribute> element, the XPath node results is an additional constraining factor.

It is possible to reference a single value within a multivalued context variable. For more information, see "Indexed Single Value" under "[Working with Multivalued Context Variables in the AttributeValues Node](#)" above.

To learn more about the differences between a single-value context variable and a multivalued variable that just happens to contain one value, see "Single-value variables vs. multivalued variables with one value" under "[Working with Multivalued Context Variables in the AttributeValues Node](#)" above.

Configuring the Resource Content Node



The Resource Content node corresponds to the <ResourceContent> element in the XACML request. This node may be optionally created under the <Resource> node.

- To add a ResourceContent node, right-click the Resource node and then select **Add Resource Content**.
- To remove the ResourceContent node, right-click the node and then select **Remove Resource Content**.

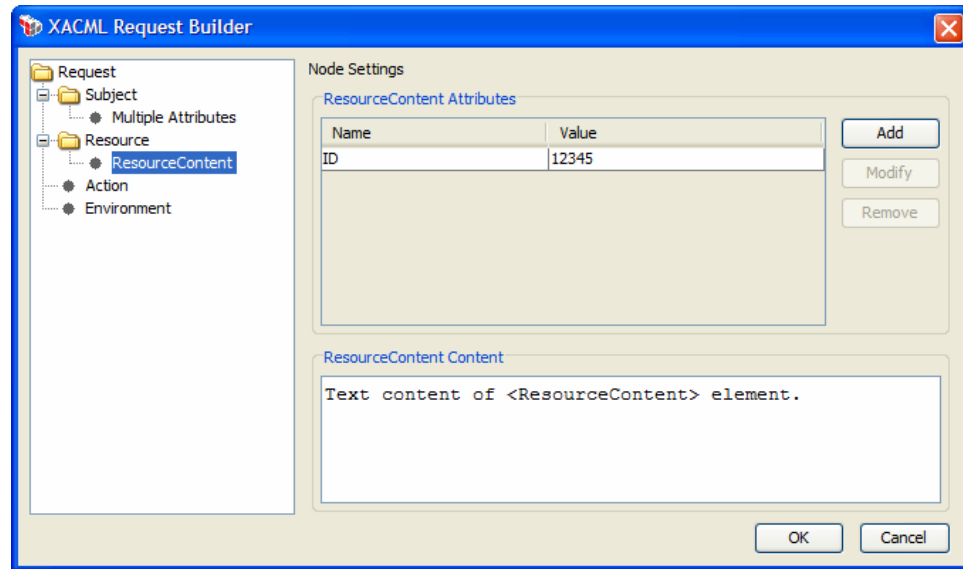



Figure 109: Resource Content node

The **Resource Content** node has the following settings:

Table 92: Resource Content node settings

Field	Description
Name	This is the name of an attribute to be placed into the <ResourceContent> Element, for example: <code><ResourceContent att1="value"></code> .
Value	This is the value of an attribute in the <ResourceContent> element; for example: <code><ResourceContent att1="value1"></code>
ResourceContent Content 	This is the text content of the <ResourceContent> element; for example: <code><ResourceContent>Sample text content</ResourceContent></code> Type the content directly into the text box. This text field supports single or multi valued variables of type String, Message, or Element.
Editing actions	To add a ResourceContent Attribute: <ol style="list-style-type: none"> 1. Click [Add]. 2. Enter the Name and Value. 3. Click [OK]. To edit a ResourceContent Attribute: <ol style="list-style-type: none"> 1. Select the row to edit. 2. Click [Modify].

Field	Description
	<ol style="list-style-type: none"> 3. Modify the appropriate fields. 4. Click [OK]. <p>To remove a ResourceContent Attribute:</p> <ol style="list-style-type: none"> 1. Select the row to delete. 2. Click [Remove].

Encrypt Element Assertion

The *Encrypt Element* assertion is used to select message elements to be encrypted in the [target message](#).

- If the target is the *response* message, encryption will occur automatically.
- If the target is the *request* message or a *message context variable*, then the [Add or Remove WS-Security](#) assertion must be added after the Encrypt Element assertion in the policy to perform the encryption.

You can add an Encrypt Element assertion for each element of the target message that you want encrypted. This assertion supports the W3C XML Signature 1.0 standard.

This assertion can only be used in a web service policy. It should be placed before the routing assertion in a policy.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about selecting the target identity for this assertion, see "Selecting a Target Identity" on page 152.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Note: When [multiple signatures](#) are used in a target message, it is mandatory to select a target identity.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.

2. Right-click the **<target>: Encrypt Element** in the policy window and select **Encrypt Element Properties** or double-click the assertion in the policy window. The assertion properties are displayed. The title of the dialog will show "Request", "Response", or "\${variableName}", depending on the target message.

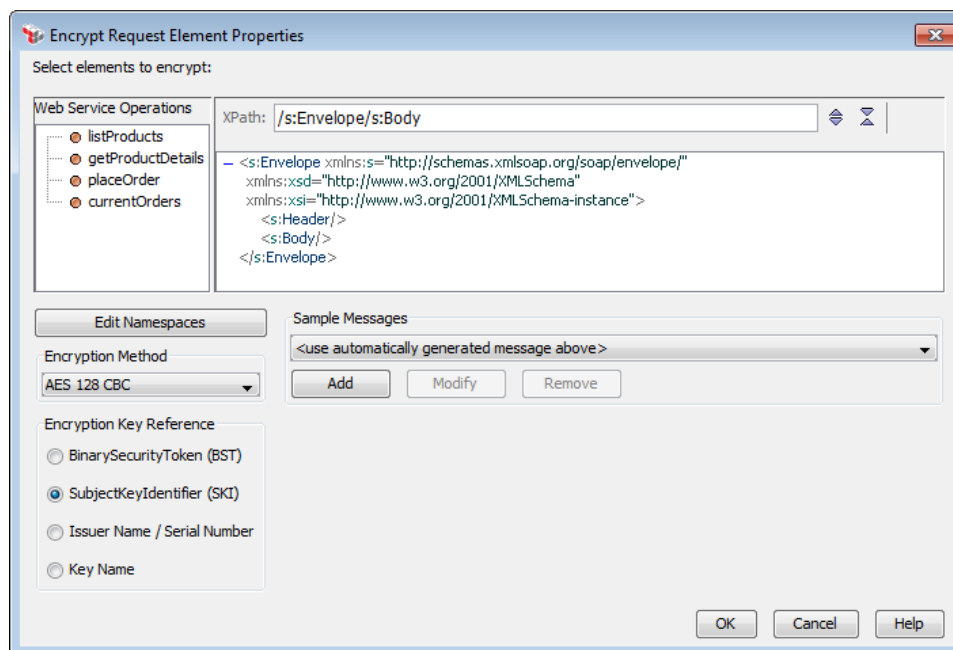


Figure 110: Encrypt Request Element Properties dialog

3. Specify the XPath and select the target element to be encrypted from the code box. For detailed instructions on using the interface to build your XPath, see "Selecting an XPath" on page 154.

The Policy Manager will not allow you to encrypt the `/soapenv:Envelope` element in the Encrypt Request Element Properties dialog. You can, however, encrypt a child element within the envelope such as `/soapenv:Envelope/soapenv:Body`.

Tip: A matching element's own opening and closing tags and tag attributes do not need to be encrypted. To force the encryption of an entire element—including opening and closing tags, attributes, and white space content—match the XPath expression to the parent element of the message. Clicking, or highlighting, an element selects it (and any child code) for the assertion encryption requirement.

4. Choose the **Encryption Method** from the drop-down list:

AES 128 CBC (default)
AES 192 CBC

AES 256 CBC
Triple DES
AES 128 GCM
AES 256 GCM

Note: The "AES-GCM" encryption options can be selected even if your security provider does not support it. However, this will result in encryption/decryption attempts to fail at runtime.

5. For **Encryption Key Reference**, select the method to use to include the SSL certificate for the Gateway:
 - **BinarySecurityToken (BST):** Use a SecurityTokenReference containing the BinarySecurityToken (BST).
 - **SubjectKeyIdentifier (SKI):** Use a SecurityTokenReference containing the SubjectKeyIdentifier (SKI).
 - **Issuer Name/Serial Number:** Use a SecurityTokenReference containing the certificates issuer distinguished name and serial number.
 - **Key Name:** Use a SecurityTokenReference containing the Key Name.

Notes: (1) Using a "Key Name" reference violates the WS-I Basic Security Profile so this reference type should be avoided whenever possible. (2) The "KeyName" element will be added inside a "SecurityTokenReference", e.g.,

```
<dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
  <wsse:SecurityTokenReference>
    <dsig:KeyName>CN=Bob,OU=OASIS Interop Test
    Cert,O=OASIS</dsig:KeyName>
  </wsse:SecurityTokenReference>
</dsig:KeyInfo>
```

6. Click **[OK]**.

Establish Outbound Secure Conversation Assertion

The *Establish Outbound Secure Conversation* assertion creates a new secure outbound conversation session using the security context identifier extracted from a Security Context Token. This outbound session includes a shared secret to be used for message decoration in future message exchanges.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Notes: (1) Outbound secure conversation sessions are stored for each distinct authenticated-user and service URL. (2) An existing outbound session will be overwritten if a new session is created for the same authenticated-user and service URL.

Context Variables Created by This Assertion

The Establish Outbound Secure Conversation assertion sets the following context variable for the session:

outboundSC.session

Attributes of the secure conversation session can be retrieved by using the following syntax:

outboundSC.session.<attribute>

For example, to access the session identifier, use **\${outboundSC.session.id}**.

The attributes are described in Table 93.

Table 93: Outbound secure conversation session attributes

Attribute	Description
id	The session identifier
user	The authenticated user
<i>To access specific attributes about the user, use the syntax:</i> outboundSC.session.user.<user_attribute>	
providerId	The user's Identity Provider ID
id	The user's identifier
login	The user's login ID
firstName	The user's first name
lastName	The user's last name
email	The user's email address
department	The user's department
subjectDn	The user's X.509 subject DN
creation	The session creation time
expiration	The session's expiration time
scNamespace	The namespace of WS-Secure Conversation

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Establish Outbound Secure Conversation to <service URL>** in the policy window and select **Outbound Secure Conversation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 111: Outbound Secure Conversation Properties

3. Configure the properties as follows.

Tip: You may use context variables in every field except for "Maximum Expiry Period" and "Security Context Token".

Table 94: Outbound Secure Conversation settings

Setting	Description
Service URL	Enter the URL of the service that issued the Security Context Token. The service URL will be used with the authenticated user information to create a mapping key to map the outbound secure conversation

Setting	Description
	<p>session that has been established.</p> <p>This field is disabled when the This session is for use with incoming request messages check box is selected (the service URL is not required in this scenario).</p> <p>Tip: If the authenticated user information is unavailable, the service URL should be specified by a unique string in order to create a unique session mapping key. (This string does not need to be a real URL, but it must be unique per secure conversation session.) The following is a sample string that creates a unique service URL:</p> <pre><service_url>?sessionId=<session_identifier></pre> <p>where <i><session_identifier></i> is the identifier of the outbound secure conversation session that has been established.</p>
Security Context Token	<p>Specify the name of the context variable from which to extract the session identifier. Enter this without the "\${}" wrapper.</p> <p>The default is rstrResponseProcess.token, which is defined in the "Process RSTR Response Assertion" on page 395.</p> <p>Note: This field requires that you specify the <i>actual name</i> of the context variable (i.e., the name without the "\${}" wrapper), not a name that is <i>resolved</i> from another variable. For example, "\${mySecurityToken}" is not permitted even if it contains the value "rstrResponseProcess.token".</p>
Client Entropy	<p>Specify the context variable containing the client entropy for creating a shared secret.</p> <p>The default is \${requestBuilder.clientEntropy}, which is defined in the "Build RST SOAP Request Assertion" on page 285.</p>
Server Entropy	<p>Specify the context variable containing the server entropy, if the RSTR response includes a server entropy.</p> <p>The default is \${rstrResponseProcessor.serverEntropy}, which is defined in the "Process RSTR Response Assertion" on page 395.</p>
Key Size	<p>Specify the context variable containing the key size, in bits, from the RSTR response. Enter "0" (zero) to use the default key size.</p> <p>The default value for this field is \${rstrResponseProcessor.keySize}, which is defined in the "Process RSTR Response Assertion" on page 395.</p>
Shared Secret	<p>Specify the context variable containing the shared secret, if the RSTR response includes a shared secret.</p> <p>The default is \${rstrResponseProcessor.fullKey}, which is defined in the "Process RSTR Response Assertion" on page 395.</p>
Session Lifetime	<p>Optionally set the lifetime of the secure conversation session.</p> <ul style="list-style-type: none"> • Create Time: Specify the context variable containing the

Setting	Description
	<p>creation time of the session in the server.</p> <p>The default is <code>\${rstrResponseProcessor.createTime}</code>, which is defined in the "Process RSTR Response Assertion" on page 395. If left blank, the current Gateway time is used.</p> <ul style="list-style-type: none"> • Expiry Time: Specify the context variable containing the expiry time of the session. <p>The default is <code>\${rstrResponseProcessor.expiryTime}</code>, which is defined in the "Process RSTR Response Assertion" on page 395.</p> <p>If the expiry time is left blank, the Gateway will use the following:</p> <p style="text-align: center;"><i>Current time + Maximum Expiry Period</i></p> <p>Tip: The cluster property <code>outbound.secureConversation.sessionPreExpiryAge</code> can be used to expire the assertion prior to the supplied expiry time; this offset can be adjusted to help prevent use of an expired session. For example, if the maximum expiry period is 20 minutes and the value of the cluster property is 5 minutes, the Gateway will use 15 minutes (20-5) as the final expiry period.</p>
Maximum Expiry Period	<p>Enter the maximum length of time for the session lifetime. A value of "0" (zero) means the original session expiry time is not limited.</p> <p>Note: The original session expiry time is defined as:</p> <p style="text-align: center;"><i>Expiry Time - Create Time</i></p>
Use System Default	<p>When specifying a token lifetime, select this check box to use the system default, as defined by the <code>outbound.secureConversation.defaultSessionDuration</code> cluster property. The default value for this property is 2 hours.</p>
[This session is for use with incoming request messages]	<p>Select this check box if this secure conversation session will be used with inbound request messages. This will allow the Security Token Service (STS) to "impersonate" that session user. Select this check box only if it is necessary to do so and only if the target STS is trusted.</p> <hr/> <p>WARNING: Sharing a secure conversation session for inbound and outbound traffic is not secure and is not recommended. Proceed only if you are an advanced user or if you are directed by CA Technical Support.</p> <hr/> <p>Clear this check box to not permit this session to be used with inbound secure conversation request messages. This setting is the default and the recommended setting.</p> <p>Note: When this option is enabled, you will not be able to access the</p>

Setting	Description
	session using the "Look Up Outbound Secure Conversation Session Assertion" on page 370. To use this session with outbound messages, use the "Require WS-Secure Conversation Assertion" on page 242 to validate the inbound session, then use the token from the <i>inboundSC.session</i> context variable with the "Add Security Token Assertion" on page 277 for outbound decoration.

- Click **[OK]**.


Evaluate SAML Protocol Response Assertion

The *Evaluate SAML Protocol Response* assertion is used to evaluate a SAML Protocol response. To create a SAML Protocol response, use the [Build SAML Protocol Response](#) assertion.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

The Evaluate SAML Protocol Response assertion is typically used as follows in a policy:

[Build SAML Protocol Request](#)
[Route via HTTP\(S\)](#)
Evaluate SAML Protocol Response

Tip: You can use context variables in many of the text fields in the wizard. These variables are evaluated at runtime as the SAML response is being constructed. 

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click **<target>: Evaluate SAML Protocol Response** in the policy window and select **SAML Protocol Response Wizard** or double-click the assertion in the policy window. The wizard appears.
- Follow the wizard to complete the assertion.

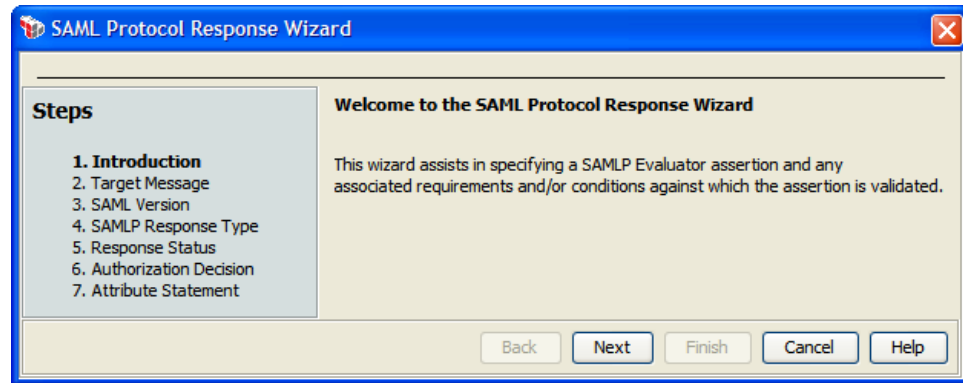



Figure 112: SAML Protocol Response Wizard

For more information about wizards, see "Wizard" under Interfaces in the *Layer 7 Policy Manager User Manual*.

Table 95: SAML Protocol Response Wizard settings

Wizard Step	Descriptions
Step 1: Introduction	Introduces the wizard.
Step 2: Target Message	Specify the location of the SAML response message for the evaluator to parse: Request , Response , or some Other Message Variable , with the default being "\${samlpResponse.message}". For more information on message type variables, see Context Variables in the <i>Layer 7 Policy Manager User Manual</i> . To learn how to change the message target, see "Selecting a Target Message" on page 153.
Step 3: SAML Version	Specify the version of the SAML response that will be evaluated by this assertion.
Step 4: SAML Response Type	Specify the type of SAML response being evaluated: <ul style="list-style-type: none"> • Authentication Request: The response contains authentication statements. Note: The Authentication Request option is available only when SAML 2.0 was selected in Step 3. • Authorization Decision Request: The response contains statements that assert a subject is permitted to perform a specified action on a specified resource. • Attribute Query Request: The response contains a list of attributes for the subject.
Step 5: Response Status	Indicate whether the Evaluate SAML Protocol Message assertion should fail if the response status could not be successfully retrieved. The system will always set the top level ResponseStatus onto the context variable <i>samlpResponse.status</i> .
Step 6: Authorization	This step is displayed only if "Authorization Decision Request" was selected in step 4.

Wizard Step	Descriptions
Validation	<p>Specify whether the assertion should fail based on the SAML response:</p> <ul style="list-style-type: none"> To never fail the assertion based a retrieved response, clear the Fail the assertion... check box. To fail the assertion unless the response matches your specified choice, select the Fail the assertion... check box and then choose a response from the drop-down list. The default is Permit, which means the assertion will fail unless the SAML response is 'Permit'. <p>The Authorization Decision Statement is stored in the context variable <i>samlpResponse.authz.decision</i>.</p>
Step 7: Attribute Statement 	<p>Specify the SAML attributes that the SAML statement <i>must</i> describe.</p> <ol style="list-style-type: none"> Click [Add] and then complete the Edit SAML Attribute Properties dialog: <ul style="list-style-type: none"> Attribute Name: Enter the name of the attribute. Attribute Namespace: Optionally enter a namespace for the attribute. This applies only to SAML 1.x. Attribute Name Format: Optionally specify a URI reference that describes the format of the attribute name. Only attributes that declare this format will be accepted. This applies only to SAML 2.x. <p>Unspecified: If no name format is provided, the default value of <i>urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified</i> is used.</p> <p>URI Reference: This option uses the URI <i>urn:oasis:names:tc:SAML:2.0:attrname-format:uri</i></p> <p>Basic: This option uses the URI <i>urn:oasis:names:tc:SAML:2.0:attrname-format:uri</i>.</p> <p>Other: Select this option to define your own attribute name format in the box below.</p> <ul style="list-style-type: none"> Attribute Friendly Name: Optionally enter a friendly name for the attribute to be used for display purposes. This applies only to SAML 2.x Attribute Value: If defining your own attribute name format, enter it here. Click [OK] to enter the attribute into the table. Repeat to configure additional attributes. <p>To modify an existing Attribute Statement, select it from the list and then click [Edit].</p> <p>To remove an Attribute Statement, select it from the list and then click [Remove].</p>

Evaluate XACML Policy Assertion

The *Evaluate XACML Policy* assertion evaluates a XACML policy and renders an authorization decision for a resource, which will be granted based on the set of attributes found in a XACML request. The XACML request can be created using the "Create XACML Request Assertion" on page 330.

The Evaluate XACML Policy assertion can retrieve the XACML request from a request or response message, or a context variable. After rendering the XACML request attributes against the XACML policy, the resulting decision can be placed either into the response message or a message context variable. The XACML policy can be configured in advance or the assertion can monitor a URL and download a new policy periodically.

Tip: The "XACML request" is also known as the "XACML decision request".

XACML Policy Validation

The Evaluate XACML Policy assertion will attempt to validate any XACML policy entered when the [OK] button is clicked. However, the assertion cannot validate a XACML policy under these circumstances:

- The invalid policy is retrieved from a monitored URL.
- The invalid policy (or policy fragment) is in a context variable that has been set prior to the Evaluate XACML Policy assertion.

In these cases, the invalid policy will be detected only during policy execution, with an error being logged.

Be aware that the Evaluate XACML Policy assertion will fail if you attempt to import a policy that contains an empty Description element, for example:

```
<Description/>
```

or

```
<Description></Description>
```

Note: You should be familiar with the XACML policy language and what constitutes a valid policy before using this assertion to process a XACML request.

Using the Assertion

1. Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the XACML Policy Properties automatically appear; when modifying the assertion, right-click **Evaluate XACML Policy Properties** in the policy window and select **XACML Policy Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

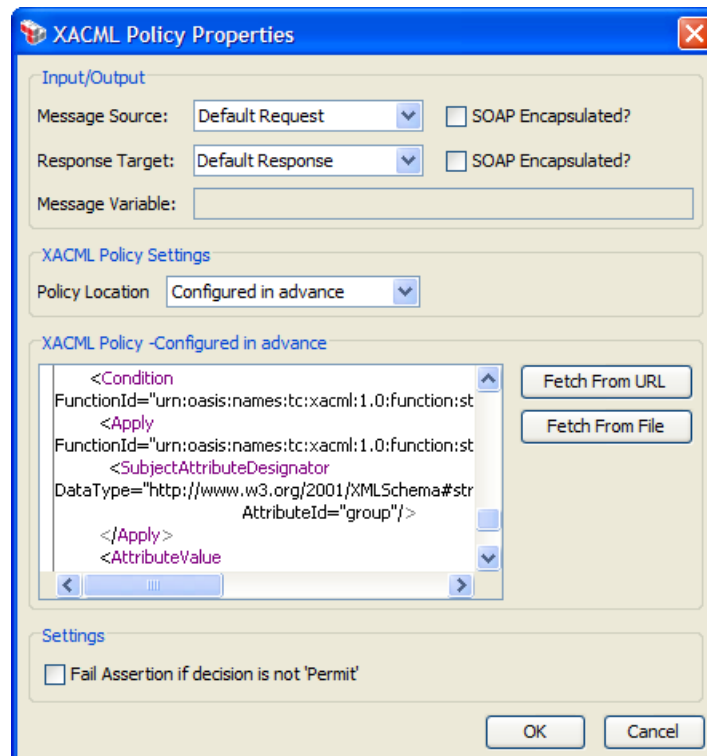







Figure 113: XACML Policy Properties

3. Configure the properties as follows:

Table 96: XACML Policy settings

Setting	Description
Message Source 	Indicate where to get the XACML request: Default Request , Default Response , or from a Context Variable that has been defined earlier in the policy (for example, using the Set Context Variable assertion). If using a context variable, it must be of type Message and its Content-Type should be 'text/xml'.
SOAP Encapsulated?	Select this check box if the XACML request is encapsulated in the Body element of a SOAP envelope. Note: If this check box is selected, the XACML request <u>must</u> be a SOAP

Setting	Description
(for message)	message, otherwise the assertion will fail.
Response Target 	Indicate where to place the XACML PDP response: in the Default Response , Default Request , or in a Message Variable (context variable of type Message). The variable does not need to exist beforehand—it will be created by this assertion. The Message Variable field is enabled if 'Message Variable' is selected.
SOAP Encapsulated? (for response)	Select this check box to encapsulate the XACML PDP response in the Body element of a SOAP envelope.
Message Variable 	This field is used only if the Response Target is 'Message Variable'. Enter the name of the variable to be created. You do not need to enclose the variable with the "\${ }" characters.
Policy Location	From the drop-down list, specify the location of the XACML policy: <ul style="list-style-type: none"> • Configure in advance: You are entering the policy in this assertion. • Monitor URL for latest value: The Gateway will obtain a policy from a URL that is monitored periodically for changes.
XACML Policy - Configure in advance 	<p>If you chose to configure the policy in advance, you have three different options:</p> <ol style="list-style-type: none"> 1. Type or copy/paste the XACML policy code directly into the code window. The policy code may contain context variables (see below). 2. Click [Fetch from URL] to enter an URL for the Gateway to retrieve the policy. <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> <p>Note: Unlike the "Monitor URL" option, the "fetch" option does not monitor the URL and the policy is only downloaded once at policy design time.</p> 3. Click [Fetch from File] to insert the policy code from a text file that you specify. <p>You may edit a policy entered using the "fetch" options if necessary.</p> <p>Note: The XACML policy maximum size is controlled by the <code>xacml.pdp.maxDownloadSize</code> cluster property.</p> <p>Using context variables in XACML policy code</p> <p>The XACML policy code may contain String values or context variables of type Message that contain XACML policy fragments or the entire XACML policy.</p> <p><i>Examples:</i></p> <ul style="list-style-type: none"> • <code>\${variableContainingPolicy}</code> is entered into the code window, where

Setting	Description
	<p>this previously defined context variable contains the entire XACML policy to use</p> <ul style="list-style-type: none"> A XACML policy is mostly configured in advance, with context variables representing policy fragments to be resolved at run time: <pre> <Policy...> <Target> ... </Target> <Rule...> \${stringVariable} </Rule> </Policy> </pre> <p>Here is another example of simple text replacement:</p> <pre> <Rule RuleId="ReadRule" Effect="\${permitDecision}" "> </pre> <p>where "\${permitDecision}" was previously defined with a value of "Permit".</p> <p>Note the following when using context variables in a XACML policy:</p> <ul style="list-style-type: none"> If the context variable contains an entire SOAP envelope, you must use an XPath expression to extract the policy itself (i.e., extract the "<Policy>" element) to a second context variable. For more information, see the "Evaluate Response XPath Assertion" on page 461. If a context variable of type Message contains the XACML policy as XML (either full policy or fragment), you must use the <code>\${variableName.mainpart}</code> part of the context variable otherwise the assertion will fail. <p>Note: For context variables of types other than Message, the <i>mainpart</i> part is not required. For more information on the various data types, see Context Variables in the <i>Layer 7 Policy Manager User Manual</i>.</p>
XACML Policy - Monitor URL for latest value	<p>If you chose to monitor a URL for the latest value, enter the URL here. The URL may contain context variables that will be resolved at run time. </p> <p>The time interval is set by the cluster property <code>xacml.pdp.policyCache.maxAge</code>. The default value for this cluster property is 300000 milliseconds (5 minutes). When the Evaluate XACML Policy assertion is processed within the policy, the policy is re-downloaded if the cached policy is older than the value of this cluster property.</p> <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> <p>Note: The XACML policy maximum size is controlled by the <code>xacml.pdp.maxDownloadSize</code> cluster property.</p>
Fail Assertion if decision is not 'Permit'	<p>A PDP response can be any of the following:</p> <p style="text-align: center;"><i>Permit</i> <i>Deny</i></p>

Setting	Description
	<p><i>Indeterminate</i> (an error occurred or some required value was missing, so a decision cannot be made)</p> <p><i>Not Applicable</i> (the request can't be answered by this service)</p> <p>Select this check box to fail the assertion if the PDP response is anything other than <i>Permit</i>. This is useful in scenarios where the policy acts as both a PEP (Policy Enforcement Point) and the PDP (Policy Decision Point). If the decision is not Permit, the policy can be configured to fail and you do not need to use an XPath expression to extract the result.</p> <p>Clear this check box to never fail the assertion, regardless of the PDP response.</p>

- Click **[OK]** when done.

Generate OAuth Signature Base String Assertion

The *Generate OAuth Signature Base String* assertion provides the ability to generate an OAuth signature base string according to the OAuth 1.0 specifications.

This assertion can be used in two distinct use cases:

- **OAuth Client:** An OAuth client policy contains the OAuth parameter values and these need to be input into the assertion.
- **OAuth Server:** An OAuth server policy receives a request that contains OAuth parameters. In this scenario, the assertion can automatically extract the required parameters from the message target.

This assertion is only used for OAuth 1.0.

Context Variables Created by This Assertion

The Generate OAuth Signature Base String assertion sets the following context variables.

Note: The default `<prefix>` is "oauth" and can be changed in the assertion properties (Figure 114 and Figure 115).

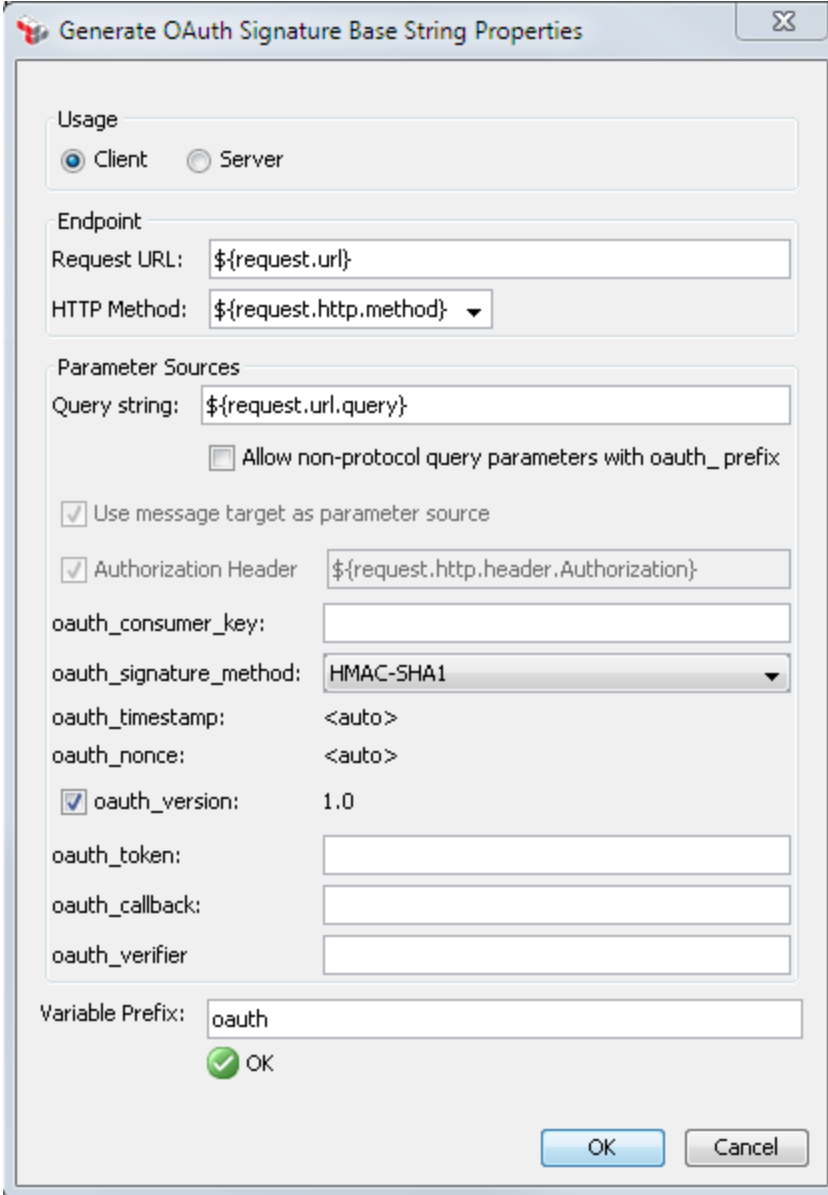
Table 97: Context variables created by Generate OAuth Signature Base String assertion

Variable	Description
<code><prefix>.sigBaseString</code>	The signature base string.
<code><prefix>.requestType</code>	Contains one of the following request types: <i>request token</i> , <i>authorized request token</i> , or <i>access token</i> .
<code><prefix>.authHeader</code>	Contains the partially completed authorization header.
<code><prefix>.<oauthParameter></code>	One variable will be created for each OAuth parameter.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **Generate OAuth Signature Base String Properties** automatically appears; when modifying the assertion, right-click **[Client|Server] Generate OAuth Signature Base String** in the policy window and select **Generate OAuth Signature Base String Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Note: All text fields in the properties dialog support expressions, except for Variable Prefix. This means a combination of text and context variables may be used and more than one variable may be referenced.



Generate OAuth Signature Base String Properties

Usage
☒ Client ☐ Server

Endpoint
 Request URL:
 HTTP Method:

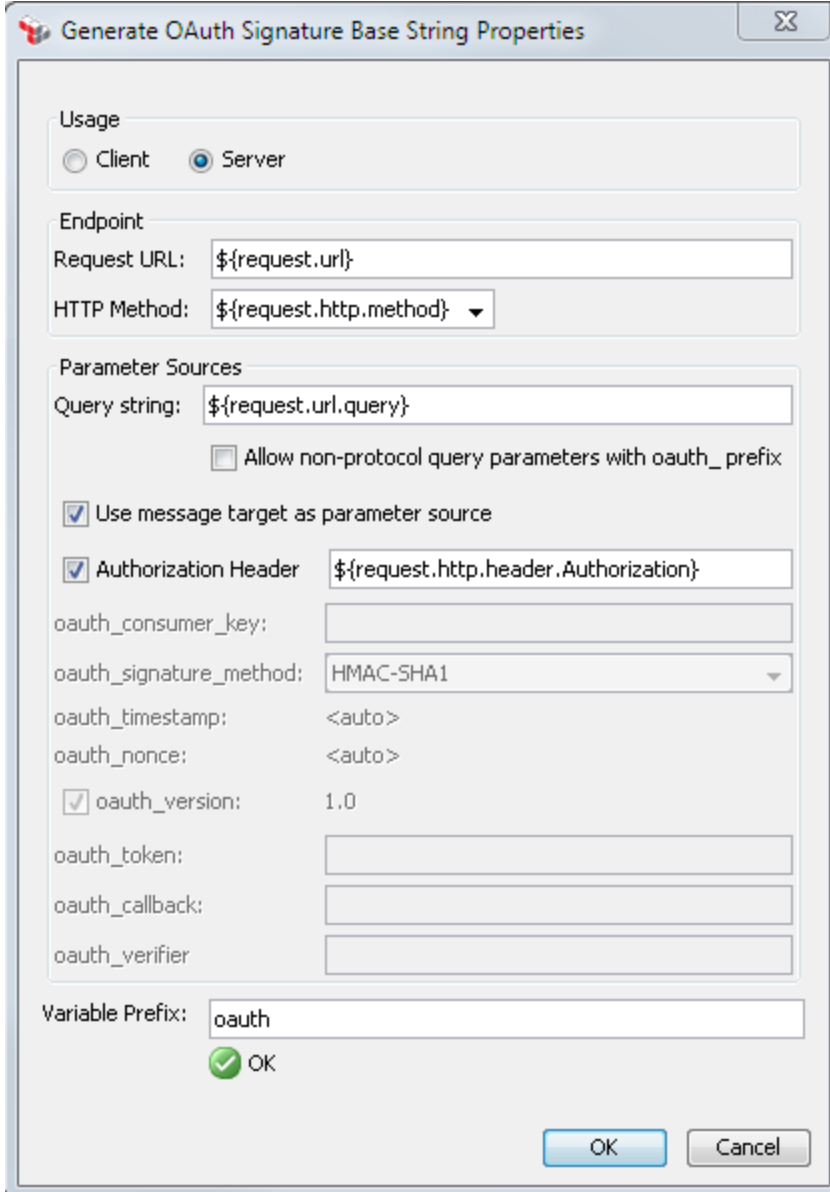
Parameter Sources
 Query string:
☐ Allow non-protocol query parameters with oauth_prefix
☒ Use message target as parameter source
☒ Authorization Header
 oauth_consumer_key:
 oauth_signature_method:
 oauth_timestamp:
 oauth_nonce:
☒ oauth_version:
 oauth_token:
 oauth_callback:
 oauth_verifier:

Variable Prefix:

☒ OK

OK Cancel

Figure 114: Generate OAuth Signature Base String Properties - Client mode



Generate OAuth Signature Base String Properties

Usage
☐ Client ☒ Server

Endpoint
 Request URL:
 HTTP Method:

Parameter Sources
 Query string:
☐ Allow non-protocol query parameters with oauth_prefix
☒ Use message target as parameter source
☒ Authorization Header
 oauth_consumer_key:
 oauth_signature_method:
 oauth_timestamp:
 oauth_nonce:
☒ oauth_version:
 oauth_token:
 oauth_callback:
 oauth_verifier:

Variable Prefix:

☒ OK





OK Cancel




Figure 115: Generate OAuth Signature Base String Properties - Server mode

3. Configure the properties as follows.

Table 98: Generate OAuth Signature Base String Properties settings

Setting	Description
Usage	<p>Select the mode for the assertion:</p> <ul style="list-style-type: none"> Client to send an OAuth request. Server to receive an OAuth request. In this case the assertion can be configured to automatically extract the required parameters from the message target, authorization header,

Setting	Description
	and request query string.
Endpoint 	<p>Configure the endpoint:</p> <ul style="list-style-type: none"> • Request URL: Enter the endpoint URL to which the OAuth request will be made. The default value is \${request.url}. • HTTP Method: Choose the method from the drop-down list. The default value is \${request.http.method}.
<i>Parameter Sources</i>	
Query String 	<p>Enter the query string, formatted according to the query portion of a valid URL. Name value pairs must be separated by the "&" character. The default is \${request.url.query}.</p> <p>Note: The value of \${request.url.query} is URL encoded. Any other value entered here must be URL-encoded to ensure this value is double-encoded when included in the generated signature base string.</p>
Allow non-protocol query parameters with oauth_ prefix	<p>Select this check box to permit non-protocol query parameters that are prefixed with "oauth_".</p> <p>Clear this check box to cause the assertion to fail if it encounters non-protocol query parameters prefixed with "oauth_". This setting is the default.</p>
Use message target as parameter choice (Server mode only)	Select this check box to allow the use of parameters extracted from a message target with a content type of <i>application/x-www-form-urlencoded</i> .
Authorization Header  (Server mode only)	Select this check box to allow parameters to be extracted from an Authorization Header. Default value is \${request.http.header.Authorization} .
oauth_consumer_key  (Client mode only)	Enter the OAuth consumer key.
oauth_signature_method (Client mode only)	Choose the OAuth signature method from the drop-down list. The default value is HMAC-SHA1 .
oauth_timestamp (Client mode only)	This value is set to <i><auto></i> , as it will be supplied at runtime.
oauth_nonce (Client mode only)	This value is set to <i><auto></i> , as it will be supplied at runtime.

Setting	Description
oauth_version (Client mode only)	Select this check box to include the oauth_version in the generated signature base string. Clear this check box to exclude the version from the string.
oauth_token  (Client mode only)	Enter an OAuth token, if necessary.
oauth_callback  (Client mode only)	Enter an OAuth callback value, if necessary.
oauth_verifier  (Client mode only)	Enter an OAuth verifier, if necessary.
Variable Prefix	Enter a prefix that will be added to the context variables created by this assertion. This prefix helps ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy. The default prefix is oauth . For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i> .

- Click **[OK]** when done.

Generate Security Hash Assertion

The *Generate Security Hash* assertion is used to generate a signature or hash with non-binary data using a configurable hash algorithm.

This assertion is configurable for specific HMAC+SHA algorithms or simply an SHA or MD5 algorithm.

Note: CA Technologies *highly* recommends using HMAC algorithms, as non-HMAC algorithms produce weak hashing that can be exploited.

Using the Assertion


- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.


2. This assertion contains default settings that are appropriate for most instances. To change any of the settings, right-click **Generate Security Hash** in the policy window and select **Generate Security Hash** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 116: Generate Security Hash Properties

3. Configure the properties as follows:

Table 99: Generate Security Hash settings

Setting	Description
Source Data 	Enter the data that will have the Signature Algorithm applied to it. You may enter text or specify a context variable that contains non-binary data. You may also enter an expression that combines static text with context variables.
Save line breaks as	Select the line break option for the source data entered above: CR LF (carriage return, line feed), LF (line feed), CR (carriage return). The default is CR LF .
Output Variable	Enter the name of the context variable that will hold the generated signature.
Signature Algorithm	Choose the algorithm to use from the drop-down list: HMAC-SHA1 HMAC-SHA256 HMAC-SHA384 HMAC-SHA512 MD5 SHA-1

Setting	Description
	SHA-256 SHA-384 SHA-512 Note the following: <ul style="list-style-type: none"> Choosing a HMAC algorithm will cause the assertion to produce a HMAC digest with the selected algorithm applied to the Source Data and Key. Choosing a non-HMAC algorithm will cause the assertion to produce a generic digest with the selected algorithm applied to the Source Data. Note: Use of non-HMAC algorithms is not recommended, as they produce weak hashing that can be exploited.
Key  <i>(HMAC algorithms only)</i>	Enter the key that will be used to generate the hash when a HMAC algorithm is selected. You may enter text or specify a context variable. You may also enter an expression that combines static text with context variables. This field is disabled when a non-HMAC algorithm has been selected. Observe the following tips about the key: <ul style="list-style-type: none"> Keep the key in a secure location. Never transmit the key over the network for any reason. If you suspect the key has been compromised, regenerate a new key and rebuild the policy. Longer keys produce a stronger hash.

4. Click **[OK]** when done.

Look Up Certificate Assertion

The *Look Up Certificate* assertion is used to look up a certificate by a variety of methods and then store that certificate's value in a context variable for later use in the policy.

You can look up certificates by:

- Name (trusted certificates only)
- SHA1 Thumbprint
- Subject Key ID
- Subject DN
- Issuer DN and Serial Number

The assertion can be configured to fail if more than one matching certificate is found.

Using the Assertion

This assertion can find the following types of certificates known to the Gateway:

- Trusted Certificate
- User certificate
- LDAP User shadow certificate (for more information, see "Trusted Gateway Accounts" in the *SecureSpan XML VPN Client User Manual*)
- Certificate from LDAP certificate cache, if enabled
- Subject certificate from any private key in the current Gateway keystore

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the Certificate Lookup Properties automatically appear. When modifying the assertion, right-click **Look Up Certificate** in the policy window and select **Certificate Lookup Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Certificate Lookup Properties

☐ Look up Trusted Certificate by Name:

☐ Look up Certificate by ThumbprintSHA1:

☐ Look up Certificate by Subject Key ID:

☐ Look up Certificate by Subject DN:

☒ Look up Certificate by Issuer DN and Serial Number:

Issuer DN:

Serial Number:

☐ Fail if multiple certificates are found






Output Variable Name:

☒ OK

Figure 117: Certificate Lookup Properties

3. Configure the dialog as follows:

Table 100: Certificate Lookup settings

Setting	Description
Look up Certificate by Name 	For this option, enter the name of the trusted certificate to be looked up. This will be matched against the CN value of the trusted certificate. You may specify a context variable.
Look up Certificate by ThumbprintSHA1 	For this option, enter the SHA-1 thumbprint (as a Base-64 string) of the encoded certificate to be looked up. You may specify a context variable.
Look up Certificate by Subject Key ID 	For this option, enter the Subject Key ID (SKI) of the certificate to be looked up. You may specify a context variable.
Look up Certificate by Subject DN 	For this option, enter the name of the certificate subject's Distinguished Name to be looked up. You may specify a context variable.
Look up Certificate by Issuer DN and Serial Number 	For this option, enter the certificate issuer's Distinguished Name, as an RFC 2253 canonical string, and the certificate's Serial Number, as a decimal number, to be looked up. You may specify a context variable.
Fail if multiple certificates are found	<p>Select this check box to fail the assertion if multiple certificates with the specified name are found.</p> <p>Clear this check box to not fail the assertion if multiple certificates with the specified name are found. This setting is the default.</p> <p>Note: The context variable specified under "Output Variable Name" below will not be populated if the assertion fails.</p>
Output Variable Name	<p>For this option, enter the name of the context variable to be used to store the results of the lookup upon successful completion of the assertion. This variable will be of type X.509 Certificate.</p> <p>Note the following:</p> <ul style="list-style-type: none"> When looking up a trusted certificate by name, the context variable will be single-valued if one certificate is found or multivalued if multiple matching certificates are found. When looking up any other certificate type, the context variable will always be single-valued and only the first matching certificate will be stored. <p>The default variable name is certificate.</p>

4. Click **[OK]**.

Look Up Outbound Secure Conversation Session Assertion

The *Look Up Outbound Secure Conversation Session* assertion is used to look up an outbound secure conversation session that has been mapped to the authenticated user and the back-end service on which the secure conversation session is established.

This assertion succeeds if at least one unexpired session is found. This assertion fails if no sessions are found or only expired sessions are found.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Context Variables Created by This Assertion

The Look Up Outbound Secure Conversation Session assertion sets the following context variable that contains all information about the session:

`<prefix>.session`

Where:

- `<prefix>` is defined in the assertion properties (default: **scLookup**)
- specific attributes about the outbound secure conversation session can be retrieved by using:

`<prefix>.session.<attribute>`

For example, to access the session identifier, use `${<prefix>.session.id}`.

The attributes are described in Table 101.

Table 101: Outbound secure conversation session attributes

Attribute	Description
id	The session identifier
user	The authenticated user
To access specific attributes about the user, use the syntax: <code><prefix>.session.user.<user_attribute></code>	
providerId	The user's Identity Provider ID
id	The user's identifier
login	The user's login ID

Attribute	Description
firstName	The user's first name
lastName	The user's last name
email	The user's email address
department	The user's department
subjectDn	The user's X.509 subject DN
creation	The session creation time
expiration	The session's expiration time
scNamespace	The namespace of WS-Secure Conversation

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click **<target>: Look Up Outbound Secure Conversation Session to <service URL>** in the policy window and select **Outbound Secure Conversation Session Lookup Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

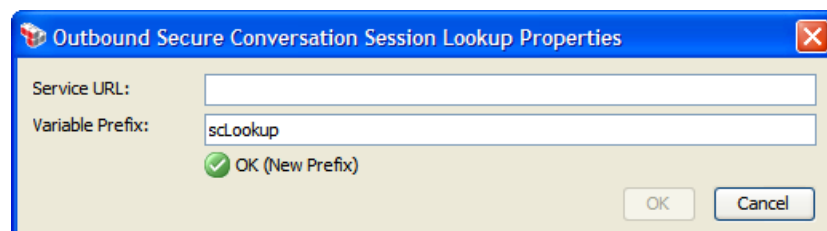



Figure 118: Outbound Secure Conversation Session Lookup Properties

- Enter the **Service URL**. This is the URL of the back-end service that will issue the security context token. 
- Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.

The default variable prefix is **scLookup**.

For an explanation of the validation messages displayed, see Context Variable Validation in the *Layer 7 Policy Manager User Manual*.

5. Click **[OK]**.

(Non-SOAP) Check Results from XML Verification Assertion

The *(Non-SOAP) Check Results from XML Verification* assertion ("Check" assertion) provides a quick way to check the contents of the context variables produced by the "(Non-SOAP) Verify XML Element Assertion" on page 391.

The following is a more in depth description of what happens when you use the "Check" assertion:

1. First, you select the signed element(s) to verify.
2. Next, you select the signature methods and digest methods that you are permitting. Optionally indicate whether to gather the signing certificates as credentials.
3. The "Check" assertion then checks the `<prefix>.elementsVerified` variable for the signed elements and notes the index position of any matches. (This is similar to using the [Look Up Item by Value](#) assertion on the `<prefix>.elementsVerified` variable.)
4. If a match is found, the assertion then checks whether the corresponding index position in the `<prefix>.signatureMethodUris` variable matches any of the "Permitted signature methods". (This is similar to using the [Look Up Item by Index Position](#) assertion on the `<prefix>.signatureMethodUris` variable, followed by an [At Least One Assertion Must Evaluate to True](#) assertion containing one or more [Compare Expression](#) assertion to check the value.)
5. If a match is found, the same thing is repeated on the corresponding index position in the `<prefix>.digestMethodUris` variable to see if it matches any of the "Permitted digest methods". (This is similar to using the [Look Up Item by Index Position](#) assertion on the `<prefix>.digestMethodUris` variable, followed by an [At Least One Assertion Must Evaluate to True](#) assertion containing one or more [Compare Expression](#) assertion to check the value.)
6. If a match is found and you are gathering signing certificates, the assertion retrieves the certificate from the corresponding index position in the `<prefix>.signingCertificates` variable and gathers it as X.509 credentials.

This assertion will succeed only when all elements in the target message that match the XPath are present in the specified verify results and were signed using one of the specified signature and digest methods.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: (Non-SOAP) Check Results from XML Verification [XPath]** in the policy window and select **(Non-SOAP) XML Verification Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

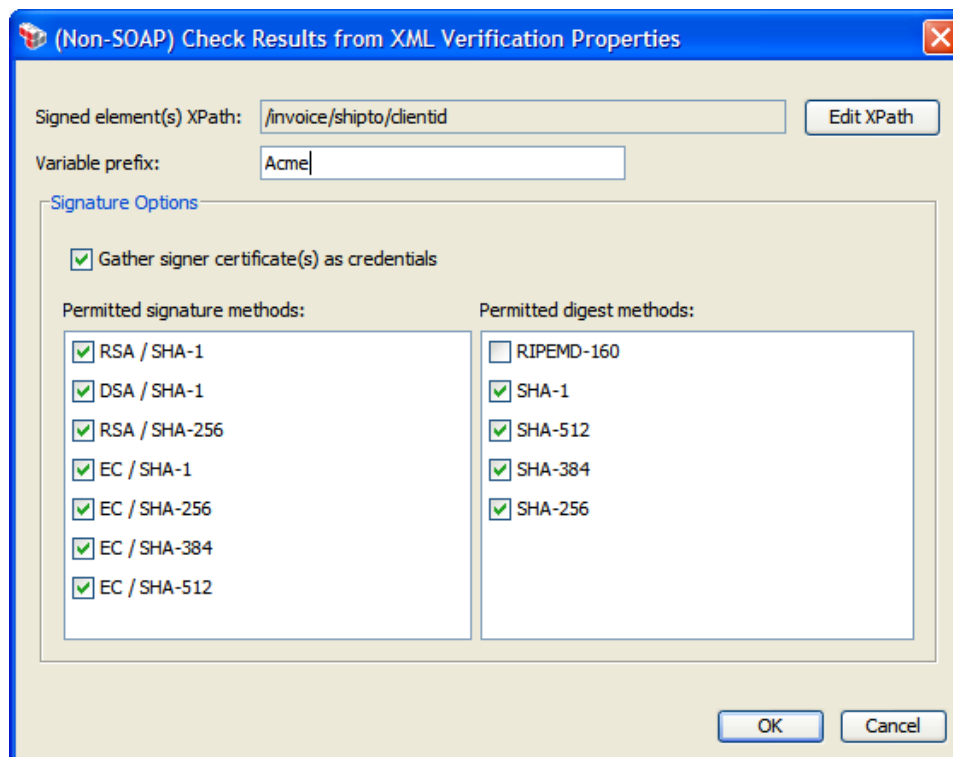


Figure 119: (Non-SOAP) Check Results from XML Verification Properties

3. Click **[Edit XPath]** to specify the signed element(s) to verify. For more information, see "Selecting an XPath" on page 154.
4. Enter the context variable prefix that was used in the [\(Non-SOAP\) Verify XML Element](#) assertion. If no prefix was used, leave the field blank.
5. Select the **Gather signer certificate(s) as credentials** check box if you want to use the signing certificate as an X.509 credential for later authorization with a specific User or Member of Group assertion. "Retrieve Credentials from Context Variable Assertion" on page 250
6. Click **[OK]**.

(Non-SOAP) Decrypt XML Element Assertion

The *(Non-SOAP) Decrypt XML Element* assertion is used to immediately decrypt one or more EncryptedData elements in an XML message (either request, response, or a message context variable). This assertion is intended only for messages not contained within a SOAP envelope. (Advanced technical users may use it on SOAP messages, with the knowledge that the resulting decorated message will almost certainly not be WS-Security compliant.)

Note: The (Non-SOAP) Decrypt XML Element assertion is intended to decrypt elements that were encrypted using the [\(Non-SOAP\) Encrypt XML Element](#) assertions.

Context Variables Created by This Assertion

The (Non-SOAP) Decrypt XML Element assertion sets the following context variables with details of the decryption. **Note:** The *<prefix>* is set in the assertion properties (Figure 120) and is optional. There is no default.

Table 102: Context variables created by (Non-SOAP) Decrypt XML Element assertion

Variable	Description
<i><prefix></i> .elementsDecrypted	Lists the elements that were decrypted.
<i><prefix></i> .encryptionMethodUris	Lists the encryption methods used.
<i><prefix></i> .recipientCertificates	Lists the recipient certificates used in the encryption.

Note: All three multivalued variables always have exactly the same number of values, with the *encryptionMethodUris* and *recipientCertificates* variables containing duplicate values as required to ensure that the encryption method and certificate for *elementsDecrypted[N]* can always be found at *encryptionMethodUris[N]* and *recipientCertificates[N]*, respectively (where 'N' is a nonnegative integer).

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: (Non-SOAP) Decrypt XML Element [XPath]** in the policy window and select **(Non-SOAP) XML Element Decryption Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

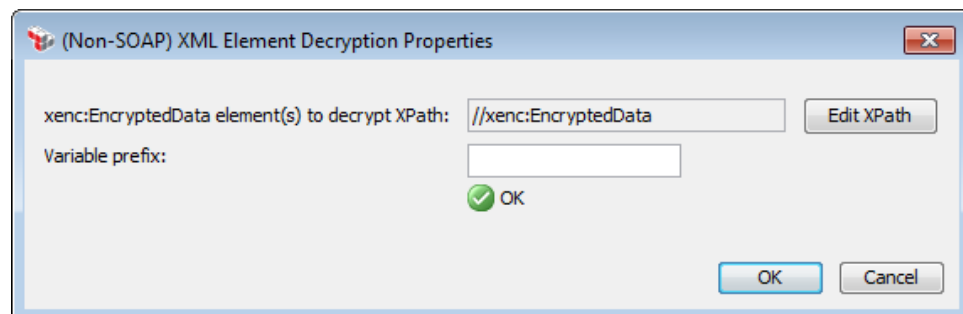


Figure 120: (Non-SOAP) XML Element Decryption Properties

3. Click **[Edit XPath]** to specify the xenc:EncryptedData element(s) to verify. For more information, see "Selecting an XPath" on page 154.
4. Optionally enter a prefix to be added to the context variables created by this assertion (see Table 102). A prefix is required if this assertion appears more than once in a policy to prevent variable values from being overwritten. **Tip:** The on-screen validator will warn you if there are any issues with the prefix name.
3. Click **[OK]**.

(Non-SOAP) Encrypt XML Element Assertion

The *(Non-SOAP) Encrypt XML Element* assertion is used to immediately encrypt one or more elements in an XML message (either request, response, or a message context variable). This assertion is designed only for messages not contained within a SOAP envelope. It is also used by the "Working with Internal Use Policies" on page 33.

Note that when this assertion is used within an Audit Message Filer internal policy, the recipient certificate should match the audit viewer key. For more information on defining the audit viewer key, see Private Key Properties in the *Layer 7 Policy Manager User Manual*.

W A R N I N G

This assertion should be used only by advanced users who have a specific need to encrypt XML elements outside of a SOAP envelope; otherwise, the "Encrypt Element Assertion" on page 346 is normally used.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: (Non-SOAP) Encrypt XML Element [XPath]** in the policy window and select **(Non-SOAP) XML Element Encryption Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

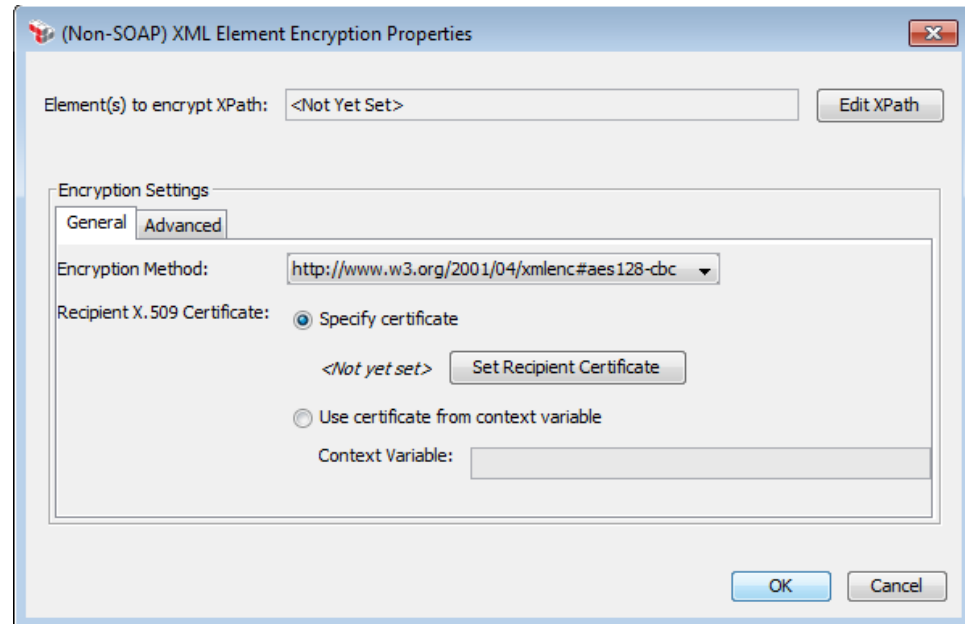


Figure 121: (Non-SOAP) XML Element Encryption Properties

3. Click **[Edit XPath]** to specify the element(s) to encrypt. For more information, see "Selecting an XPath" on page 154.
4. Configure the **Encryption Settings**. For more information, see Configuring Encryption Settings in the *Layer 7 Policy Manager User Manual*.

Note: You must specify a recipient certificate, otherwise the assertion will always fail.

5. Click **[OK]** when done.

(Non-SOAP) Sign XML Element Assertion

The *(Non-SOAP) Sign XML Element* assertion is used to immediately sign one or more elements in an XML message (either request, response, or a message context variable).

This assertion is designed only for messages not contained within a SOAP envelope.

W A R N I N G

This assertion should be used only by advanced users who have a specific need to sign XML elements outside of a SOAP envelope. If working with a SOAP document, use the "Sign Element Assertion" on page 407 instead.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about selecting a private key for this assertion, see *Selecting a Custom Private Key* in the *Layer 7 Policy Manager User Manual*.

Sign Element vs. Immediate Sign XML Element

The (Non-SOAP) Sign XML Element assertion is designed to sign XML elements that are not within a SOAP message. This signing occurs immediately and the signature is inserted into the contents of the XML element.

By comparison, the "Sign Element Assertion" on page 407 is used to sign elements within a SOAP message. This signing is scheduled in advance and conforms to WS-Security standards. The signature is added to the message's security header; the element itself is untouched.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: (Non-SOAP) Sign XML Element [XPath]** in the policy window and select **(Non-SOAP) XML Element Signature Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 122: (Non-SOAP) XML Element Signature Properties

3. Configure the properties as follows:

Table 103: (Non-SOAP) XML Element Signature settings

Setting	Description
Edit XPath	<p>Click [Edit XPath] to specify the element(s) to sign. For more information, see "Selecting an XPath" on page 154.</p> <p>Note that only elements can be signed. One Signature element is created per element being signed. The Signature is added as the last child of the element being signed; the Signature always uses the Enveloped transform and always includes the entire signing certificate in the KeyInfo as X509Data.</p>
Target URI Reference	<p>The signature reference requires an ID for the target element. Indicate how the Gateway should determine the ID:</p> <ul style="list-style-type: none"> • Automatic: Select this to instruct the Gateway to look for an existing ID based on a built-in list of possible attribute names. If one is found, then its value is used. If one is not found, the Gateway will add an 'Id' attribute to the element being signed, with a randomly generated ID value, and references that. <p>Note: Having a new attribute 'Id' added may cause some difficulties if the schema of the signed element does not allow an 'Id' attribute. If this is the case, then enter a specific ID attribute name instead.</p> • Specify ID Attribute Name: Select this option to manually specify the name of the ID attribute to use. If you choose this option, the Gateway will no longer recognize the built-in list of names. Instead, it will use the specified attribute value if it already exists on the target element and will generate a new one if it doesn't exist. <p>Enter the name as a string value in one of the following formats:</p> <ul style="list-style-type: none"> • NAME (e.g., <i>abc</i>) • PREFIX:NAME (e.g., <i>abc:xyz</i>) • {URI}NAME (e.g., <i>{urn:issn:1535-3613}abc</i>) • {URI}PREFIX:NAME (e.g., <i>{urn:issn:1535-3613}abc:xyz</i>) <p>Note: If a URI is specified, they must be absolute. Relative URIs cannot be used in signatures.</p> <p>Tip: If a prefix is specified, it may not necessarily be used. The Gateway will first attempt to reuse an existing namespace declaration for the namespace URI, if one exists, regardless of its prefix. If the Gateway needs to add a new namespace declaration, it will attempt to use the requested prefix if it is available. However if the requested prefix is already used in a different namespace URI, the Gateway will substitute a different prefix instead.</p>

Setting	Description
Signature Location	<p>Specify where the signature should be located:</p> <ul style="list-style-type: none"> • Add one Signature to....: Select this option to add a signature to each signed element. Choose where the signature should be added using the drop-down list: as the first or last child of the signed element. • Create detached signature and....: Select this option to create a detached signature which can later be added to the same or different document. A detached signature is a single signature that covers all elements matching the XPath of the elements to sign. It is placed in the context variable that you specify here and is not added to the document. • Include Enveloped transform: When creating a detached signature, select this check box to optionally include the Enveloped transform. Do this if the detached signature will be manually added to the document as a descendent of one of the signed elements.
Signature Type	<p>Specify the hash algorithm to use for the Signature Digest or Reference Digest.</p> <ul style="list-style-type: none"> • Signature Digest: By default, an appropriate signature digest will be selected based on the signing key size and the current value of the <i>wss.decorator.digsig.messagedigest</i> cluster property. If you wish to use a specific digest, select it from the drop-down list. • Reference Digest: By default, references will be created using the same digest algorithm as the signature digest. If you wish them to use a specific digest instead, select it from the drop-down list.

4. Click **[OK]** when done.

(Non-SOAP) Validate SAML Token Assertion

The *(Non-SOAP) Validate SAML Token* assertion is used to validate a SAML token that was not delivered using WS-Security. This assertion will validate the Subject, Statements, Conditions, and Signatures in a SAML token that is not contained in a SOAP header.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Adding and Configuring the Assertion

1. Add the Require SAML Token Profile assertion to the policy development window as described in [Adding an Assertion](#). The **SAML Token Properties** appears.

2. Complete the properties as shown below.

Editing the Assertion

1. In the policy development window, right-click on the assertion and then select **(Non-SOAP) Validate Assertion Properties**. The assertion properties are displayed.
2. Modify the tabs as necessary. Refer to the corresponding step below for information about each tab.
3. Click **[OK]** when done.

Step 1: Introduction

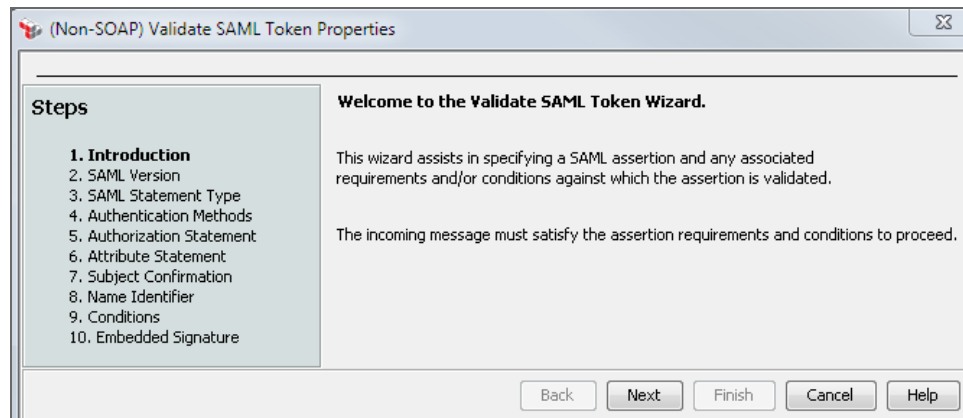


Figure 123: (Non-SOAP) Validate SAML Token Properties - Step 1

This step introduces the Non-SOAP Validate SAML Token properties.

Step 2: SAML Version

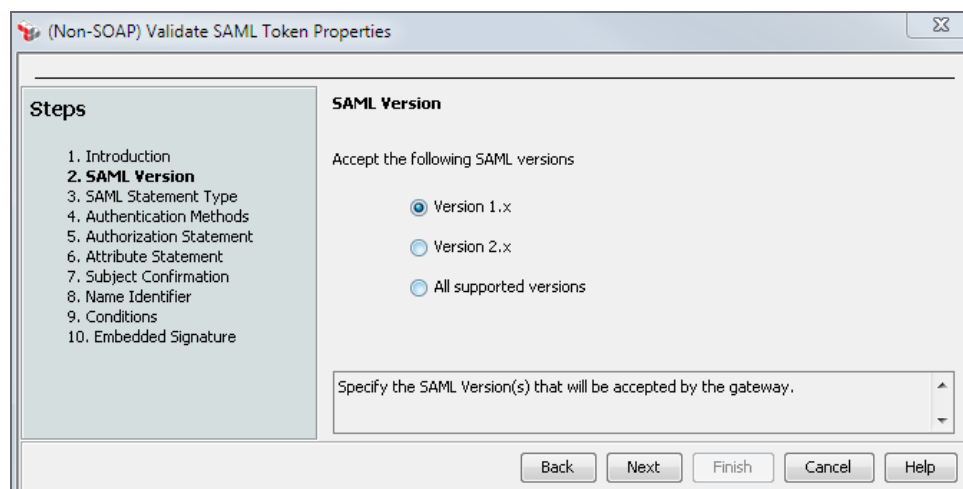


Figure 124: (Non-SOAP) Validate SAML Token Properties - Step 2

Specify which SAML versions will be accepted by the Gateway: version 1.1, version 2.0, or any supported version.

Step 3: SAML Statement Type

(Non-SOAP) Validate SAML Token Properties

Steps

1. Introduction
2. SAML Version
- 3. SAML Statement Type**
4. Authentication Methods
5. Authorization Statement
6. Attribute Statement
7. Subject Confirmation
8. Name Identifier
9. Conditions
10. Embedded Signature

Please select the SAML statement

☒ Authentication Statement
☐ Authorization Decision Statement
☐ Attribute Statement

Select the SAML Statement Type you wish to configure.

Back Next Finish Cancel Help

Figure 125: (Non-SOAP) Validate SAML Token Properties - Step 3

Select the type of SAML statement to configure:

- **Authentication Statement:** Proceed to [Step 4](#).
- **Authorization Decision Statement:** Proceed to [Step 5](#).
- **Attribute Statement:** Proceed to [Step 6](#).

Step 4: Authentication Methods

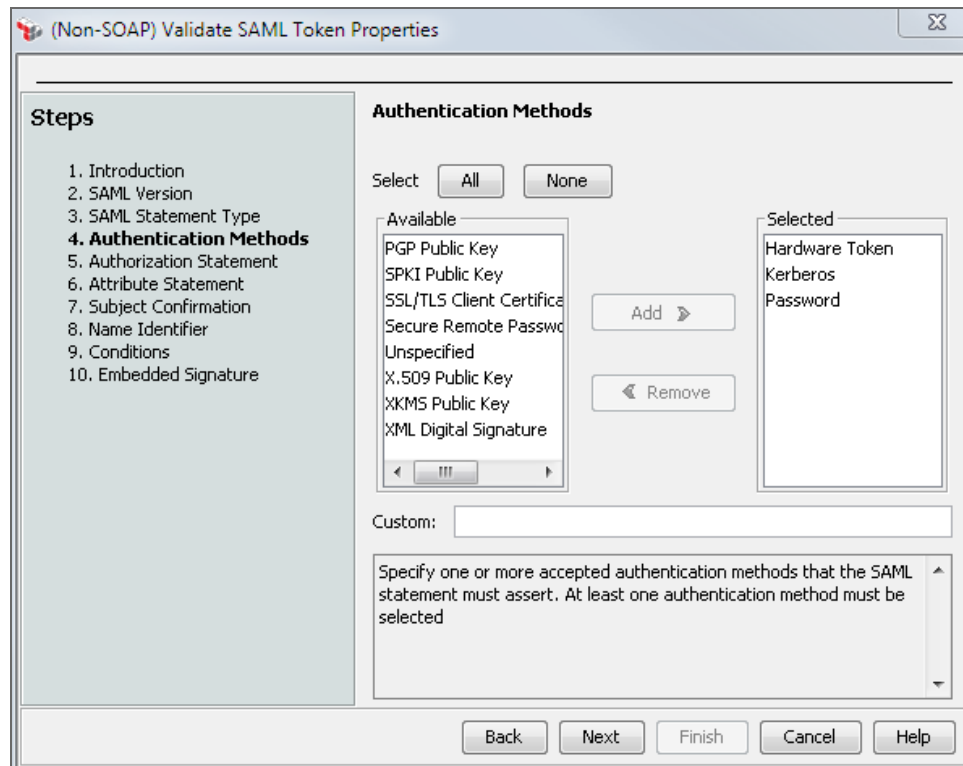


Figure 126: (Non-SOAP) Validate SAML Token Properties - Step 4

Select the authentication methods that will be enforced by the assertion. You must choose at least one method.

Hints:

- Hold down the [Ctrl] or [Shift] keys to select multiple items at once.
- Click **[All]** to choose every available authentication method.
- Click **[None]** to quickly clear the **Selected** list and start again.
- Select the **Unspecified** method to allow authentication by an unspecified method.
- The **Available** list only displays the methods that are applicable to the SAML version chosen in Step 2 of the wizard.

In the **Custom** field, optionally enter any URI custom authentication methods, separated by spaces. You may reference context variables (either single- or multi-valued variables with space-separated URI values).

Note: The **SSL/TLS Certificate Based Client Authentication** method is not related to the [Require SSL or TLS Transport](#) assertion. This method refers to the original authentication, not to the current request which may or may not have used SSL. The SAML-supported authentication methods are outlined in the **SAML 1.1** and **2.0** specification documents provided at <http://www.oasis-open.org>

Proceed to Step 7: Subject Confirmation.

Step 5: Authorization Statement

(Non-SOAP) Validate SAML Token Properties

Steps

1. Introduction
2. SAML Version
3. SAML Statement Type
4. Authentication Methods
- 5. Authorization Statement**
6. Attribute Statement
7. Subject Confirmation
8. Name Identifier
9. Conditions
10. Embedded Signature

Authorization Statement

Resource

Action

Action Namespace

Specify the Resource [required] that the SAML statement MUST describe; the Resource Action [required] and the Action Namespace [optional].

Back Next Finish Cancel Help

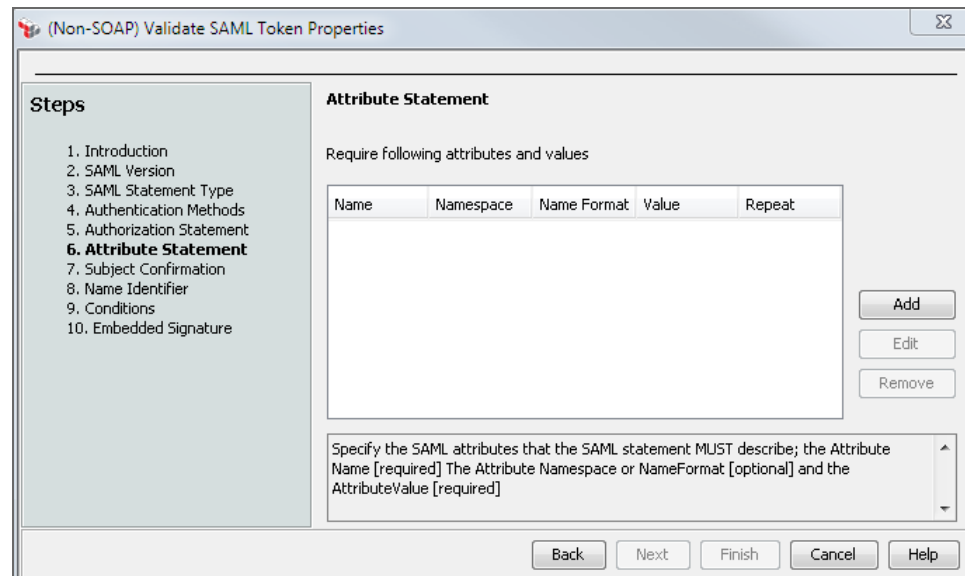
Figure 127: (Non-SOAP) Validate SAML Token Properties - Step 5

Specify the resource that the SAML statement must describe, the resource action, and the action namespace.

- **Resource:** Enter a value for the resource that the SAML statement must describe (for example, "http://acme.org").
- **Action:** Enter an action value for the resource (for example, "GET").
- **Action Namespace:** Optionally enter a corresponding action namespace value (for example, "acmeNamespace").

Proceed to Step 7: Subject Confirmation.

Step 6: Attribute Statement



(Non-SOAP) Validate SAML Token Properties

Steps

1. Introduction
2. SAML Version
3. SAML Statement Type
4. Authentication Methods
5. Authorization Statement
- 6. Attribute Statement**
7. Subject Confirmation
8. Name Identifier
9. Conditions
10. Embedded Signature

Attribute Statement

Require following attributes and values

Name	Namespace	Name Format	Value	Repeat

Add Edit Remove

Specify the SAML attributes that the SAML statement MUST describe; the Attribute Name [required] The Attribute Namespace or NameFormat [optional] and the AttributeValue [required]

Back Next Finish Cancel Help

Figure 128: (Non-SOAP) Validate SAML Token Properties - Step 6

Define one or more SAML attributes that must be described by the SAML statement.

1. Click **[Add]** and then complete the Edit SAML Attribute Constraints dialog:
 - **Attribute Name:** Enter the name of the attribute.
 - **Attribute Namespace:** Optionally enter a namespace for the attribute. This applies only to SAML 1.1.
 - **Attribute Name Format:** Optionally specify a URI reference that describes the format of the attribute name. Only attributes that declare this format will be accepted. This applies only to SAML 2.0.
 - **Unspecified:** If no name format is provided, the default value of *urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified* is used.
 - **URI Reference:** This option uses the URI *urn:oasis:names:tc:SAML:2.0:attrname-format:uri*.
 - **Basic:** This option uses the URI: *urn:oasis:names:tc:SAML:2.0:attrname-format:basic*.
 - **Other:** Select this option to define your own attribute name format in the box below.
 - **Attribute Value:** To require an exact variable match, select **Specific Value** and then enter a set value. To require that a particular attribute be present, but

allow it to have any non-empty value rather than requiring a specific match, select the **Allow any non-empty value** option.

When a non-empty attribute value is required, you can separately validate the attribute contents using XPath expressions, transient variables, and the [Compare Expression](#) assertion.

To modify an attribute statement, select it and click **[Edit]**. To delete an attribute statement, select it and click **[Delete]**.

2. Click **[OK]** to enter the attribute into the table. Repeat to configure additional attributes.

To modify an existing Attribute Statement, select it from the list and then click **[Edit]**.

To remove an Attribute Statement, select it from the list and then click **[Remove]**.

Tip: The attribute values validated by the Attribute Statement are available in context variables. For more information, see "Context Variables Created by This Assertion" in "Require SAML Token Profile Assertion" on page 228.

Step 7: Subject Confirmation

(Non-SOAP) Validate SAML Token Properties

Steps

1. Introduction
2. SAML Version
3. SAML Statement Type
4. Authentication Methods
5. Authorization Statement
6. Attribute Statement
- 7. Subject Confirmation**
8. Name Identifier
9. Conditions
10. Embedded Signature

Subject Confirmation

Accept the following Subject Confirmation Methods

Subject Confirmation Methods

- ☒ Holder of Key
- ☒ Sender Vouches
- ☒ Bearer
- ☐ None

Subject Confirmation Data

Recipient:

- ☐ Check Address
- ☒ Check Validity Period

Specify one or more subject confirmations that will be accepted by the gateway and whether the message signature is required as the proof material

Back Next Finish Cancel Help

Figure 129: (Non-SOAP) Validate SAML Token Properties - Step 7

Select one or more subject confirmation methods that should be accepted by the Gateway and indicate whether the message signature is required as the proof material:

Holder-of-Key

This allows SAML tokens that use the *Holder-of-Key* subject confirmation method (with the standard URI *urn:oasis:names:tc:SAML:1.0:cm:holder-of-key* or *urn:oasis:names:tc:SAML:2.0:cm:holder-of-key*, depending on the selected SAML version in Step 2 of the wizard). For such assertions, the Gateway will require that the subject demonstrate possession of the private key corresponding to the public key in the Subject certificate.

The Holder-of-Key subject confirmation method currently requires that the request ticket's "SubjectConfirmation" element contain a "KeyInfo" element that contains a complete copy of the Subject's X.509 certificate. Any other form of Holder-of-Key ticket will be rejected by the Gateway.

The request Subject may use one of two methods to prove that they hold this key:

- The request includes at least one element covered by a valid [WSS message signature](#), and the signing certificate is the Subject certificate. Or,
- The request arrived over SSL/TLS with [client certificate authentication](#), and the client certificate exactly matches the Subject certificate.

Sender Vouches

This allows SAML tokens that use the *Sender Vouches* subject confirmation method (with the standard URI *urn:oasis:names:tc:SAML:1.0:cm:sender-vouches* or *urn:oasis:names:tc:SAML:2.0:cm:sender-vouches*, depending on the selected SAML version in Step 2 of the wizard). For such assertions, the Gateway will require that the sender, presumably different from the subject, vouches for the verification of the subject.

The Sender Vouches subject confirmation method is typically used only in a SAML identity bridging policy.

Three conditions must be met in order to use the Sender Vouches confirmation method:

- An existing trust relationship with the sender ("Attesting Entity") must be configured in the Gateway. To do this, import the sender's certificate, configured as a "SAML Attesting Entity" certificate, into the Trust Store. For more information, see [Managing Certificates](#).
- The SAML ticket used by the SAML token must be bound to the request message by one of the following methods:
 - Send the request over SSL using the sender certificate as the SSL client certificate, OR

- If SSL is not used, then the SAML ticket needs to be bound to the message with a [WSS signature](#). One complication here is that the SAML ticket does not necessarily contain or refer to the sender certificate; it usually contains or refers to the subject certificate and, assuming that the ticket is signed, contains or refers to the certificate of the ticket issuer. In this method, therefore, the WSS signature must cover both the SAML token and the relevant portions of the rest of the message that use the sender certificate as the signing certificate.
- The format of the request message must conform to the OASIS Web Services Security standards: SAML Token Profile 1.0 (for SAML 1.1) or SAML Token Profile 1.1 (for SAML 2.0). The Gateway does not support references to SAML tokens that are not included with the request message.

The OASIS Web Services Security: SAML Token Profile 1.0 standards document is available online at: www.oasis-open.org/committees/download.php/1048/WSS-SAML-06.pdf.

Bearer

This allows SAML tokens that use the *Bearer Token* subject confirmation method (with the standard URI *urn:oasis:names:tc:SAML:1.0:cm:bearer* or *urn:oasis:names:tc:SAML:2.0:cm:bearer*, depending on the selected SAML version in Step 2 of the wizard). Like HTTP cookies, such assertions will always be assumed to belong to whatever message contains them, and the subject will be assumed to be the sender of the message.


The Bearer Token subject confirmation method does not protect against an attacker modifying the message or stealing a copy of the assertion and attaching it to an unauthorized message. To protect the secrecy of the SAML token when using the Bearer subject confirmation method, be sure to select the **SSL-TLS Certificate Based Client Authentication** check box in Step 4 of the SAML Token Profile Wizard.

None

This allows SAML tokens that do not contain a subject confirmation method.

Not having a subject confirmation method exposes the system to various threats. To protect the secrecy of the SAML token when a confirmation method is not used, be sure to select the "SSL-TLS Certificate Based Client Authentication" option in Step 3 of the SAML Token Profile Wizard.

If SAML version 2.0 is permitted, complete the **Subject Confirmation Data** fields:

- **Recipient:** This property allows the expected recipient to be configured. You may enter the name directly or enter a String context variables. Leave this field blank to allow any recipient. 

- **Check Address:** Select this check box to validate the "Address" attribute. Currently, the Gateway only supports IPv4 addresses.
- **Check Validity Period:** Select this check box to check the time period validity period in the request. The permissible clock skew for validation is defined by the cluster properties *samlAssertion.validate.notBeforeOffsetMin* and *samlAssertion.validate.notOnOrAfterOffsetMin*.

Note: If there are no validity period constraints in the request message, then there is nothing to check and validation (of the time period constraints) will always succeed.

Note that if any of the Subject Confirmation Data fields fail validation, the assertion will not fail.

Step 8: Name Identifier

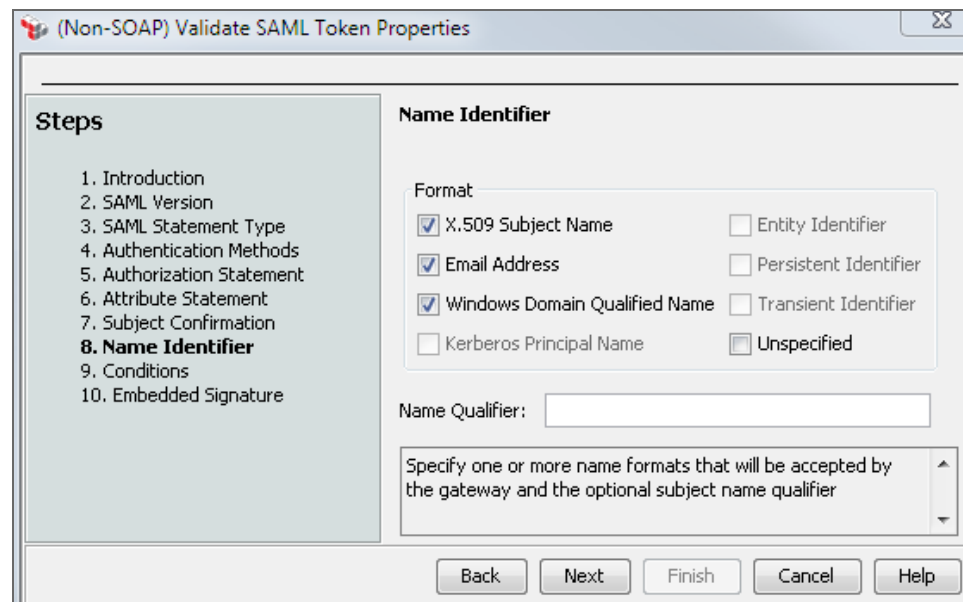



Figure 130: (Non-SOAP) Validate SAML Token Properties - Step 8

Specify the name formats that are acceptable to the Gateway; optionally enter a subject name qualifier:

- **Name Qualifier:** Optionally enter a subject name identifier (for example, "www.example.com"). You may reference context variables. 
- **Format:** Select one or more subject name formats that should be accepted by the Gateway. Select the **Unspecified** check box if the subject name format is not known. This will cause the Gateway to attempt to match the subject name

identifier specified in the **Name Qualifier** field against the user login property. If the **Name Qualifier** field is blank, then the Gateway will not verify the Name Qualifier attribute value.

You can only select name formats applicable to the SAML version chosen in Step 2 of the wizard.

Step 9: Conditions

(Non-SOAP) Validate SAML Token Properties

Steps

1. Introduction
2. SAML Version
3. SAML Statement Type
4. Authentication Methods
5. Authorization Statement
6. Attribute Statement
7. Subject Confirmation
8. Name Identifier
- 9. Conditions**
10. Embedded Signature

Conditions

☒ Check Assertion Validity Period

Maximum Expiry Time: minutes


Audience Restriction:

Optionally indicate whether the token's validity period should be checked, whether the token's expiry time should be further restricted, or whether to restrict the audience for the SAML token.

Back Next Finish Cancel Help

Figure 131: (Non-SOAP) Validate SAML Token Properties - Step 9

In this step, you can specify any conditions to be observed.

- **Check Assertion Validity Period:** Select this check box to verify that the SAML token is still within its validity period, using the current Gateway time. Clear this check box to not check the validity period within the token.
- **Maximum Expiry Time:** Specify the maximum allowable expiry time period for the SAML token. The Gateway will use the earlier of the expiry date or the specified period. This allows you to restrict the token's expiry date with an earlier date. (If the specified date is later than the token's expiry date, then the token's date takes priority.) Tokens that exceed the expiry time will cause policy consumption to fail and audit message code 6108 will be logged.
The default is **0** (zero), which indicates that token expiration is not checked. The maximum allowable expiry time is 100 years.
- **Audience Restriction:** Enter an audience restriction constraint into the field. You may reference context variables. 

Step 10: Embedded Signature

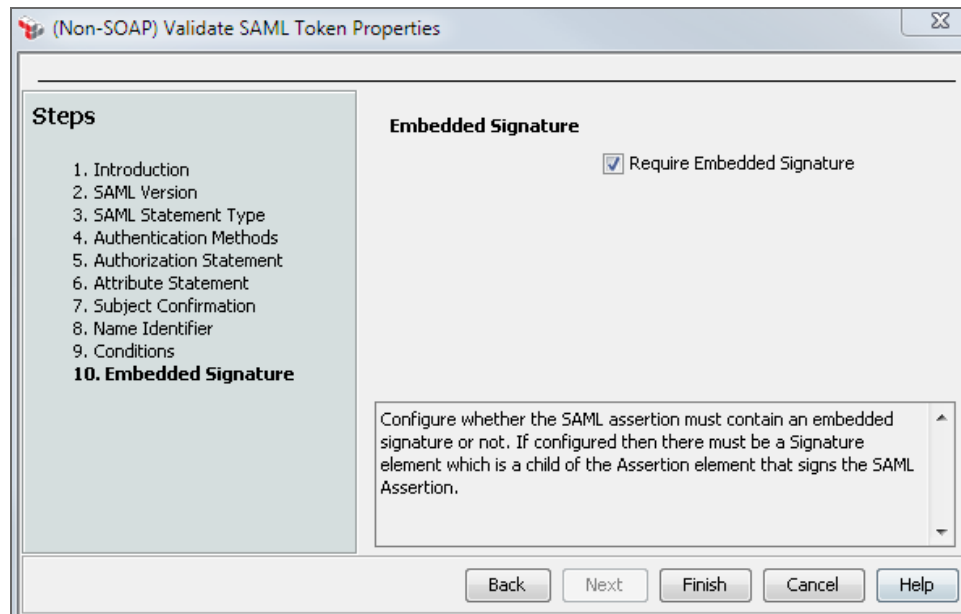


Figure 132: (Non-SOAP) Validate SAML Token Properties - Step 10

Select the **Require Embedded Signature** check box to require an embedded signature in the SAML token. An invalid signature will cause the assertion to fail.

Clear the check box if an embedded signature is not required.

(Non-SOAP) Verify XML Element Assertion

The (Non-SOAP) Verify XML Element assertion is used to immediately verify one or more Signature elements in an XML message (either request, response, or a message context variable).

This assertion supports the special prefix "local:" in the ID attribute, for matching the namespace URI against the owning element rather than the attribute.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Context Variables Created by This Assertion

The (Non-SOAP) Verify XML Element assertion sets the following context variables with details of the verification. **Note:** The `<prefix>` is set in the assertion properties (Figure 133) and is optional. There is no default.

Table 104: Context variables created by (Non-SOAP) Verify XML Element assertion

Variable	Description
<p><prefix> .elementsVerified</p>	<p>Lists the elements that were verified.</p> <p>Detailed technical view</p> <p>The <i>elementsVerified</i> are the target elements covered by the signature. A ds:Signature element created by third-party software (or by the Gateway or the Securespan XML VPN Client, if using WSS) may cover many elements with a single signature. Each covered element has its own row in this table, though the <i>signatureElements</i> column will contain the same ds:Signature element for each such row. Multiple levels of multi-matching are possible:</p> <ul style="list-style-type: none"> • The XPath may match more than one ds:Signature element. Every matching Signature will be verified. • Each ds:Signature may have references to more than one covered element. Each covered element will be included in its own row in the results table.
<p><prefix> .signatureMethodUri</p>	<p>Lists the signature methods used.</p>
<p><prefix> .digestMethodUri</p>	<p>Lists the digest methods used.</p>
<p><prefix> .signingCertificates</p>	<p>Lists the X.509 certificates used to sign the elements.</p>
<p><prefix> .signatureValues</p>	<p>Lists the signature values in Base-64 format.</p>
<p><prefix> .signatureElements</p>	<p>Lists the ds:Signature elements for each signature.</p>

Note: Similar to the "(Non-SOAP) Decrypt XML Element Assertion" on page 374, all these context variables will always contain the same number of values. All (except for *elementsVerified*) may contain duplicate values as needed to ensure that the indexes always line up with the corresponding element.

Tip: Use the "(Non-SOAP) Check Results from XML Verification Assertion" on page 372 to check that these results contain expected values.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.

- Right-click **<target>: (Non-SOAP) Verify XML Element [XPath]** in the policy window and select **XML Element Verification Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

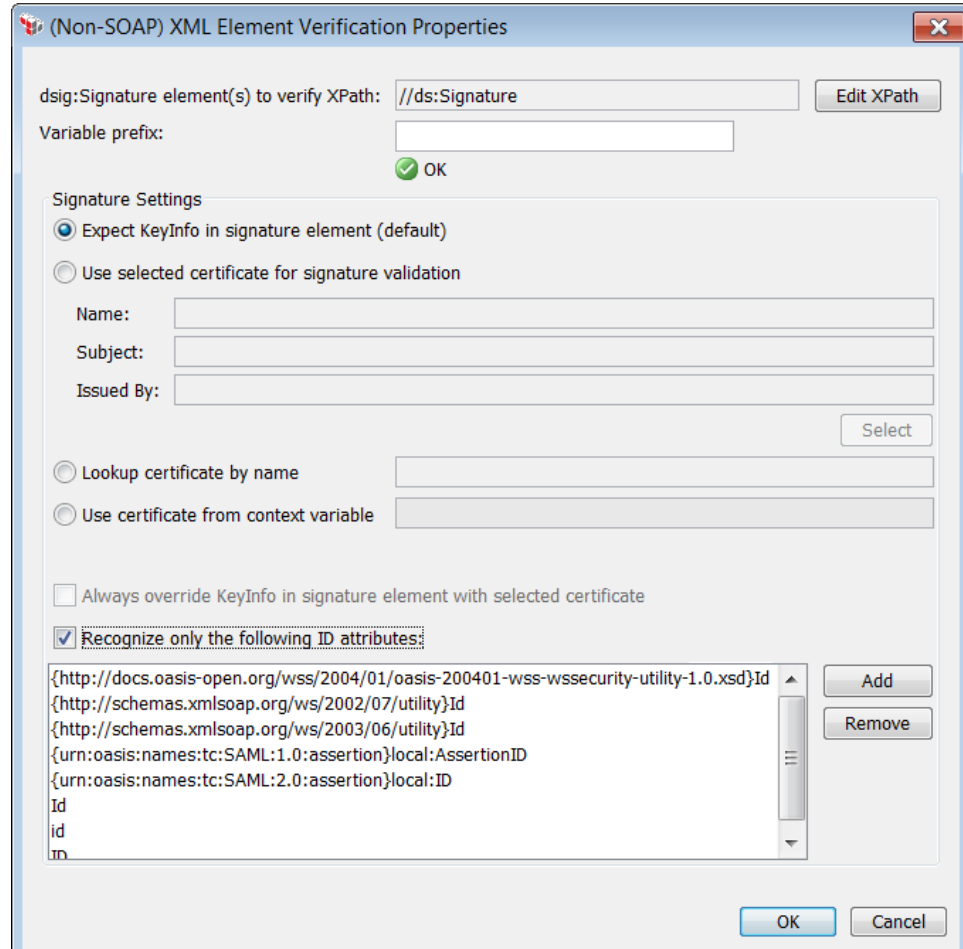



Figure 133: (Non-SOAP) XML Element Verification Properties

- Configure the properties as follows:

Table 105: (Non-SOAP) XML Element Verification settings

Setting	Description
Edit XPath	Click [Edit XPath] to specify the dsig:Signature element(s) to verify. For more information, see "Selecting an XPath" on page 154.
Variable prefix	Optionally, enter a prefix that will be added to the context variables created by this assertion . This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy. For an explanation of the validation messages displayed, see Context

Setting	Description
	Variable Validation in the <i>Layer 7 Policy Manager User Manual</i> .
<i>Signature Settings</i>	
Expect KeyInfo in signature element (default)	<p>Choose this option to use the certificate identified by the <code><ds:keyInfo></code> element within the signature in the message. This setting is the default.</p> <p>Note: The certificate is for the default recipient. To override this default recipient, see "Changing the WSS Assertion Recipient" on page 146.</p>
Use selected certificate for signature validation	<p>Choose this option to browse for the certificate to use. Click [Select] and then locate the certificate. The certificate details will appear in the Name, Subject, and Issued By fields. Examine the details to ensure that it is the correct certificate.</p>
Look up certificate by name	<p>Choose this option to manually specify the certificate to use for validation.</p> <p>Note: Ensure that the specified certificate exists, otherwise the assertion will fail.</p>
Use certificate from context variable 	<p>Choose this option to specify a context variable that will resolve to the certificate name at run time. If more than one certificate matches the name, then the first valid certificate is used.</p>
Always override KeyInfo in signature element with selected certificate	<p>Select this check box to always use the selected certificate, regardless of whether the <code><ds:keyInfo></code> element specifies a certificate.</p> <p>Clear this check box to use the selected certificate only if the <code><ds:keyInfo></code> element does not specify a certificate. If it does, it will be used instead of the selected certificate. This setting is the default.</p> <p>This option is available only when a certificate has been manually selected.</p>
Recognize only the following ID attributes	<p>Select this check box to specify the attribute names to recognize when looking for the elements that a signature may reference.</p> <p><i>To add an attribute:</i></p> <ol style="list-style-type: none"> Click [Add]. Enter the ID attribute either as a NAME (e.g., <i>NewAttr</i>) or {URI}NAME (e.g., <i>{urn:oasis:names:tc:SAML:2.0:assertion}NewAttr</i>). Click [OK]. <p><i>To remove an attribute:</i></p> <ol style="list-style-type: none"> Select the line to remove. Click [Remove].

Setting	Description
	<p>Clear this check box to recognize only the default set of ID attributes:</p> <p>{http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd}Id</p> <p>{http://schemas.xmlsoap.org/ws/2002/07/utility}Id</p> <p>{http://schemas.xmlsoap.org/ws/2003/06/utility}Id</p> <p>{urn:oasis:names:tc:SAML:1.0:assertion}AssertionID</p> <p>{urn:oasis:names:tc:SAML:2.0:assertion}ID</p> <p>Id</p> <p>id</p> <p>ID</p> <p>Note: The special prefix <i>"local:"</i> in the ID attribute matches the namespace URI against the owning element rather than the attribute. All other prefixes are ignored.</p>

4. Click **[OK]**.

Process RSTR Response Assertion

The *Process RSTR Response* assertion takes an RSTR response message as an input and processes this message to get the security context.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Context Variables Created by This Assertion

The Process RSTR Response assertion sets the following context variables with details about the security context. **Note:** The default *<prefix>* is "rstrResponseProcessor" and can be changed in the assertion properties (Figure 134).

Table 106: Context variables created by Process RSTR Response assertion

Variable	Description
<i><prefix></i> .token	Stores the token from the security context (either SAML or Security Context Token).
<i><prefix></i> .createTime	Stores the create time of the secure conversation session, in absolute UTC time.
<i><prefix></i> .expiryTime	Stores the expiry time of the secure conversation session, in absolute UTC time.
<i><prefix></i> .	Stores the server entropy, if the RSTR response message contains an

Variable	Description
serverEntropy	entropy. This variable does not apply to SAML Tokens.
<prefix>.fullKey	Stores the full key, if the RSTR response message contains an encrypted key or a binary secret.
<prefix>.keySize	Stores the size of the key, in bits, from the RSTR response. Contains zero if the key size is not present.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click **Process RSTR Response** in the policy window and select **RSTR Response Processor Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

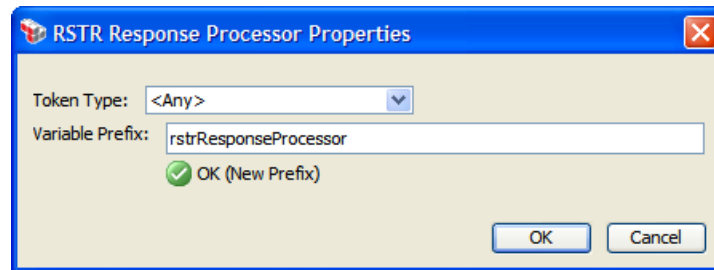


Figure 134: RSTR Response Processor Properties

- Choose the token type to be requested: SAML or Security Context Token. If SAML, select the SAML version (1.1 or 2.0).
- Optionally, enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.

The default prefix is **rstrResponseProcessor**.

For an explanation of the validation messages displayed, see Context Variable Validation in the *Layer 7 Policy Manager User Manual*.

- Click **[OK]**.

Protect Against Message Replay Assertion

The *Protect Against Message Replay* assertion is used to protect the Gateway against possible replay attacks. This replay protection can either be cluster-wide (default) or per node, depending on the setting of the cluster property *cluster.replayProtection.multicast.enabled*.

Note the following important issues when using this assertion:

- Depending on the expiry period set in the assertion, using the Protect Against Message Replay assertion in a Gateway cluster may increase request message processing time and require more memory. To mitigate this, place this assertion after a [Authenticate User or Group](#) or [Authenticate Against Identity Provider](#) assertion to help confine the protection to successfully authenticated messages, thereby reducing system processing and memory requirements.
- This assertion should not be used in any policy that will process messages from JMS destinations that are configured with the "On completion" acknowledgment mode without a specified failure queue.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about selecting the target identity for this assertion, see "Selecting a Target Identity" on page 152.

Details for Advanced Users

The Protect Against Message Replay assertion uses an internal replay ID. This ID is based on either a WS-Addressing Message ID or the timestamp of the request combined with other information that depends on how the message was signed:

- For a request message signed with a WS-Security one-shot X.509 signature, the replay ID is comprised of the following:
 - The SHA-1 of the WS-Addressing MessageID, if present, or the timestamp creation date
 - The signing certificate's subject and issuer DNs
 - The signing certificate's subject key identifier
- For a request message signed with a key derived from a [WS-SecureConversation](#) security context, the replay ID is the MessageID or timestamp created date and the security context identifier.

- For a request message signed with a key derived from an EncryptedKey, the replay ID is the MessageID or timestamp created date and the EncryptedKeySHA1 value.
- For a request message signed with a WS-Security Kerberos token, the replay ID is the MessageID or timestamp created date and the SHA-1 of the Kerberos token.

In all cases, the granularity of the timestamps is determined by the message sender. While the Securespan XML VPN Client always uses at least millisecond-granular timestamps (with a random count of up to one million nanoseconds, to reduce the chance of an ID collision), many tools will use second-granular timestamps by default, resulting in spurious duplicate IDs if MessageIDs are not used and more than one message is sent per second per signing identity.

The Protect Against Message Replay assertion offers two different modes: *Default* or *Custom*.

Default Mode

The assertion first attempts to use a signed WS-Addressing Message ID in the message as the basis for replay protection. If the Securespan XML VPN Client is deployed, you can enforce the presence of Message IDs by using the "Require WS-Addressing Assertion" on page 477.

Note: A Message ID that is present but not signed will not be used by the Protect Against Message Replay assertion. The assertion will use a signed time stamp instead, if one is available.

If no Message ID is present (and the policy is not configured to enforce the presence of one), the message time stamp is used for replay protection. The Gateway will reject a message as a possible replay if detects any of the following:

- A duplicate creation time stamp in a message
- An expired time stamp is present
- The creation time stamp is more than 30 days old.

In the Default mode, the Protect Against Message Replay assertion behaves exactly the same as the *WSS Replay Protection* assertion found in versions prior to 5.2.

Custom Mode

In this mode, you may specify a context variable that contains the identifier to check and how long the identifier should be saved. This allows you to verify non-SOAP messages. It will not perform signature verification or validate the timestamp.

Note: The Custom mode only deals with checking for replay of the identifier. The policy administrator is responsible for ensuring that the identifier can be trusted and that the current time is within the time stamp created/expires times.

The custom mode allows you to create your own custom replay protection policy fragment when combined with other assertions.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click **<target>: Protect Against Message Replay** in the policy window and select **Message Replay Protection Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

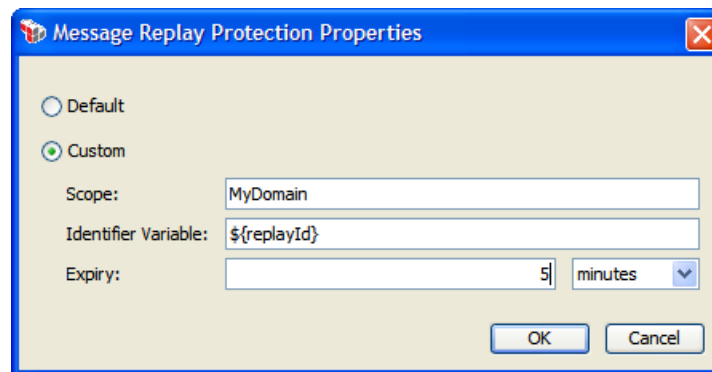




Figure 135: Message Replay Protection Properties

- Configure the properties as follows:

Table 107: Message Replay Protection settings

Setting	Description
Default Custom	Choose the mode of operation: [Default] or [Custom] . Refer to the introduction to this topic for a description of each mode. The [Default] mode replicates the functionality in the <i>WSS Replay Protection</i> assertion in versions prior to 5.2. This mode requires no further configuration.
Scope 	The replay scope lets you specify a scope for the uniqueness of the message identifier. For example, a message identifier scheme may be global, or per service, or could use some other granularity.

Setting	Description
	<p>Specify a scope for the uniqueness; context variables are permitted. Examples:</p> <p>Service scoped: <code>\${service.oid}</code></p> <p>Customer scoped: <code>Customer 7</code> (maximum 250 chars)</p> <p>Global scope: <code><leave blank></code></p> <p>Tip: The scoping can be performed by the policy author (for example, by specifying an identifier as <code>\${service.oid}/\${myId}</code>) but such an approach risks collisions if other services do not use service-scoped identifiers.</p>
Identifier Variable 	<p>Specify a context variable containing the Message ID to be processed. You can enter the variable in the format <code>\${myVar}</code> or <code>myVar</code>.</p> <p>Ensure that this Message ID has been signed and is unique.</p>
Expiry	<p>Specify how long the identifier should be saved. This expiry time is the lifetime of the message—that is, the amount of time the identifier will be stored in the cache from the time it was received. The default is 5 minutes.</p> <p>Tip: The expiry time should be greater than 0 and less than 25 days.</p>

- Click **[OK]** when done.

Require Encrypted Element Assertion

The *Require Encrypted Element* assertion is used to require that specified message elements are encrypted in the [target message](#).

You can add a Require Encrypted Element assertion for each element of the target message that you want to verify as encrypted. This assertion supports WS-Security 1.0 and 1.1.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Note: Setting the WSS recipient to one other than "Default" will cause the Require Encrypted Element assertion to always succeed.

This assertion is intended for use in a web service policy. It should be placed before the routing assertion in a policy when targeting the request message.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click the **<target>: Require Encrypted Element** in the policy window and select **Encrypted Element Properties** or double-click the assertion in the policy window. The assertion properties are displayed. The title of the dialog will show "Request", "Response", or "\${variableName}", depending on the target message.

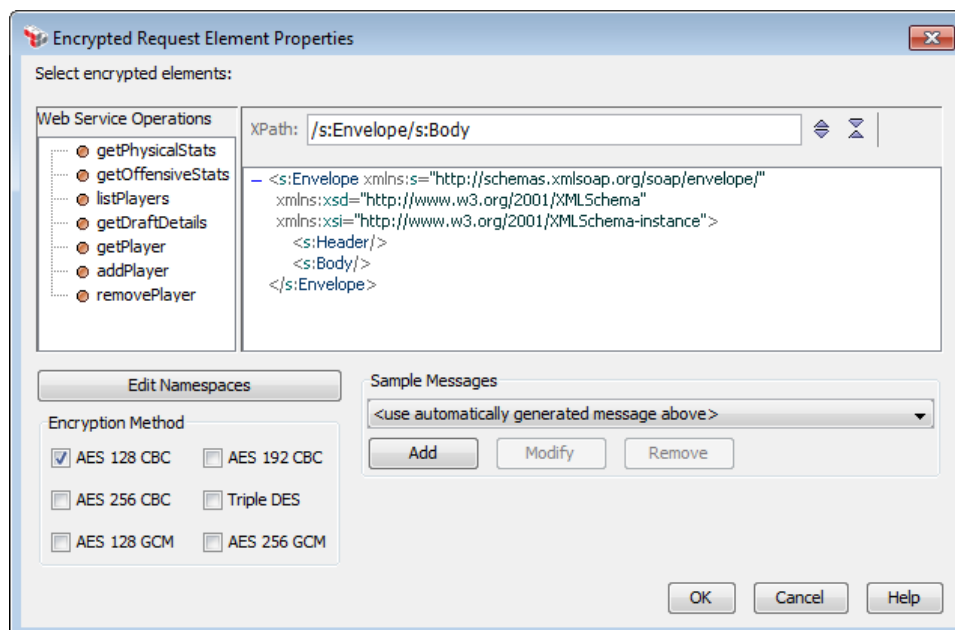


Figure 136: Encrypted Element Properties

3. Specify the XPath and indicate which element from the target message must be encrypted in the code box. For detailed instructions on using the interface to build your XPath, see "Selecting an XPath" on page 154.
4. Select the check box next to the **Encryption Methods** that may be used in the target message:

AES 128 CBC (default)
 AES 192 CBC
 AES 256 CBC
 Triple DES
 AES 128 GCM
 AES 256 GCM

Note: If your security provider does not support the "AES-GCM" encryption options, encryption/decryption attempts may fail at runtime if these options are selected.

5. Click **[OK]**.

Require Signed Element Assertion

The *Require Signed Element* assertion is used to enforce that specific message elements in the [target message](#) have been signed by the specified identity.

You can add a Require Signed Element assertion for each element of the target message that you want to verify as signed. This assertion supports WS-Security 1.0 and 1.1.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about selecting the target identity for this assertion, see "Selecting a Target Identity" on page 152.

Note: Setting the WSS recipient to one other than "Default" will cause the Require Signed Element assertion to always succeed. For more information, see "Changing the WSS Assertion Recipient" on page 146.

Note: The Require Signed Element assertion is intended for use in web service policies. If the target is the response message, ensure the assertion is placed *after* the routing assertion. If the target is the request message, the assertion should be placed *before* the routing assertion and that a credential assertion is present in the policy: [Require WS-Security Signature Credentials](#), [Require WS-Secure Conversation](#), [Require WS-Security Kerberos Token Profile Credentials](#), [Require SAML Token Profile](#), or [Require Encrypted UsernameToken Profile Credentials](#).

Context Variables Created by This Assertion

The Require Signed Element assertion sets the following context variables. **Note:** The *<prefix>* is defined in the assertion properties (Figure 137).

IMPORTANT: There is no default prefix—if no prefix is specified in the properties, then no context variables will be set by this assertion.

Table 108: Context variables created by Require Signed Element assertion

Variable	Description
<code>\${<prefix>.element}</code>	Contains the signature element.
<code>\${<prefix>.token.type}</code>	Contains the token type, retrieved from the following sources for each

Variable	Description
	<p>token type:</p> <ul style="list-style-type: none"> For Kerberos, from a WSS Kerberos assertion For SAML, from a version 1.1 or 2.0 SAML token For SymmetricKey, from an EncryptedUsernameToken or Require WS-Secure Conversation assertion For X.509, from a BinarySecurityToken, Issuer/Serial reference or SubjectKeyIdentifier reference
<code>\${<prefix>.token.element}</code>	Contains the security token element, such as a binary security token. May be empty for some token types.
<code>\${<prefix>.token.attributes.*}</code>	<p>Contains the token attributes; one variable will be created for each attribute. Note that certain attributes may be empty depending on the token type.</p> <ul style="list-style-type: none"> <code>\${<prefix>.token.element}</code>: <code>\${<prefix>.token.attributes.*}</code>: <ul style="list-style-type: none"> For the X.509 token type, the available attributes are the same as for a certificate. For the SAML token type, the following attributes may be present: <p>issuer.certificate: The certificate of the SAML issuer. subject.certificate: The certificate of the SAML subject. signing.certificate: The certificate used to sign the message.</p> <p>For each of these certificate attributes, the certificate attributes are available.</p> <p>To learn more about the certificate attributes, see "Certificate Attributes Variables" under Context Variables in the <i>Layer 7 Policy Manager User Manual</i>.</p>

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click **<target>: Require Signed Element** in the policy window and select **Signed Element Properties** or double-click the assertion in the policy window. The assertion properties are displayed. The title of the dialog will show "Request", "Response", or "\${variableName}", depending on the target message.

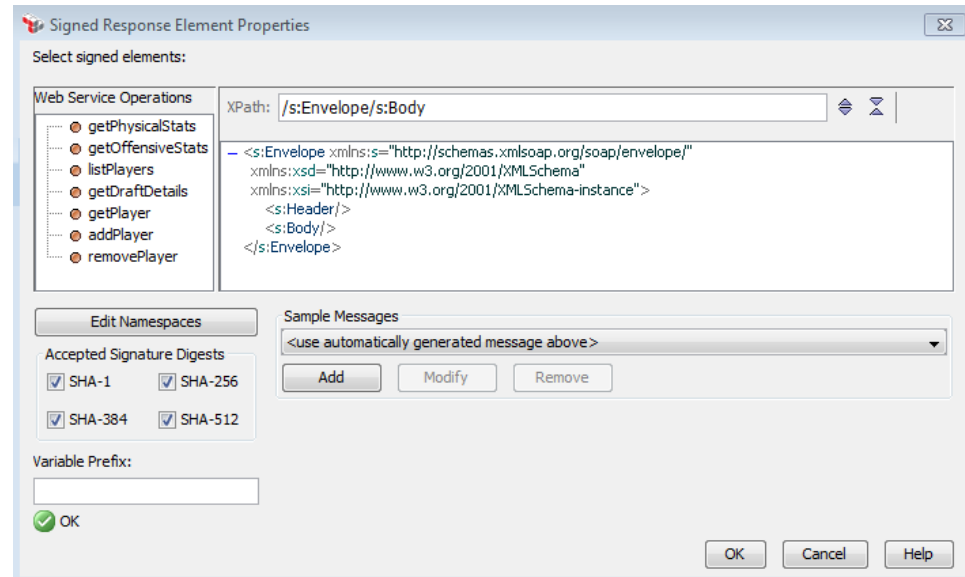


Figure 137: Signed Element Properties

3. Specify the XPath and indicate which element from the target message must be signed in the code box. For detailed instructions on using the interface to build your XPath, see "Selecting an XPath" on page 154.
4. Under **Accepted Signature Digests**, select which digest algorithms are supported in the signature. By default, all the following signature digests are accepted: **SHA-1, SHA-256, SHA-384, SHA-512**.
5. For **Variable Prefix**, enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.

For an explanation of the validation messages displayed, see Context Variable Validation in the *Layer 7 Policy Manager User Manual*.

6. Click **[OK]**.

Require Timestamp Assertion

The *Require Timestamp* assertion is used to enforce the presence of a timestamp in the target message. When this assertion is added to a policy, the Gateway will check that the timestamps adhere to all of the following conditions (all time comparisons are against the Gateway time):

- The SOAP header in the target message contains a valid `<wsu:Timestamp>` element.
- If a created date is present in the timestamp, the date is no more than one minute in the future.
- An expiry date is present in the timestamp and that date is no more than one minute in the past.
- An expiry time is present in the timestamp and the current time of the Gateway is no later than the `<wsu:Created>` time + the Maximum Expiry Time configured in this assertion or the request SOAP `<wsu:Expires>` time, whichever occurs earlier.

You can optionally specify that a security signature be required for all timestamps.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about selecting the target identity for this assertion, see "Selecting a Target Identity" on page 152.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Notes: (1) Timestamps in a request message, even if invalid or expired, are not checked unless the Require Timestamp assertion is present in a policy. 2) This assertion does not override the duration of the timestamp in the message—it simply allows a timestamp to be longer than the default 5 minutes allowed by the [Require WS-Security Signature Credentials](#) assertion. If the Securespan XML VPN Client is used to add WSS headers to the message, the timestamp duration will always be 5 minutes, regardless of what other timestamp assertions are used.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.

This assertion can be used immediately. Further configuration is not necessary unless you want to change the default settings. (Note the positioning of this assertion in the policy if you use the default setting of "Require Signature"—see Table 1.)

2. To change the settings, right-click **<target>: Require [Signed] Timestamp** in the policy window and select **Timestamp Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

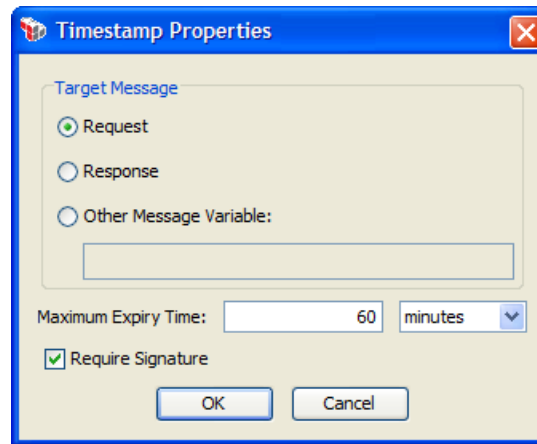


Figure 138: Timestamp Properties

3. Configure the properties as follows:

Table 109: Timestamp settings

Setting	Description
Target Message	<p>Select the message to check for a timestamp:</p> <ul style="list-style-type: none"> • Request: The request message will be checked. • Response: The response message will be checked. • Other Context Variable: A context variable will be checked. This context variable must be of type "message" and must be predefined or has been set in the policy prior to the Require Timestamp assertion. For more information on Message variables, see Context Variables in the <i>Layer 7 Policy Manager User Manual</i>.
Maximum Expiry Time	<p>Select the unit of measure from the drop-down list (milliseconds, seconds, minutes, hours), then enter the maximum permitted expiry time. Fractional measurements are permitted. An expiry time of '0' (zero) means the request expires immediately. The default is 60 minutes, with a one minute grace period.</p>
Require Signature	<p>Select this check box to require that the timestamp be digitally signed. This setting is the default.</p>

Setting	Description
	<p>If a signature is required, one of the following assertions must appear before the Require Timestamp assertion in the policy:</p> <ul style="list-style-type: none"> • Require WS-Security Signature Credentials • Require WS-Security Kerberos Token Profile Credentials • Require WS-Secure Conversation • Require Encrypted UsernameToken Profile Credentials • Require SAML Token Profile (using the "Holder-of-Key" subject confirmation method, with Require Message Signature enabled, in step 6 of the SAML Token Profile Wizard).

4. Click **[OK]** when done.

Sign Element Assertion

The *Sign Element* assertion is used to select message elements to be signed in the [target message](#).

- If the target is the *response* message, signing will occur automatically.
- If the target is the *request* message or a *message context variable*, then the [Add or Remove WS-Security](#) assertion must be added after the Encrypt Element assertion in the policy to perform the signing.

You can add a Sign Element assertion for each element of the target message that you want signed. This assertion supports WS-Security 1.0 and 1.1.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about selecting a private key for this assertion, see Selecting a Custom Private Key in the *Layer 7 Policy Manager User Manual*.

To learn more about changing the WSS Recipient for this assertion, see "Changing the WSS Assertion Recipient" on page 146.

Notes: The Sign Element assertion is intended for use in web service policies. If the target is the response message, ensure the assertion is placed *after* the routing assertion. If the target is the request message, the assertion should be placed *before* the routing assertion.

Using the Assertion

1. Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Sign Element** in the policy window and select **Sign Element Properties** or double-click the assertion in the policy window. The assertion properties are displayed. The title of the dialog will show "Request", "Response", or "\${variableName}", depending on the target message.

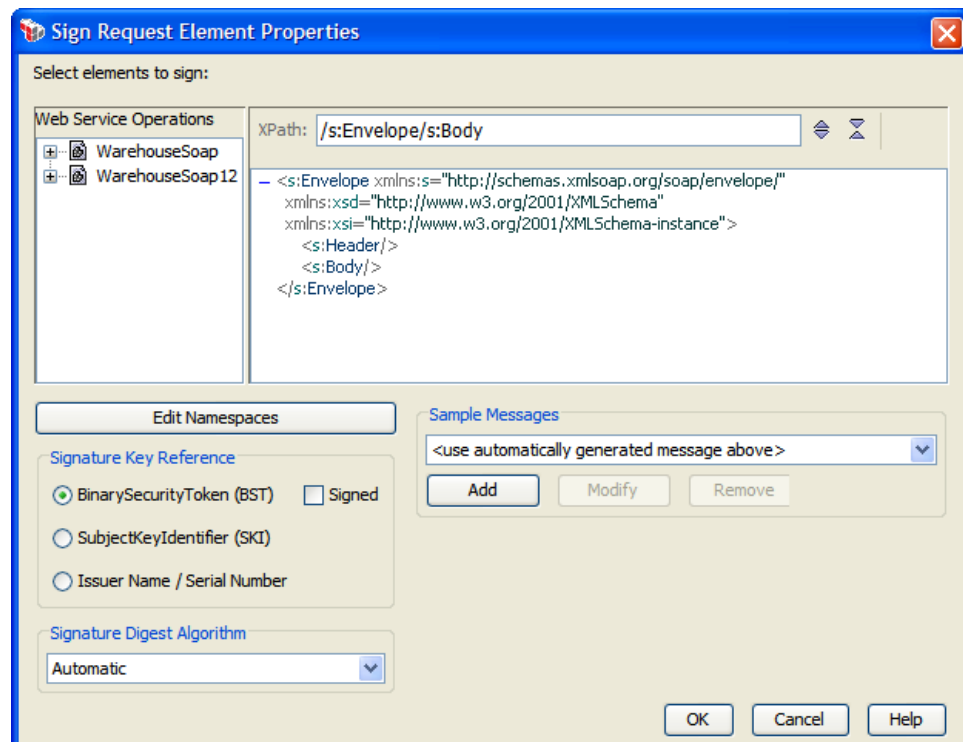


Figure 139: Sign Element Properties

3. Specify the XPath and select the target element to be sign from the code box. For detailed instructions on using the interface to build your XPath, see "Selecting an XPath" on page 154.
4. For **Signature Key Reference**, select the method to use to include the SSL certificate for the Gateway:
 - **BinarySecurityToken (BST):** The certificate is embedded within the message and does not require the recipient to already possess a copy of the signing certificate. This results in larger messages, but is more compatible. This setting is the default.

- **Signed:** Select the **Signed** check box if the BinarySecurityToken must be digitally signed.
 - **SubjectKeyIdentifier (SKI):** Use SecurityTokenReference containing the SubjectKeyIdentifier (SKI). This produces smaller messages, but at the risk of decreased compatibility.
 - **Issuer Name/Serial Number:** Use a SecurityTokenReference containing the certificates issuer distinguished name and serial number. This produces smaller messages, but at the risk of decreased compatibility.
5. For the **Signature Digest Algorithm**, select one of the following options:
- **Automatic:** The algorithm used for signature digest is determined by the *wss.decorator.digsig.messagedigest* cluster property.
 - **Any setting other than 'Automatic':** The selected digest algorithm is used, overriding the setting in the *wss.decorator.digsig.messagedigest* cluster property. The selected digest will be used for both the signature method and the digest method.

Note: If the selected combination of signing key type and digest algorithm has no corresponding signature method implementation (for example, signing with a DSA private key with any digest algorithm other than SHA-1) then the signature will fail when the decoration requirements are later applied to the message.

6. Click **[OK]** when done.

Use WS-Security 1.1 Assertion

The *Use WS-Security 1.1* assertion indicates that the policy is compliant with WS-Security 1.1. It ensures that if the policy is configured to perform WS-Security processing on the response, the Gateway will apply *SignatureConfirmation* elements to the decorated response.

A policy is WS-Security 1.1 compliant if it includes these elements:

- the Use WS-Security 1.1 assertion
- at least one WS-Security signing/encryption assertion enforced on the request (for example, [Require WS-Security Signature Credentials](#), [Require WS-Secure Conversation](#), [Sign Element](#), [Encrypt Element](#))
- at least one WS-Security signing/encryption assertion acting on the response (for example, [Add Timestamp](#), [Sign Element](#), [Encrypt Element](#))

This assertion has no impact on the Securespan XML VPN Client and it has no effect if added to a policy that does not otherwise require WS-Security.

Note: A response message will likely be rejected if 1.1-style WS-Security is expected and the Use WS-Security 1.1 assertion is *not* present in the policy.

Using the Assertion

- Add the assertion as described in "Adding an Assertion" on page 112.
The assertion is added to the policy window; no further configuration is required.

Chapter 7: Message Validation/ Transformation Assertions

Notes: (1) Depending on which Gateway product you have installed, not all the assertions shown below may be available. See Features by Product in the *Layer 7 Policy Manager User Manual* for a list of which features are available for each product. (2) This category may also include custom-created encapsulated assertions. For more information, see "Working with Encapsulated Assertions" on page 126.

In the Policy Manager, the following assertions are available in the Message Validation/Transformation category of the [Assertions] tab:

Character Encoding	412
Add or Remove XML Element(s) Assertion	414
Add WS-Addressing Assertion	416
Context Variables Created by This Assertion	416
Applying the WS-Addressing Elements	416
Signing the WS-Addressing Elements	417
Apply JSON Transformation Assertion	419
Apply XSL Transformation Assertion	424
Context Variables Created by This Assertion	425
Compress Messages to/from SecureSpan XVC Assertion	429
Customize Error Response Assertion	430
Decode MTOM Message Assertion	432
Encode/Decode Data Assertion	435
Encode to MTOM Format Assertion	437
Enforce WS-Security Policy Compliance Assertion	441
Enforce WS-I BSP Compliance Assertion	441
Enforce WS-I SAML Compliance Assertion	443
Evaluate JSON Path Expression Assertion	445
Context Variables Created by This Assertion	445
Evaluate Regular Expression Assertion	449
Context Variables Created by This Assertion	450
Evaluate Request XPath Assertion	458
Context Variables Created by This Assertion	458
Evaluate Response XPath Assertion	461
Context Variables Created by This Assertion	462
Evaluate WSDL Operation Assertion	465
Process SAML Attribute Query Request Assertion	466

Context Variables Created by This Assertion	466
Process SAML Authentication Request Assertion	472
Context Variables Created by This Assertion	472
Replace Tag Content Assertion	475
Require WS-Addressing Assertion	477
Context Variables Created by This Assertion	477
Set SAML Response Status Code Assertion	480
Translate HTTP Form to MIME Assertion	482
Translate MIME to HTTP Form Assertion	484
Validate Certificate Assertion	486
Context Variables Created by This Assertion	486
Validate HTML Form Data Assertion	488
Validate JSON Schema Assertion	490
Validate MTOM Message Assertion	493
Validate or Change Content Type Assertion	495
Validate SOAP Attachments Assertion	497
Validate XML Schema Assertion	499

The Message Validation/Transformation assertions configure the XML transformations and validation schemas applied to service messages.

Character Encoding

HTTP PUT and POST requests, as well as most HTTP responses, typically include a *Content-Type* header that declares the kind of content being returned. For text documents like XML and HTML, the Content-Type header can include an additional "encoding" parameter declaring how the characters in the content were encoded into bytes for transfer. For example, the most common Content-Type for XML documents is:

Content-Type: text/xml; charset="utf-8"

Web servers often infer the Content-Type for static files based on the file extension; some may even read the first few bytes of the file to make a more informed deduction. Occasionally, systems will send HTTP requests or responses with a Content-Type header that doesn't match the contents, either because the system is unable to extrapolate the actual type of the content, or because it has guessed incorrectly.

The [Evaluate Regular Expression](#) assertion works with characters rather than bytes, so it needs to decode the content before it can evaluate a regular expression against it. In order to decode the content, this assertion needs to know the encoding scheme that was used originally.

If the Content-Type header is missing or has no "charset" parameter, the Gateway will assume the content was encoded with ISO8859-1 (as per the RFC2616 HyperText Transfer Protocol). For content that only contains 7-bit characters (i.e., code points between U+0000 and U+007F), both UTF-8 and ISO8859-1 will encode identical bytes, so this class of error will not cause problems. However, other encodings, such as UTF-16, will still have issues.

Note: UTF-8 can encode any Unicode character, including those used in the vast majority of the world's languages, whereas ISO8859-1 is restricted to a small subset of characters, primarily ones that are relevant to Western European languages. There are many other non-Unicode character sets, each designed for use in different locales, but ISO8859-1 is the most common in North America and is the default for Microsoft Windows.

The following are examples of characters that cannot be encoded using 7 bits (ISO8859-1 encodes them using bytes with numeric values > 127, whereas UTF-8 encodes them using multiple bytes):

- "smart quotes" (also known as curly quotes)
- *en* and *em* dashes (not dashes or hyphens)
- copyright © and trademark ® ™ symbols
- accented characters
- currency symbols other than \$

Summary

If the assumed or declared encoding is ISO8859-1, the [Evaluate Regular Expression](#) assertion will never fail due to a character conversion error, because any byte can be decoded into a valid ISO8859-1 character. However, if the content is assumed or declared to be ISO8859-1 but the content was actually encoded with UTF-8 and contains non-7-bit characters, the document may be silently corrupted.

In this case, enter "UTF-8" in the *Override character encoding* field to correctly decode the content.

On the other hand, if the content is assumed or declared to have been encoded with UTF-8, but the content actually contains 8-bit ISO8859-1 characters, the Gateway will likely throw an exception during the decoding process and the Evaluate Regular Expression assertion will fail, since UTF-8 has a prescribed syntax for non-7-bit characters that few ISO8859-1 sequences will match accidentally.

In this case, enter "ISO8859-1" in the *Override character encoding* field to correctly decode the content.

Add or Remove XML Element(s) Assertion

The *Add or Remove XML Element(s)* assertion is used to add or remove XML element to or from a target message—for example, individual signatures. An example context variable input might be "requestXPath.elements" from an XPath expression (note the trailing "s" in elements is mandatory). You can also reference specific index positions of multi-valued context variables (for example, "requestXPath.elements[0]" to reference the first entry in the variable).

This assertion is intended to be used with context variables set by the following assertions: [Sign Element](#), [Evaluate Request XPath](#), or [Evaluate Response XPath](#).

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Note: The Add or Remove XML Element(s) assertion will not work on messages that have been modified by the "Apply XSL Transformation Assertion" on page 424. The audit "Message is not XML" will be recorded if you try.

Using the assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **Add or Remove XML Element Properties** automatically appear; when modifying the assertion, right-click **<target>: Add or Remove XML Elements** in the policy window and select **Add or Remove XML Elements Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

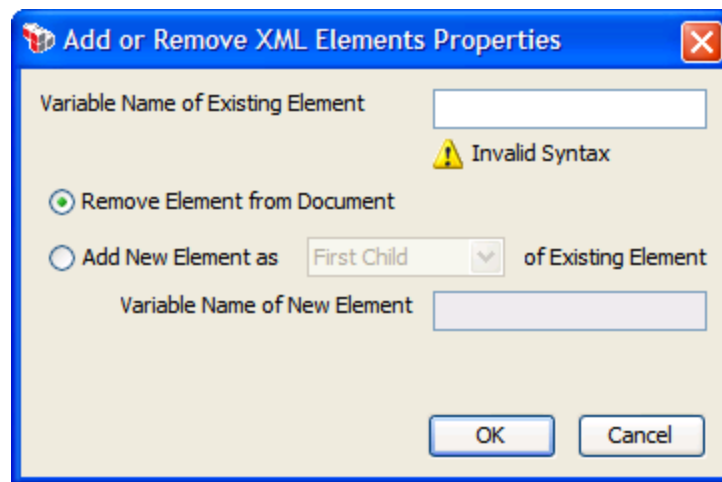




Figure 140: Add or Remove XML Elements Properties

3. Configure the properties as follows.

Table 110: Add or Remove XML Elements settings

Setting	Description
Variable Name of Existing Element 	<p>If you are adding a new XML element, enter the name of the context variable containing the existing XML element to which you are adding the new element.</p> <p>If you are removing an element, specify the context variable that contains a reference to remove a specific signature or other element (s). This variable must have been set by one of the assertions listed at the beginning of this topic.</p>
Remove Element from Document	<p>Select this option to remove the XML element specified in the Variable Name of Existing Element field.</p>
Add New Element as _____ of Existing Element	<p>Select this option to add a new XML element, then select the position in which the new element will appear:</p> <p>First Child Last Child Previous Sibling Next Sibling</p> <p><i>Example:</i></p> <pre> <Parent> <PreviousSibling/> <ExistingElement> <FirstChild/> <LastChild/> </ExistingElement> <NextSibling/> </Parent> </pre>
Variable Name of New Element 	<p>When adding an element, enter the context variable containing the element to be added. This variable can contain an element selected by one of the listed assertions at the beginning of this topic, or it can contain a text string (provided it is a valid XML document fragment).</p>

Note: The ".element" variable is not compatible with the Add or Remove XML Element(s) assertion when the XPath assertion ([Evaluate Request XPath](#) or [Evaluate Response XPath](#)) is used to capture the DOM node value.

4. Click **[OK]** when done.

Add WS-Addressing Assertion

The *Add WS-Addressing* assertion is used to add WS-Addressing elements to a target message and optionally sign them.

The WS-Addressing elements configured in this assertion's properties are added to the SOAP header of the target message. If any WS-Addressing element to be added to the target message already exists, the existing element will be removed and a new element will be added. Any existing WS-Addressing elements in the SOAP message header will not be modified unless they are overwritten by an element configured in this assertion.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Context Variables Created by This Assertion

The Add WS Addressing assertion sets the following context variables. **Note:** The default `<prefix>` is "wsa" and can be changed in the assertion properties (Figure 141).

`<prefix>.action`
`<prefix>.messageId`

Where:

- `<prefix>.action` contains the value of the *wsa:Action* value resolved at runtime
- `<prefix>.messageId` contains the message identifier resolved at runtime

Tip: The `<prefix>.action` context variable can be used in [routing assertions](#) to ensure that any outgoing SOAPAction or Content-Type action parameter matches any WS-Addressing Action property value in the message being routed.

Applying the WS-Addressing Elements

If the target message for this assertion is the request or a context variable, you need to add the [Add or Remove WS-Security](#) assertion after the Add WS-Addressing assertion in the policy for the WS-Addressing elements to be applied:

```
Request: Add WS-Addressing
Request: Apply WS-Security
```

The Add or Remove WS-Security assertion is not required if the target is the response message.

Signing the WS-Addressing Elements

To sign the WS-Addressing elements, use either of the following assertions:

- "Configure WS-Security Decoration Assertion" on page 309, with [**Sign WS-Addressing Headers**] selected in the [**Signing**] tab
- "Sign Element Assertion" on page 407, which is designed to sign any element present in a message

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **Add WS-Addressing Properties** automatically appear; when modifying the assertion, right-click **<target>: Add WS-Addressing** in the policy window and select **Add WS-Addressing Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

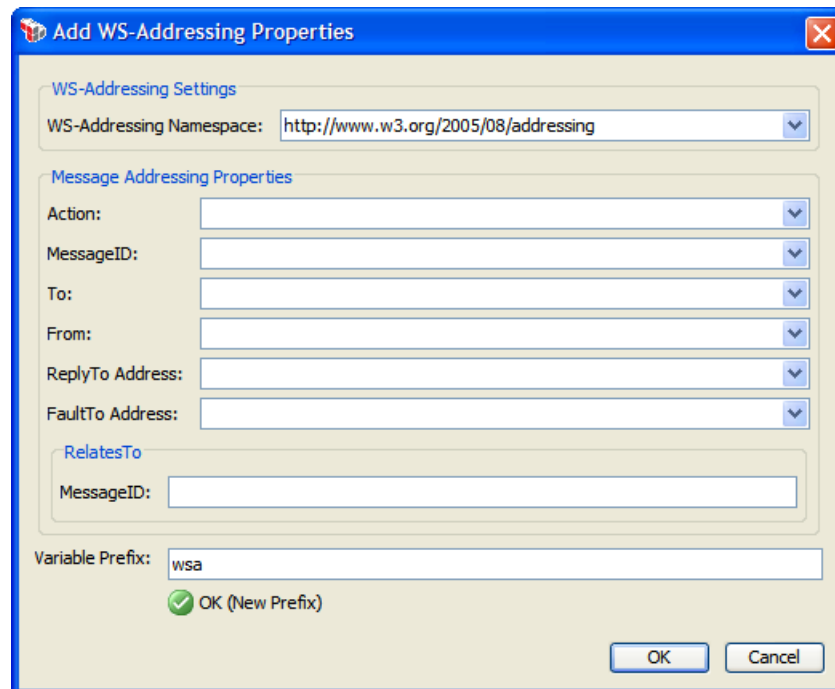


Figure 141: Add WS-Addressing Properties

3. Configure the properties as described in Table 111. Note the following tips:


- You may enter context variables of **type** String in any of the fields. 
- For fields with a drop-down list, you are free to enter your own value if a suitable value does not appear in the list. Note that any value you enter will not appear in the drop-down list (i.e., you must manually enter that value again in the future if you wish to use it).

Table 111: Add WS-Addressing settings

Setting	Description
WS-Addressing Namespace	From the drop-down list, select the WS-Addressing namespace to use or enter your own namespace.
Action	<p>Specify how to obtain the <i>wsa:Action</i> value by selecting a strategy from the drop-down list. The <i>wsa:Action</i> value is placed in the context variable <code>\${<prefix>.action}</code>.</p> <ul style="list-style-type: none"> • Obtain from target message: Select this option to first check the SOAPAction header for the Action value. If not, found, try the action Content Type parameter next; for example: <i>content-type=*;action=...</i> • Explicit from WSDL (Input): Select this option to first search for a <i>wsaw:Action</i> attribute on the <i>wsdl:input</i> element (child of <i>wsdl:portType->wsdl:operation</i>) from the WSDL. If not found, try searching for a <i>soap:operation</i> SOAPAction attribute from the WSDL next. • Explicit from WSDL (Output): Select this option to search for a <i>wsaw:Action</i> attribute on the <i>wsdl:output</i> element (child of <i>wsdl:portType->wsdl:operation</i>) from the WSDL. Unlike the option above, this option does not fall back onto the <i>soap:operation</i> SOAPAction. <p>Note: The <i>wsa:Action</i> is a required element. If no value for SOAPAction is found, then the assertion will fail.</p> <p>Tip: If you would like to employ more than one strategy to obtain the <i>wsa:Action</i> value, then add multiple instances of the assertion to the policy.</p>
MessageID	<p>Optionally enter a message identifier or select "<auto>" to have the system automatically generate a unique message ID. If left blank, the message ID will not be included in the SOAP header.</p> <p>The message ID is made available in the context variable <code>\${<prefix>.messageId}</code>.</p>
To	Optionally enter the URI of the endpoint of the message. If left blank, the endpoint will not be included in the SOAP header
From	Optionally enter the URI of the endpoint from which the message originated. If left blank, the originating endpoint will not be included in

Setting	Description
	the SOAP header
ReplyTo Address	Optionally enter the URI of the endpoint to which replies for the request message should be sent. If left blank, the ReplyTo address will not be included in the SOAP header
FaultTo Address	Optionally enter the URI of the endpoint to which fault messages should be sent. If left blank, the FaultTo address will not be included in the SOAP header
RelatesTo MessageID	Optionally specify the ID of a related message. The only relationship type currently supported is "Reply", with the value being <i>http://www.w3.org/2005/08/addressing/reply</i> . If left blank, the RelatesTo Message ID will not be included in the SOAP header
Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>The default prefix is wsa.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

- Click **[OK]** when done.

Apply JSON Transformation Assertion

The *Apply JSON Transformation* assertion lets you transform messages from JSON to XML, or from XML to JSON.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Tips: (1) If you need to transform XML to JSON, consider the "Apply XSL Transformation Assertion" on page 424 instead for greater control. In most cases, a JSON to XML transformation would be followed by an the "Apply XSL Transformation Assertion" on page 424 to produce more sophisticated results. (2) Place a "Protect Against JSON Document Structure Threats Assertion" on page 678 before this assertion to protect against DOS attacks.

Using the Assertion

- Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the JSON Transformation Properties automatically appear; when modifying the assertion, right-click **<target>: Apply JSON Transformation** in the policy window and choose **JSON Transformation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

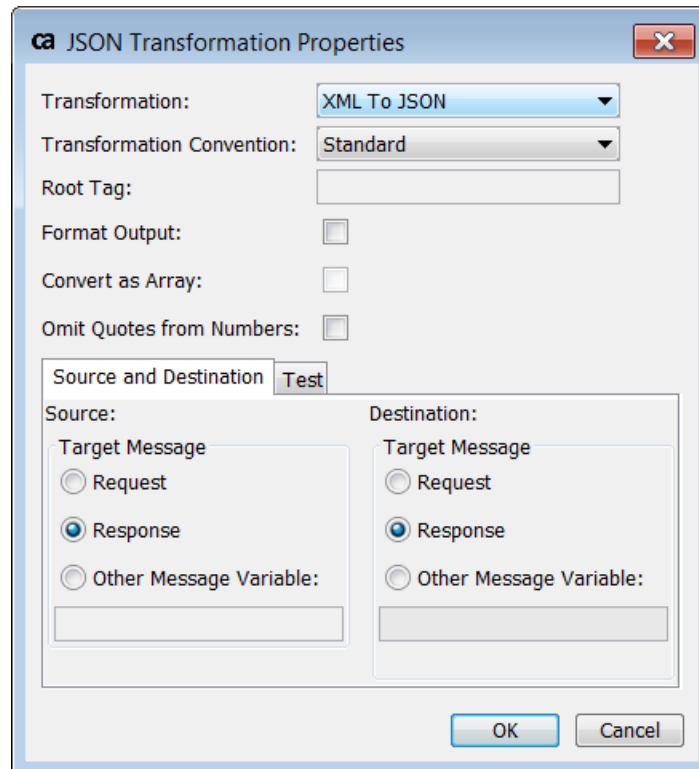



Figure 142: JSON Transformation Properties

3. Configure the properties as follows:

Table 112: JSON Transformation Settings

Setting	Description
Transformation	<p>Choose the type of transformation:</p> <ul style="list-style-type: none"> • JSON To XML: Transforms JSON to XML. • XML To JSON: Transforms XML to JSON. <p>Tip: For transforming XML to JSON, consider the Apply XSL Transformation assertion for greater control.</p>

Setting	Description
Transformation Convention	<p>Choose the convention to use for the transformation to control the appearance of the transformed JSON object:</p> <ul style="list-style-type: none"> • Standard: This option should be used for one-way conversions from JSON to XML or (basic) XML to JSON. Be aware that more complex XML structures (such as namespace, attributes) may not be transformed correctly. Do not use this option if round-tripping is required (that is, JSON>XML>JSON, or XML>JSON>XML). <p>Tip: For greater flexibility in XML-to-JSON transformations, consider the "Apply XSL Transformation Assertion" on page 424.</p> <ul style="list-style-type: none"> • JSONML: This option provides a more robust conversion of XML to JSON and is recommended if round-tripping is required (XML>JSON>XML). The JSONML convention contains rules for mapping more complex XML structures to simpler JSON structures and can offer a lossless conversion. <p>Note: For more details on choosing a convention to use, see "Choosing a Transformation Convention" below.</p>
Root Tag 	<p>Enter the root tag for the transformation convention. The root tag is the root element name of the output XML message. You may reference context variables.</p> <p>The root tag text field is only enabled when the operation is "JSON to XML" and the transformation convention is "Standard".</p>
Format Output	<p>Select this check box to format the transformed data to be human readable.</p> <p>Clear this check box to retain the native formatting of the transformed data for reduced payload size. This setting is the default.</p>
Convert as Array	<p>Select this check box to convert the XML document/fragment (using the JSONML convention) into JSON in "array form".</p> <p>Clear this check box to convert the XML document/fragment in "object form".</p> <p>For more information, see http://www.jsonml.org/syntax/.</p>
Omit Quotes from Numbers	<p>Select this check box to not enclose numbers within quotes; for example: {"test": 123}. This reproduces Gateway behavior prior to version 7.1.</p> <p>Clear this check box to enclose numbers within quotes; for example: {"test": "123"}. This is the default.</p> <p>Note: The Gateway will omit quotes only for values that will not</p>

Setting	Description
	have their string representation modified by a round trip conversion to a numeric type and back. For example, the value 1234.5678 will be emitted without quotes, but the value 123456789.1234 will be quoted because if it were treated as a number its string representation would change to "1.234567891234E8".

Choosing a Transformation Convention

Be aware that the JSON file format differs depending on whether the Standard or JSONML conventions are used. For example:

- XML to JSON using "Standard" will create an output in the standard JSON format. This output can be used in later applications where standard JSON is expected, but it will fail if the later application expects JSONML format.
- XML to JSON using "JSONML" will create an output in the JSONML format. This output can be used in later applications where JSONML is expected, but it will fail if the later application expects the standard JSON format.
- JSON to XML using "Standard" will expect the input to be in standard JSON format. If it is in JSONML format, the assertion will fail.
- JSON to XML using "JSONML" will expect the input to be in the JSONML format. If it is in standard JSON format, the assertion will fail.

Note that this has implications if you intend to do a round-trip conversion: the output from the first conversion must match the expected input format of the returning conversion in the round trip, otherwise the Apply JSON Transformation assertion will produce unexpected results or will fail.

4. Choose the targets for both the **Source** and the **Destination** of the transformation.
 - **Request:** Transformation will be applied to the request message.
 - **Response:** Transformation will be applied to the response message.
 - **Other Message Variable:** Transformation will be applied to the specified context variable. This context variable must be of type "message" and must be predefined or has been set in the policy prior to the Apply JSON Transformation assertion. For more information on Message variables, see "Context Variable Data Types" under "Context Variables" in the *Layer 7 Policy Manager User Manual*.

Tip: The message target can also be set outside of the assertion properties. For more information, see "Selecting a Target Message" on page 153.

```
<?xml encoding="UTF-8">
<rootTag>
  <firstname>Bob</firstname>
  <lastname>Smith</lastname>
</rootTag>
```

- d. Select the **Format Output** check box to improve the readability of the output. Clear this check box to display the output in its internal representation.
- e. Paste the input code in the **Test Input** box.
- f. Click [**Test**]. Examine the results in the **Resulting Output** box to see if this is what you intended.

Note: When using the "Standard" transformation convention for JSON to XML, a null value is treated the same as the string "null".

6. Click [**OK**] when done.

Apply XSL Transformation Assertion

The *Apply XSL Transformation* assertion lets you define or specify an XSL stylesheet using the XSL Transformations (XSLT) language. You can define a stylesheet any of the following ways:

- **Configured in advance:** Hard code a stylesheet in this assertion; this is embedded within the policy and is always used for transformations
- **Monitor a URL:** Policy Manager will monitor a specific URL and download the latest stylesheet
- **Fetch URL during process:** The stylesheet will be fetched from a URL within a message.

The XSLT stylesheet specified in this assertion is applied to the message when the policy is run. The transformed message is the one that is processed by subsequent policy assertions and is eventually forwarded to the requestor (response) or service (request).

A policy can contain an Apply XSL Transformation assertion for both request and response messages. If the assertion is for a *Request*, position it *before* the routing assertion; if it is for a *Response*, position it *after* the routing assertion.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Notes: (1) For security reasons, the Apply XSL Transformation assertion does not support XSL stylesheets or documents originating from external URIs (it ignores calls using the *import* or *include* elements or the *document()* function). For more information about these elements, please refer to <http://www.w3.org/TR/xslt>. (2) The Apply XSL Transformation assertion will use the encoding of the current message for transformations; the XSL output encoding is ignored. (3) To reference a context variable from this assertion, that variable must be defined in an `<xsl:param>` element near the top of the stylesheet, within the `<XSL:stylesheet>` element.

Context Variables Created by This Assertion

The Apply XSL Transformation assertion sets information about the `<xsl:message>` element in the following context variables:

`<prefix>.messages`
`<prefix>.messages.first`
`<prefix>.messages.last`

Where:

- `<prefix>` is defined in the assertion properties (default: **xslt**)
- **messages** returns all the message elements in a multivalued context variable; individual elements are available using indexing
- **messages.first** returns the first message element (this is a "shortcut" variable that is equivalent to `${<prefix>.messages[0]}`)
- **messages.last** returns the last message element (this is a shortcut variable that is equivalent to `${<prefix>.messages[n]}`, where 'n' is the index of the last item, if known ahead of time)

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the XSL Transformation Properties automatically appear; when modifying the assertion, right-click **<target>: Apply XSL Transformation** in the policy window and choose **XSL Transformation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

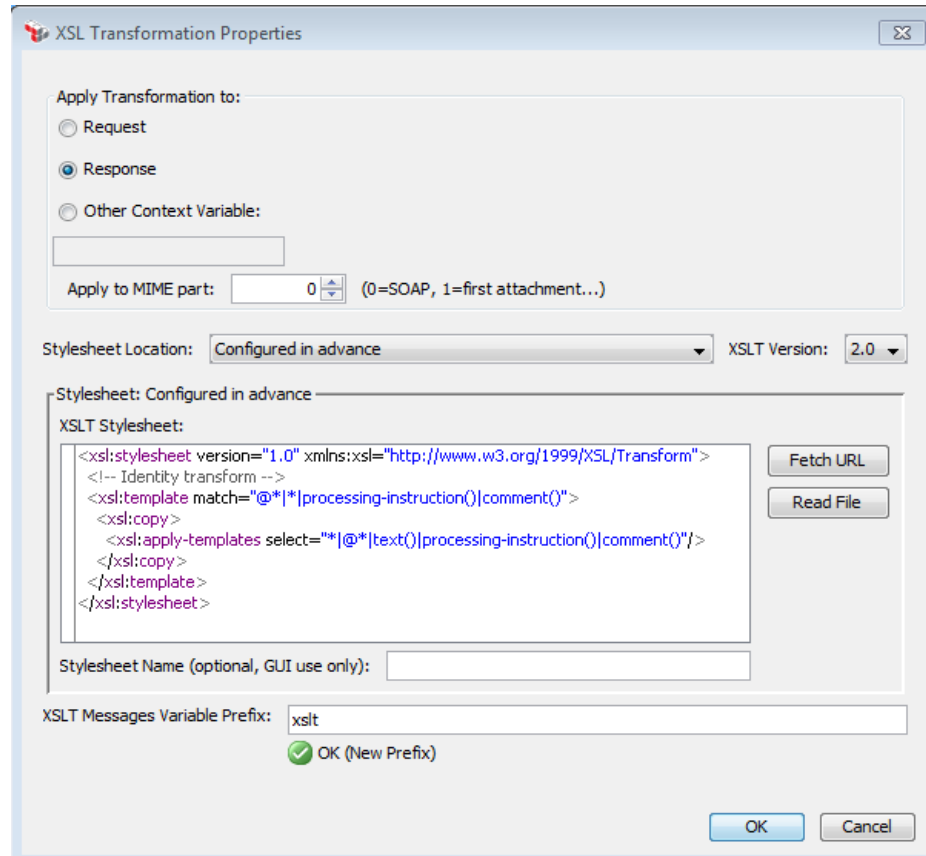


Figure 144: XSL Transformation Properties

3. Choose the target of the transformation:
 - **Request:** Transformation will be applied to the request message.
 - **Response:** Transformation will be applied to the response message.
 - **Other Context Variable:** Transformation will be applied to the specified context variable. This context variable must be of type "message" and must be predefined or has been set in the policy prior to the Apply XSL Transformation assertion. For more information on Message variables, see "Context Variable Data Types" under "Context Variables" in the *Layer 7 Policy Manager User Manual*.

Tip: The message target can also be set outside of the assertion properties. For more information, see "Selecting a Target Message" on page 153.


4. Indicate which part of the request will be affected by the transformation under the **Apply to MIME part** setting:


- Enter **0** to transform the SOAP message only
- Enter **1** to transform the first attachment to the message only
- Enter higher values to transform the **N**th attachment to a message (2 = transform second attachment, 3 = transform third attachment, etc.)

Note: The Apply XSL Transformation assertion transforms only one MIME part at a time. You can add multiple assertions to the policy if there are several MIME parts to transform.

5. From the **Stylesheet Location** drop-down list, specify where the stylesheet is coming from:

Table 113: Transformation stylesheet locations

Stylesheet Location	Description
<p>Configured in advance</p> 	<p>Choose this option to define an XSLT stylesheet that will be embedded in the policy and used for all transformations.</p> <ol style="list-style-type: none"> 1. Specify the transformation using any of the following methods: <ul style="list-style-type: none"> • Manually type the code into the XSLT Stylesheet box or copy and paste the code from another source. • Load the transformation from a URL by clicking [Fetch URL] and then typing in the URL. <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> • Load the transformation from a local file by clicking [Read File] and then browsing to the appropriate file. <p>Note: The XSLT stylesheet maximum size is controlled by the <i>xslDownload.maxSize</i> cluster property.</p> 2. Review the content of the XSL Stylesheet box and edit if necessary. You can right-click within the box for some useful tools to help you edit. For more information, see "Using the XML Editor" on page 159. <p>Tip: You can use context variables in this assertion by inserting <code>xsl:param</code> statements into the XSL Stylesheet box. The values of the matching context variables will be passed to the compiled stylesheet. For example, to use the variable <code>\${foo}</code> inside a stylesheet, use this syntax:</p> <pre><xsl:param name="foo" select=""/> ... <xsl:variable name="bar" select="\$foo"/></pre> 3. Optionally enter a name for the transformation in the Stylesheet Name field. This name will appear next to the assertion name in the policy development window. This will help you recognize the

Stylesheet Location	Description
	transformations more easily.
Monitor URL for latest value 	<p>Choose this option to continuously monitor a location to ensure the latest stylesheet is used. Type the address in the URL to monitor field. The URL may contain context variables that will be resolved at run time. By default, Gateway will download the stylesheet from this address every 5 minutes.</p> <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> <p>Note: The XSLT stylesheet maximum size is controlled by the <code>xslDownload.maxSize</code> cluster property.</p>
Fetch from URL in processing instruction <i>(applicable only to non-SOAP messages)</i>	<p>Choose this option to have Gateway retrieve the stylesheet from a recognizable URL within a non-SOAP message.</p> <ol style="list-style-type: none"> Define a list of regular expressions ("Regex") to ensure that stylesheets are retrieved from only legitimate URLs: <ul style="list-style-type: none"> To add an expression, click [Add] and then enter a regular expression To modify an expression, choose the expression and then click [Edit] To remove an expression, choose the expression and then click [Remove]. <p>You can add multiple expressions if necessary. You must define at least one expression.</p> <p>Example</p> <p>You can enter an explicit URL, such as</p> <p><code>http://somedomain.com/xsl/stylesheet.xsl</code> or you can enter a regex expression such as <code>http[s]?://.*?/**.xsl</code></p> <p>This expression will allow:</p> <p><code>http://hugh.l7tech.com/xsl/OrderProduct-request.xsl</code></p> <p>But it will disallow:</p> <p><code>http://hugh.l7tech.com/xsl/harmless.xsl</code></p> <p>To learn about regular expressions, visit http://www.regular-expressions.info/</p> In the Allow messages with no stylesheet declaration check box, indicate whether stylesheet references are mandatory. <ul style="list-style-type: none"> Select this check box to permit SOAP messages to have no stylesheet declarations Clear this check box to enforce that every SOAP message must have a URL pointing to an XSLT stylesheet to transform the message. If none is found, a SOAP fault is generated.

6. Select the appropriate XSLT version from the drop down.
7. Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.

The default variable prefix is **xslt**.

For an explanation of the validation messages displayed, see Context Variable Validation in the *Layer 7 Policy Manager User Manual*.

8. Click **[OK]**.

Compress Messages to/from SecureSpan XVC Assertion

The *Compress Messages to/from SecureSpan XVC* assertion is used to indicate that messages going to or coming from the Securespan XML VPN Client should be compressed using the *gzip* algorithm. Compressing XML/SOAP messages can decrease the transfer times and improve performance, especially for large HTTP payloads.

Message routing will proceed as normal when compression is in effect. The Gateway will decompress the payload upon reception, then compress the response payload before returning the message to the Securespan XML VPN Client.

When compression is in effect, the Gateway uses the *uncompressed* message size to validate against the set limit (as defined by the *io.xmlPartMaxBytes* cluster property).

The Compress Messages to/from SecureSpan XVC assertion will always succeed, unless it is configured to fail if it receives an uncompressed request. If you do not want your Gateway to accept compressed messages, modify the *request.compress.gzip.allow* cluster property.

Note: Messages are compressed only when submitted by the Securespan XML VPN Client, or if the service endpoint is another Gateway. For more information, see the [Request HTTP Rules] tab of the "Route via HTTP(S) Assertion" on page 529.

For more information, see *Configuring HTTP Compression* in the Securespan XML VPN Client documentation.

Using the Assertion

1. Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Compress Messages to/from SecureSpan XVC** in the policy window and select **Compression Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

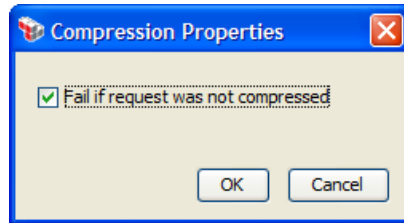


Figure 145: Compression Properties

3. Select the check box to cause the assertion to fail if the request could not be compressed for whatever reason (this will be noted in the audit logs). Clear the check box to always have the assertion succeed.
4. Click **[OK]** when done.

Customize Error Response Assertion

The *Customize Error Response* assertion lets you configure the error response for a service. You can select the following responses when an error is encountered:

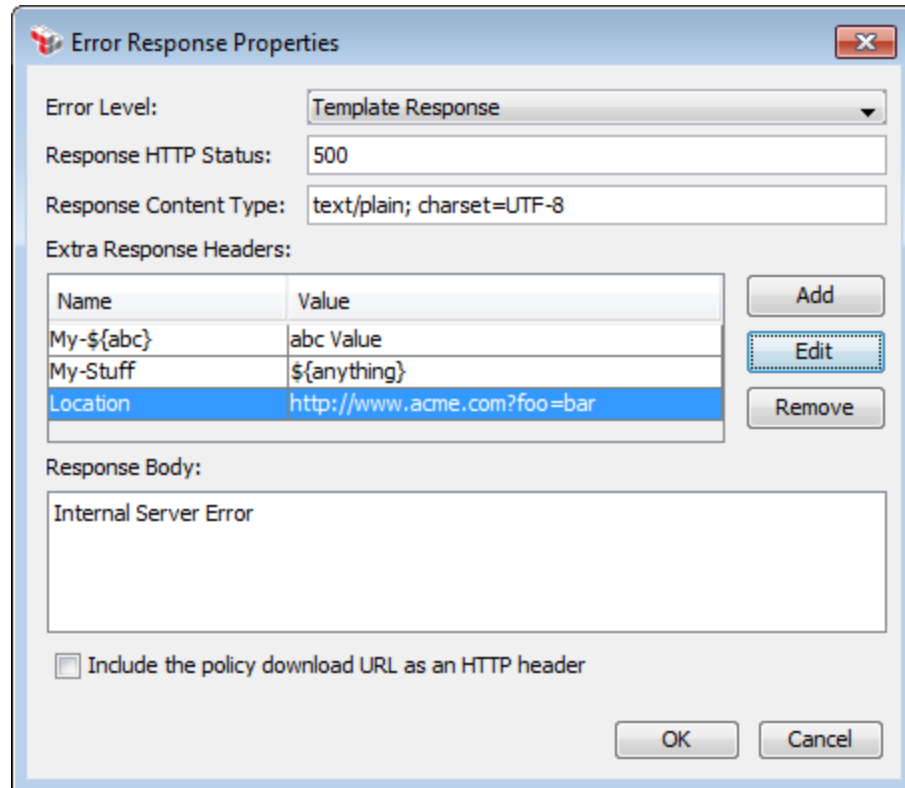
- **Template response:** Lets you define your own message to be returned. This is the default.
- **Drop connection:** When the policy fails, simply drop the connection without providing any response.

Tip: For SOAP services, consider using the "Customize SOAP Fault Response Assertion" on page 607 as well, as it contains SOAP-specific settings.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.

2. Right-click **Customize Error Response** in the policy window and select **Error Response Properties** or double-click the assertion in the policy window. The assertion properties are displayed.



The dialog box titled "Error Response Properties" contains the following fields and controls:

- Error Level:** A dropdown menu set to "Template Response".
- Response HTTP Status:** A text field containing "500".
- Response Content Type:** A text field containing "text/plain; charset=UTF-8".
- Extra Response Headers:** A table with two columns: "Name" and "Value".



Name	Value
My- <code>{abc}</code>	abc Value
My-Stuff	<code>{anything}</code>
Location	http://www.acme.com?foo=bar



 To the right of the table are buttons for "Add", "Edit", and "Remove".
- Response Body:** A large text area containing the text "Internal Server Error".
- ☐ **Include the policy download URL as an HTTP header**
- At the bottom right are "OK" and "Cancel" buttons.

Figure 146: Error Response Properties

3. Configure the error response:

Table 114: Error Response settings

Setting	Description
Error Level	Select the response to use for an error: <ul style="list-style-type: none"> • Template Response: The assertion will return the response in the Response Body. • Drop Connection: The connection is simply dropped, with no response ("stealth" mode).
Response HTTP Status 	Enter the HTTP status code to return. Alternatively, you may enter a context variable that contains the status code.
Response Content-Type 	Enter the valid Content-Type header to return. Alternatively, you may enter a context variable that contains the Content-Type.

Setting	Description
Extra Response Headers 	<p>Optionally specify one or more custom HTTP headers to be inserted into the custom error response (for example, to transmit application-specific metadata).</p> <p>Note: Be sure you are aware of the consequences of adding headers to a message. If in doubt, please contact your system administrator.</p> <p><i>To add a header:</i></p> <ol style="list-style-type: none"> 1. Click [Add]. 2. Enter the Name and Value of the header. You may reference context variables in both fields. 3. Click [OK]. The header is added to the list. <p><i>To modify a header:</i></p> <ol style="list-style-type: none"> 1. Select the header to modify and then click [Edit]. 2. Modify the Name or Value as required. 3. Click [OK]. <p><i>To remove a header:</i></p> <ul style="list-style-type: none"> • Select the header to remove and then click [Remove].
Response Body 	<p>Enter the text to be returned in the response. This text will be returned as entered; the assertion performs no validation. You may include context variables in the response body to provide additional informational information about the error.</p> <p>Note: The Response Body does not support use of a Document Type Definition (DTD) for XML content.</p>
[Include the policy download URL as an HTTP header]	<p>Select this check box to indicate that the policy download URL is included as an HTTP header in the response if it is required.</p> <p>Clear this check box to not include the policy download URL in the HTTP header in a response.</p>

4. Click **[OK]** when done.

Decode MTOM Message Assertion

The *Decode MTOM Message* assertion is used to process MTOM-optimized messages and change them to regular SOAP messages. MTOM-optimized messages are based on the Message Transmission Optimization Mechanism (MTOM) specification.

The target message for this assertion can be selected from within the assertion properties or by right-clicking the assertion in the policy window and choosing "Select Message Target". For more information on the latter, see "Selecting a Target Message" on page 153.

Tip: The "target message" in this assertion is the message that is being decoded. In the assertion properties, this is the "Source Message". The "Target Message" in the properties is simply the destination to hold the decoded message.

MIME Multipart Messages

A SOAP service normally does not permit MIME multipart messages unless it is explicitly specified in the service's WSDL document or the policy contains an assertion that processes MIME multipart. Adding a Decode MTOM Message assertion that targets a request message in a policy will mean that the service(s) for that policy will permit MIME multipart messages.

MTOM Messages and WS-Security

By default, the Gateway will automatically decode MTOM-encoded messages if the message contains a WS-Security header that will be processed by the Gateway. The decoder will remove the packaging, leaving a regular SOAP message for WS-Security processing. The decoded MTOM message must be smaller than the permitted size for the XML part of a message.

For more information on the cluster properties involved, refer to the following properties in Gateway Cluster Properties in the *Layer 7 Policy Manager User Manual*:

mtom.decodeSecuredMessages
io.xmlPartMaxBytes

Note: Attachments to an decoded MTOM message will be approximately 1/3 larger after decoding due to its Base64-encoding. This may cause the overall message size to exceed the limit specified by the *io.xmlPartMaxBytes* cluster property.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Decode MTOM Message** in the policy window and select **MTOM Decode Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

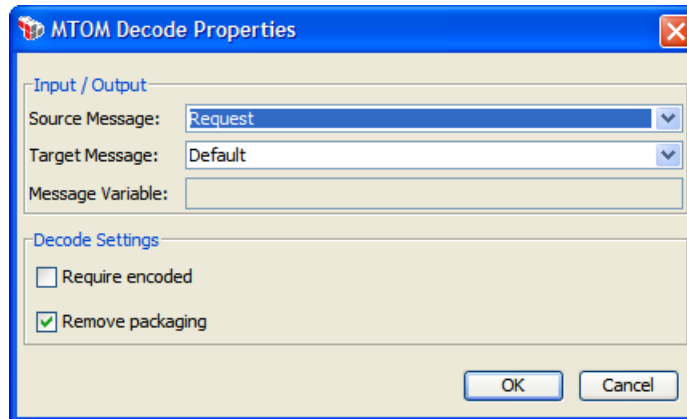



Figure 147: MTOM Decode Properties

3. Configure the properties as follows:

Table 115: MTOM Decode Settings

Setting	Description
<i>Input/Output</i>	
Source Message	<p>Specify which message should be decoded: Request, Response, or a message context variable, if one has been defined by this point in the policy. Message variables will appear as "\${variable_name}" in the drop-down list.</p> <p>Tip: The Source Message can also be changed by right-clicking the assertion in the policy window and choosing "Select Target Message". For more information, see "Selecting a Target Message" on page 153.</p>
Target Message	<p>Specify the output message to hold the results of the decoding. Choose from:</p> <ul style="list-style-type: none"> • Default: Place the result back into the source message. • Request: Place the result into the request message. • Response: Place the result into the request message. • Message Variable: Place the result into the message context variable specified below.
Message Variable 	<p>If the Target Message is "Message Variable", enter the name of the message context variable here, in the format: \${variable_name}. If the variable does not already exist, it will be created.</p>
<i>Decode Settings</i>	
Require encoded	<p>Select this check box to require that the source message be MTOM-encoded, otherwise the assertion will fail.</p> <p>Clear this check box to not fail the assertion if the source message is</p>

Setting	Description
	not MTOM-encoded.
Remove packaging	<p>Select this check box to turn the message into a regular SOAP format. Clear this check box to place any attachment into the message but leave the message in MTOM format.</p> <p>This option is useful if you want to route a message in MTOM format but want to (for example) perform an XSL transformation or XML Schema validation on the message (including the attachment data).</p> <p>Tip: When a message is decoded, the XML message size is restricted by the <i>io.xmlPartMaxBytes</i> cluster property. If the message is too large to be converted, then the assertion will fail.</p>

- Click **[OK]** when done.

Encode/Decode Data Assertion

The *Encode/Decode Data* assertion is used to encode data to and from Base64 and URL Encoded formats.

- **For Base64:** This assertion can encode variables of type String, Message, or X.509 Certificate into a String or Message Base64-encoded variable. It can decode a String or Message variable into a String, Message, or X.509 Certificate variable.
- **For URL Encoding:** This assertion supports encoding/decoding to/from a String or Message variable.

To see the results of the encoding or decoding, audit the output variable using the "Add Audit Detail Assertion" on page 600.

Note: Encoding and decoding will only be performed on the main (first) part of a Message variable. Other MIME parts can be decoded using the 'parts' variables, but cannot be the target of decoding. For example, use `${message.parts.2.body}` to decode the body from the second MIME part of the message. For more information on the 'parts' variables, see "Transport Layer Variables" under Context Variables in the *Layer 7 Policy Manager User Manual*.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.



- Right-click the assertion in the policy window and select **Encode/Decode Data Properties** or double-click the assertion in the policy window. The assertion properties are displayed. (Note that depending on how the assertion is configured, it may appear in the policy as "**Base64 Encode...**", "**Base64 Decode...**", "**URL Encode...**", or "**URL Decode...**".)

Figure 148: Encode/Decode Properties

- Configure the properties as follows:

Table 116: Encode/Decode Data settings

Setting	Description
Encode/Decode	<p>From the drop-down list, select the operation you wish to perform:</p> <ul style="list-style-type: none"> Base64 Encode: Select this to encode any supported variable type (String, Message, X.509 Certificate) into a String or Message Base64 encoded variable. Base64 Decode: Select this to decode a String or Message variable into a String, Message of any supported Content-Type, or X509 Certificate. Base16 (hex) Encode: Select this to use hex (Base16) encoding. Base16 (hex) Decode: Select this to use hex (Base16) decoding. URL Encode: Select this to encode to a text variable (String or Message).

Setting	Description
	<ul style="list-style-type: none"> URL Decode: Select this to decode from a text variable to a text variable.
Source Variable 	Specify the context variable that contains the input value. You may enter the variable with or without the "\${}" wrapper. Note: Multivalued context variables are not supported.
Target Variable 	Specify the context variable that will hold the output. If this variable does not already exist, it will be created. As with the Source Variable, you may include the "\${}" wrapper but this is not necessary.
<i>Target Options</i>	
Data Type	Select whether the target variable is of type String , Message , or X.509 Certificate . For more information about variable data types, see "Set Context Variable Assertion" on page 656.
Content-Type	Specify the Content-Type for the target variable. Available only when the data type is ' Message '.
<i>Encode/Decode Options</i>	
Character Encoding	Enter the character encoding to use during encode/decode.
Strict	Select this option to scan the Base64 data for illegal Base64 characters. If any are found, the assertion will fail. Clear this option to prevent the assertion from failing due to illegal Base64 characters. This option is available only when performing a ' Base64 Decode '.
Multiple lines	Select this option to break the output into multiple lines during encoding. Clear this check box to render the output in a single line.
Line break every X characters	If the output is being broken into separate lines, specify where the line break should occur in the encoded Base64 data. The default is after every 76 characters.

4. Click **[OK]** when done.

Encode to MTOM Format Assertion

The *Encode to MTOM Format* assertion is used to create optimized messages. It will convert a regular SOAP message to an optimized MIME multipart/related serialization format based on the Message Transmission Optimization Mechanism (MTOM) specification.

The target message for this assertion can be selected from within the assertion properties or by right-clicking the assertion in the policy window and choosing "Select Message Target". For more information on the latter, see "Selecting a Target Message" on page 153.

Tip: The "target message" in this assertion is the message that is being encoded. In the assertion properties, this is the "Source Message". The "Target Message" in the properties is simply the destination to hold the encoded message.

SOAP 1.2 is required for standard MTOM encoding. If a SOAP 1.1 message is encoded, it will result in a non-standard optimized message. MTOM encoding will not change the SOAP version.

Auditing MTOM Messages

When an MTOM-encoded message is audited, only the XML part is recorded. Any attachments are not included in the audited data.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Encode to MTOM Format** in the policy window and select **MTOM Encode Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

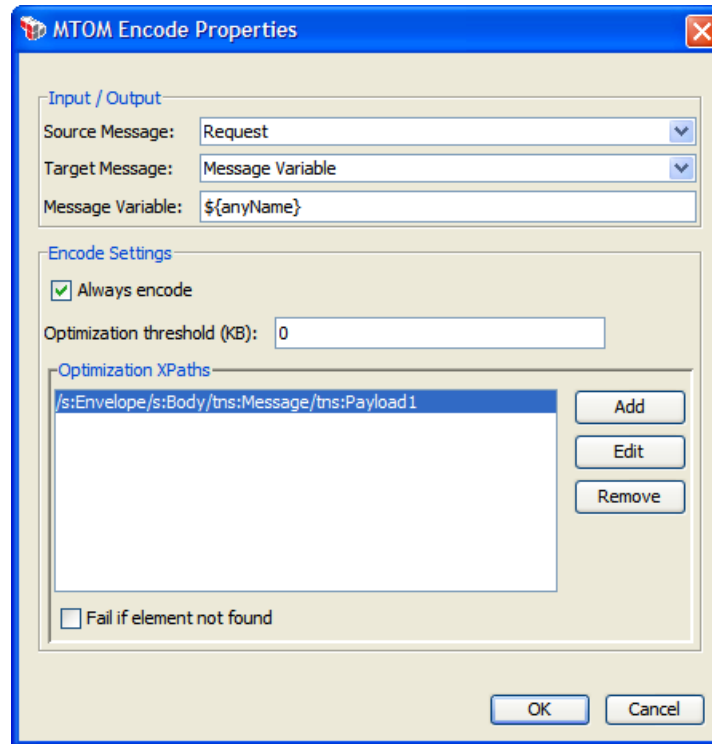




Figure 149: MTOM Encode Properties

3. Configure the properties as follows:

Table 117: MTOM Encode Settings

Setting	Description
<i>Input/Output</i>	
Source Message	<p>Specify which message should be encoded: Request, Response, or a message context variable, if one has been defined by this point in the policy. Message variables will appear as "\${variable_name}" in the drop-down list.</p> <p>Tip: The Source Message can also be changed by right-clicking the assertion in the policy window and choosing "Select Target Message". For more information, see "Selecting a Target Message" on page 153.</p>
Target Message	<p>Specify the output message to hold the results of the encoding. Choose from:</p> <ul style="list-style-type: none"> • Default: Place the result back into the source message. • Request: Place the result into the request message. • Response: Place the result into the request message. • Message Variable: Place the result into the message context variable specified below.

Setting	Description
Message Variable 	If the Target Message is "Message Variable", enter the name of the message context variable here, in the format: <code>\${variable_name}</code> . If the variable does not already exist, it will be created.
<i>Encode Settings</i>	
Always encode	Select this check box to always encode the message, even if no attachments will be created. This setting is the default. Clear this check box to skip encoding if no elements were found or if elements were found but were below the "Optimization threshold".
Optimization threshold	Optionally enter a threshold size for encoding. The MTOM encoding will not occur if each individual selected element content is smaller than this threshold <i>and</i> " Always encode " is not selected. When selected element content is larger than the threshold size, it will be optimized. When the threshold is not met, the element's content is not optimized.
Optimization XPaths 	This section lets you enter specific XPaths to select the elements containing content for optimization. <ul style="list-style-type: none"> • To add an XPath, click [Add]. See "Selecting an XPath" on page 154 for more details. • To modify an XPath on the list, select it and then click [Edit]. See "Selecting an XPath" on page 154 for more details. • To remove an XPath from the list, select it and then click [Remove].
Fail if element not found	Select this check box to have the assertion fail if any of the optimization XPaths do not match any elements. Clear this check box to not fail the assertion if no elements are found.

4. Click [**OK**] when done.

Enforce WS-Security Policy Compliance Assertion

The *Enforce WS-Security Policy Compliance* assertion helps you construct a policy that is compliant with the WS-Security Policy specifications. When this assertion is present, the policy validation window will warn you of any WS Security Policy errors when any of the following assertions are present in the policy:

- Encrypt Element (header, body only)
- Require Timestamp
- Require WS-Security Signature Credentials
- Require WS-Security UsernameToken Profile Credentials
- Require SSL or TLS Transport
- Require SSL or TLS Transport with Client Authentication
- Sign Element (header, body only)

Note: When you download a WSDL of a policy containing the Enforce WS-Security Policy Compliance assertion, the WS-Security Policy is attached. If the original WSDL contains an attached policy, that policy is replaced by the Gateway's WS-Security Policy for the service. For more information on downloading a WSDL, see *WSDL Proxy & Policy Downloads* in the *Layer 7 Installation and Maintenance Manual*.

The Enforce WS-Security Policy Compliance assertion supports WS-Security Policy version 1.1. For more information, refer to <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>.

Using the Assertion

- Add the assertion to the policy development window. For more information, see [Adding an Assertion](#).

The assertion is added to the policy window; no further configuration is required.

Enforce WS-I BSP Compliance Assertion

The *Enforce WS-I BSP Compliance* assertion checks incoming and/or outgoing requests for compliance with the WS-I Basic Security Profile 1.0 specifications.

Use this assertion to:

- Restrict encryption, signature, algorithms, etc., to those permitted
- Ensure strict adherence to namespaces
- Enforce adherence for required/restricted elements, attributes, and attribute values

- Enforce referencing constraints (for example, reference by ID for local security tokens).

This assertion implements the rules contained in the *Basic Security Profile Version 1.0* specifications located at: <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>.

To view the audit records generated by this assertion, see Gateway Audit Events in the *Layer 7 Policy Manager User Manual*.

Note: When the Enforce WS-I BSP Compliance assertion is present in a policy path, it performs validations to help ensure compliance. For example, you will receive a validation error if an [Encrypt Element](#) assertion used AES 192 bit encryption.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Enforce WS-I BSP Compliance** in the policy window and select **WS-I BSP Compliance Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

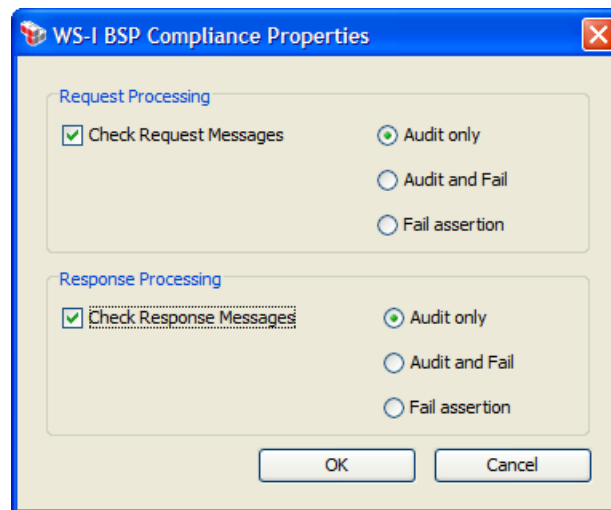


Figure 150: WS-I BSP Compliance Properties

3. Configure the properties as follows:

Table 118: WS-I BSP Compliance Properties settings

Setting	Description
Check Request Message	<p>Select this check box to check request messages for conformance to WS-I BSP specifications. Clear this check box to not check requests for WS-I BSP conformance.</p> <p>This setting is selected by default if the assertion is placed <i>before</i> the routing assertion in the policy.</p>
Check Response Message	<p>Select this check box to check response messages for conformance to WS-I BSP specifications. Clear this check box to not check responses for WS-I BSP conformance.</p> <p>This setting is selected by default if the assertion is placed <i>after</i> the routing assertion in the policy.</p>
Audit only	Select Audit only to generate an audit record when non-conformance in the request or response is detected. No SOAP fault occurs and the assertion does not fail.
Audit and Fail	Select Audit and Fail to generate both an audit record and a SOAP fault when non-conformance in the request or response is detected; the assertion also fails.
Fail assertion	Select Fail assertion to generate a SOAP fault and fail the assertion when non-conformance in the request or response is detected. No audit record is generated.

Tip: The audit record indicates the rule that was broken (Rxxxx). You can look up the rule on www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html to see more information. No audit record is created when a request or response conforms to the specifications.

- Click **[OK]** when done.

Enforce WS-I SAML Compliance Assertion

The *Enforce WS-I SAML Compliance* assertion checks incoming and/or outgoing requests for compliance with the SAML Token specifications.

Use this assertion to:

- Ensure strict adherence to namespaces
- Enforce adherence for required/restricted elements, attributes, and attribute values
- Enforce referencing constraints (for example, reference by ID for local security tokens).

This assertion implements the rules contained in the SAML Token section of the *Basic Security Profile Version 1.0* specifications located at <http://www.w3.org/Profiles/BasicSecurityProfile-1.0.html>.

To view the audit records generated by this assertion, see Gateway Audit Events in the *Layer 7 Policy Manager User Manual*.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Enforce WS-I SAML Compliance** in the policy window and select **WS-I SAML Compliance Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

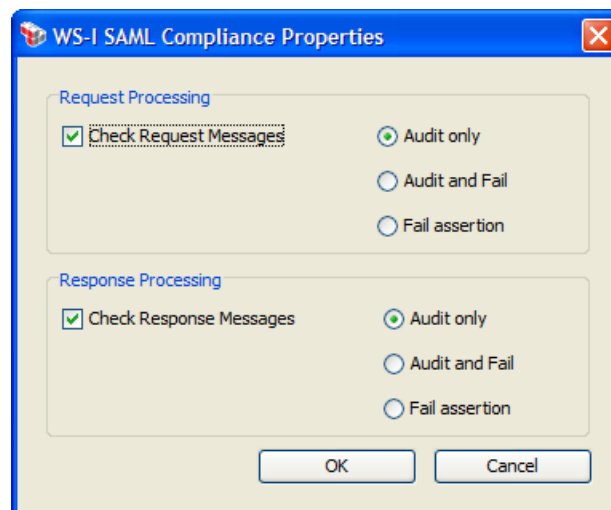


Figure 151: WS-I SAML Compliance Properties

3. Configure the properties as follows:

Table 119: WS-I SAML Compliance Properties settings

Setting	Description
Check Request Message	<p>Select this check box to check request messages for conformance to the SAML Token section of the WS-I BSP specifications. Clear this check box to not check requests for conformance.</p> <p>This setting is selected by default if the assertion is placed <i>before</i> the routing assertion in the policy.</p>

Setting	Description
Check Response Message	Select this check box to check response messages for conformance to the SAML Token section of the WS-I BSP specifications. Clear this check box to not check responses for conformance. This setting is selected by default if the assertion is placed <i>after</i> the routing assertion in the policy.
Audit only	Select Audit only to generate an audit record when non-conformance in the request or response is detected. No SOAP fault occurs and the assertion does not fail.
Audit and Fail	Select Audit and Fail to generate both an audit record and a SOAP fault when non-conformance in the request or response is detected; the assertion also fails.
Fail assertion	Select Fail assertion to generate a SOAP fault and fail the assertion when non-conformance in the request or response is detected. No audit record is generated.

Tip: The audit record indicates the rule that was broken (Rxxxx). You can look up the rule on www.ws-i.org/Profiles/SAMLTOKENProfile-1.0.html to see more information. No audit record is created when a request or response conforms to the specifications.

4. Click **[OK]** when done.

Evaluate JSON Path Expression Assertion

The *Evaluate JSON Path Expression* assertion is used to query JSON objects, similar to querying XPaths. You enter a JSON expression and this assertion will parse the target message and place the results in context variables.

Tip: Place a "Protect Against JSON Document Structure Threats Assertion" on page 678 before this assertion to protect against DOS attacks.

Context Variables Created by This Assertion

The Evaluate JSON Path Expression assertion sets the following context variables. **Note:** The default `<prefix>` is "jsonPath" and can be changed in the assertion properties (Figure 152).

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Table 120: Context variables created by Evaluate JSON Path Expression assertion

Variable	Description
<code><prefix>.found</code>	Indicates whether a match was found for the expression: <ul style="list-style-type: none"> true = Match found false = No match found
<code><prefix>.count</code>	Returns the number of matches.
<code><prefix>.result</code>	Returns the result of the match, if a single match was made.
<code><prefix>.results</code>	Returns the results of the match, if multiple matches were made.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the optional assertion, the Evaluate JSON Path Expression Properties automatically appear; when modifying the assertion, right-click **<target>: Evaluate JSON Path Expression** in the policy window and select **Evaluate JSON Path Expression Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

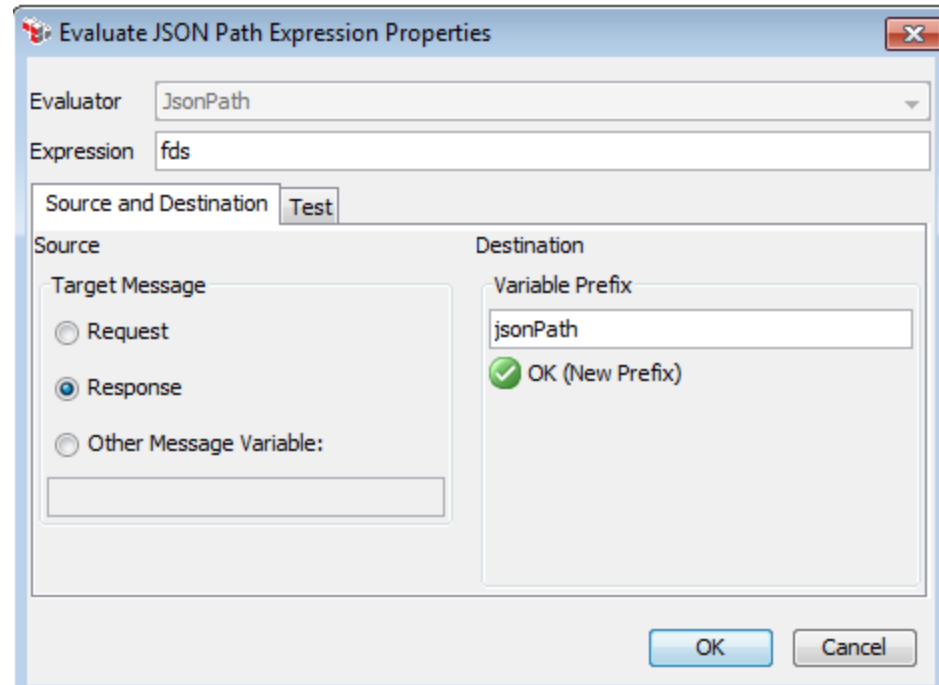



Figure 152: Evaluate JSON Path Expression Properties - [Source and Destination] tab

3. Configure the optional evaluator and expression:

Table 121: Evaluate JSON Path Expression Properties - basic settings

Setting	Description
Evaluator	Currently, the only supported evaluator is JsonPath .
Expression 	<p>Enter the expression to be matched against a message. You may reference context variables.</p> <p>Note: The Expression field can only contain a single expression. To evaluate multiple expressions, configure multiple Evaluate JSON Expression assertions within a policy.</p>

4. Configure **[Source and Destination]** tab as follows:

Table 122: Evaluate JSON Path Expression Properties - [Source and Destination] tab

Setting	Description
Target Message	<p>Specify whether to match against the Request, Response, or Other Message Variable that contains the value to analyze. If other variable, specify the variable name in the box. (You do not need to enclose the variable name within the "\${" characters.)</p> <p>Tip: The message target can also be set outside of the assertion properties. For more information, see "Selecting a Target Message" on page 153.</p>

Setting	Description
Variable Prefix	Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy. For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i> .

5. Select the **[Test]** tab to test your JSON expression against sample test input.
 - In the **Test Input** box, paste some JSON code that might be found in the target message. When you click the **[Test]** button, the assertion attempts to the specified expression against the test input.
 - The **Test Output** box shows the results of the match (see Figure 153). Examine the results carefully to see if this is what you intended. Figure 153 illustrates how the assertion interprets the test input given the sample expression shown:
 - For the test input shown in Figure 2, the test results are *found = true* and *count = 2*. This means that there were two inputs that fulfilled the criteria: *Nigel Rees* and *2: Evelyn Waugh*. The assertion was able to locate the author but not the category, title, or price, because they were not specified.
 - If you used the test input "bicycle", no test results will appear because it does not fulfill the expression criteria ".author".

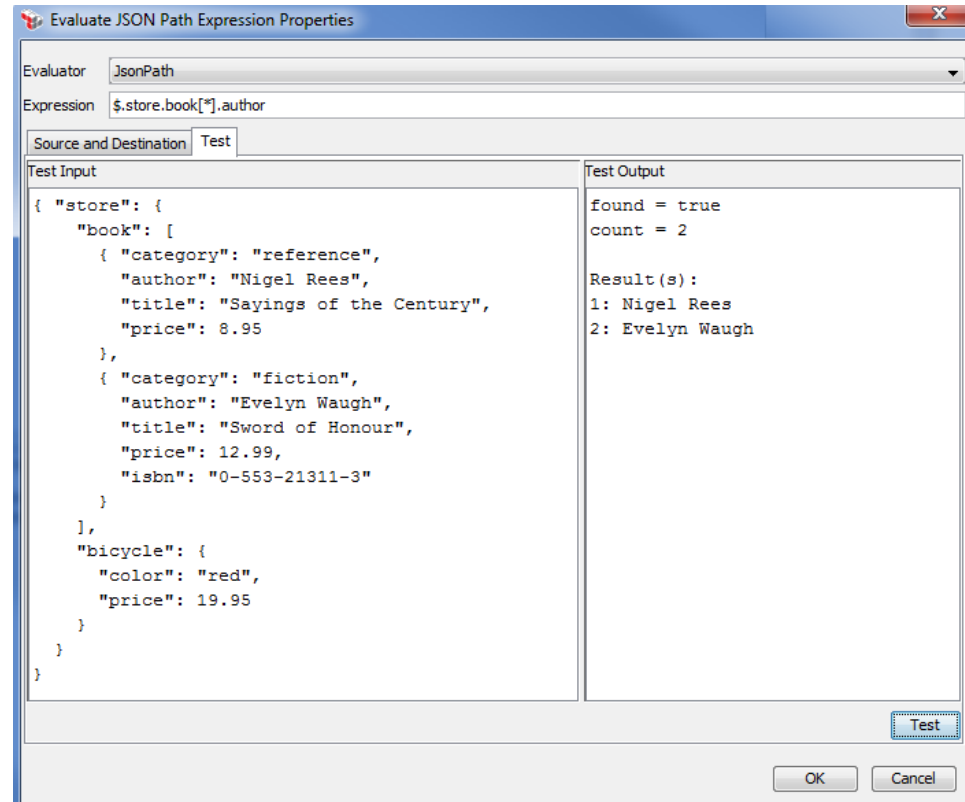


Figure 153: Evaluate JSON Path Expression Properties - [Test] tab

6. Click **[OK]** when done.

Evaluate Regular Expression Assertion

The *Evaluate Regular Expression* assertion is a powerful tool for detecting, filtering, and/or changing service messages. The assertion allows you to define one or more values that, when present in an incoming request message, will yield a specific processing outcome, change the content of the matched message, or detect particular patterns.

The Evaluate Regular Expression assertion has a wide range of uses. For example, it can be configured to enforce a consistent telephone number format in request and response messages. The assertion will then scan messages for telephone numbers—any number that does not conform to the format specified in the assertion will be altered prior to processing the message. The assertion can also be used for message mediation: using its match and replace functionality, the assertion can convert small item changes, such as adding a special tag after a particular keyword is detected.

Pattern matching and replacement are also useful for protecting service applications from various web service and XML application threats.

A policy can contain multiple Evaluate Regular Expression assertions, placed anywhere before or after the [routing](#) assertion.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about regular expressions, consider a web tutorial such as this site: <http://www.regular-expressions.info/>.

Context Variables Created by This Assertion

The Evaluate Regular Expression assertion can optionally populate a multivalued context variable with the values from designated capture groups in the expression.

Example:

- Multivalued context variable name: *phone*
- Regular expression containing three capture groups:
`\((\d{3})\) (\d{3}) - (\d{4})`
- Input string: *(800) 555-1234*

Using the default settings for the assertion, the multivalued variable *phone* will contain the following values:

- **`${phone[0]}`** is set to the entire string matched by the regular expression – "(800) 555-1234"
- **`${phone[1]}`** is set to the first capture group – "800"
- **`${phone[2]}`** is set to the second capture group – "555"
- **`${phone[3]}`** is set to the third capture group – "1234"

Note that the variable *phone* is created even if the **Do not proceed if pattern matches** option was selected.

Note: Capture groups always exist if any parentheses are present in the expression. However, they are saved only when a context variable is specified.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.

- When adding the assertion, the Regular Expression Properties automatically appear; when modifying the assertion, right-click **<target>: Evaluate Regular Expression** in the policy window and select **Regular Expression Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Regular Expression Properties

Display Name: (optional; GUI use only)

Regular Expression:

Save in context variable: ☒ OK

Replacement:

Source and Destination

Source

☐ Request

☐ Response

☒ Other Context Variable:

MIME/Multipart Messages

MIME Part: (For multipart message: 0 - first part, 1 - second part, ...)

Character Encoding: ☒ Default ☐ Override:

Destination

☐ Proceed if pattern matches

☐ Do not proceed if pattern matches

☒ Stop searching after first successful match

☒ Match and Replace (always proceed)

☐ Repeat successful replacements up to times

Context Variable:

☒ OK (optional; for capture groups)


☒ Include matched substring in capture


OK Cancel

Figure 154: Regular Expression Properties - [Source and Destination] tab

3. Configure the display name and regular expressions:

Table 123: Regular Expression Properties - basic settings

Setting	Description
Display Name	<p>Optionally enter a "friendly name" for this regular expression. This name is for display purposes only in the policy window.</p> <p>Tip: A friendly name should briefly describe the purpose of the assertion. This name helps you distinguish between several Evaluate Regular Expression assertions in a policy.</p>
Regular Expression 	<p>Enter the regular expression value to be matched. You may reference context variables within the expression. Note that any context variable will be treated as literals when the syntax is checked during design time, but will be resolved to their actual values during runtime. For details, see "Example: Context variables in the regular expression" below.</p> <p>Note: The Regular Expression field can only contain a single expression. To evaluate multiple expressions, configure multiple Evaluate Regular Expression assertions within a policy.</p>

Setting	Description
Ignore Case	Select this check box to ignore the case of any matching values in the incoming request message. Clear this check box to enforce case matching.
Save in context variable	<p>Optionally reference a context variable that will hold the regular expression pattern in effect at runtime.</p> <p>This is most useful if context variables were used in the regular expression pattern, as the exact pattern will depend on the resolved variables during runtime. If no context variables were used, then this variable will contain the exact string entered in the Regular Expression field.</p>
Replacement (only used with "Match and Replace" option) 	<p>Enter the replacement value or format that will replace the value or format specified in the Regular expression field. You may reference context variables.</p> <p>Tip: Use the [Test] tab to verify that the replacement is working correctly.</p>

Example: Context variables in the regular expression

Suppose you enter the following regular expression:

hi\${there}bob

When you test this expression during design time, the assertion will match all the characters literally: h, i, \$, {, t, h, e, r, e, }, b, o, b.

At runtime, the assertion will match the characters h, i, followed by whatever is currently in the variable *\${there}*, followed by the characters b, o, b.

If at runtime *\${there}* contains regular expression metacharacters such as [,], |, ^, \$, etc., they will be matched as literals (in other words, they lose their metacharacter interpretation). For example, consider this policy fragment:

Set variable \${there} to [a-z]

Request: match regex hi\${there}bob

This fragment will match this request:

We all scream hi[a-z]bobbies again!

But it will not match this request:

We all scream hipbobbies again!

If you modify the fragment to replace the context variable with its actual content, the metacharacters will be interpreted as expected:

Request: match regex hi[a-z]bob

This will result in the second example above being matched, but not the first.

4. In the [Source and Destination] tab, configure the assertion as follows:

Table 124: Regular Expression Properties - Source & Destination settings

Setting	Description
Source	<p>Specify whether to match against the Request, Response, or Other Context Variable that contains the value to analyze. If other variable, specify the variable name in the box. (You do not need to enclose the variable name within the "\${ }" characters.)</p> <p>Tip: The message target can also be set outside of the assertion properties. For more information, see "Selecting a Target Message" on page 153.</p>
Destination	<p>Specify how to proceed based on the results of the pattern matching:</p> <ul style="list-style-type: none"> • Proceed if pattern matches: Causes the assertion to return success if the pattern matches. Whether the message is permitted to proceed ultimately depends on the outcome of the policy. • Do not proceed if pattern matches: Causes the assertion to return failure if the pattern matches. Whether the message is blocked ultimately depends on the outcome of the policy. This option is particularly useful for protecting against specific service threats. • Stop searching after first successful match: This check box is available when either Proceed if pattern matches or Do not proceed if pattern matches is selected. <ul style="list-style-type: none"> • Select this check box to instruct the assertion to stop after a successful match. This setting is the default. • Clear this check box to instruct the assertion to find (and capture, if applicable) all matches in the target string, not stopping after the first match. • Match and Replace (always proceed): If the content matches the regular expression, replace it with the content from the Replacement field. Tip: If multiple replacements are required, use several Evaluate Regular Expression assertions in the policy. <ul style="list-style-type: none"> • Repeat successful replacements up to x times: <ul style="list-style-type: none"> • Select this check box to repeat any "replace all" step that made at least one replacement. This will continue until either there is no more item to replace, or the 'x' iteration limit has been reached. See "Example: 'Repeat successful replacements' option" below for an example of how this option works. • Clear this check box to not repeat any successful replacement step. This setting is the default. • Context Variable: Optionally enter the name of a context variable if you wish to record capture groups. For more information, see "Context Variables Created by Assertion" in

Setting	Description
	<p>the introduction to this assertion. For information on naming rules, see "Context Variable Naming Rules" under <i>Context Variables</i> in the <i>Layer 7 Policy Manager User Manual</i>.</p> <ul style="list-style-type: none"> • Include matched substring in capture: This check box has an effect only when a Context Variable has been specified. It controls whether the matched substring should be included in the capture. For a detailed explanation of this option, see "Example: Including matched substring" below. This option is enabled by default to replicate the functionality of the assertion prior to version 6.2.
MIME/Multipart Messages	<p>Specify how to handle MIME/multipart messages:</p> <ul style="list-style-type: none"> • MIME Part: For multipart messages, specify which part of the message should be matched against the regular expression value, where '0' is the first part, '1' is the second part, etc. The default is '0'. <p>This setting is not used for messages that have only a single part.</p> • Character Encoding: Select Default to use the default character encoding or Override to override how the Gateway decodes the message. For example, if a UTF-8 encoded message arrives with a Content-Type incorrectly declaring its character encoding as ISO8859-1, then enter "UTF-8" to override. For more information, see "Character Encoding" on page 412.

Example: "Repeat successful replacements"

The following example illustrates the functionality of the **Repeat successful replacements** check box: adding commas to a large number.

Say you have a large integer and you wish to add commas for improved readability. The length of this integer is unknown ahead of time. You can accomplish this using the following settings in the Evaluate Regular Expression assertion:

Regular expression: `^(?-?\d+)(\d{3})`

Replacement: `$1,$2`

☒ *Repeat successful replacements up to* **9999** *times*

Before processing, the target message contains this integer:

92349854732933493424982745249587

After processing, it will contain: 92,349,854,732,933,493,424,982,745,249,587

How this works behind the scene:

The assertion examines the integer and will attempt to match all the leading digits that lack commas, followed by three more digits that lack commas. It then replaces this with the same string, but with a comma before the last three digits. This is repeated until the assertion cannot find four consecutive digits. Here is a sample of the assertion in progress:

Repeat up to 0 times: 92349854732933493424982745249,587
 Repeat up to 1 times: 92349854732933493424982745,249,587
 Repeat up to 2 times: 92349854732933493424982,745,249,587
 Repeat up to 7 times: 92349854,732,933,493,424,982,745,249,587
 Repeat up to 9999 times: 92,349,854,732,933,493,424,982,745,249,587

If the **Repeat successful replacements** check box is not selected, the assertion will stop after one pass and the resulting output would be "92349854732933493424982745249,587".

Example: "Including matched substring"

As described under ["Context Variables Created by This Assertion"](#), you can optionally save the "capture groups" that are automatically created by this assertion by entering a variable name in the assertion properties. You indicate the part of the pattern to be captured by enclosing them within parentheses. When the assertion is run, the matching part of the pattern is then captured and saved to the context variable when matched successfully.

For example, suppose you have the following data:

name="John Smith", phone=604-555-1234
name="Sue Smith", phone=604-555-5678

You wish to extract the phone number (which for this example we will assume that it is always in the xxx-xxx-xxxx format). This can be accomplished with the following regular expression:

phone=(\d\d\d-\d\d\d-\d\d\d\d)

Scenario 1: Run the assertion with the following settings:

Stop searching after first successful match = **enabled**
Include matched substring in capture = **enabled**
Context variable = **p**

This will be the result:

\${p[0]} = "phone=604-555-1234"
\${p[1]} = "604-555-1234"

A few points to note about these results:

- The variable "p" that is created is a multivalued context variable.
- The first value in "p" ($\{p[0]\}$) contains the entire substring matched by the regular expression.
- The second value in "p" ($\{p[1]\}$) contains the first capture group, which is as indicated in the regular expression.
- The phone number for Sue Smith is not captured, because you've instructed the assertion to stop search after a successful match is made.
- Prior to version 6.2, the Evaluate Regular Expression assertion always stopped after the first match.

Scenario 2: Run the assertion with the following settings:

Stop searching after first successful match = **disabled**

Include matched substring in capture = **enabled**

Context variable = **p**

This will be the result:

$\{p[0]\}$ = "phone=604-555-1234" (entire substring matched by the regex, for the 1st successful match)

$\{p[1]\}$ = "604-555-1234" (1st group, for the 1st successful match)

$\{p[2]\}$ = "phone=604-555-4332" (entire substring matched by the regex, for the 2nd successful match)

$\{p[3]\}$ = "604-555-4332" (1st capture group, for the 2nd successful match)

Note that when the "Stop searching..." option is disabled, Sue Smith's phone number is captured.

Scenario 3: Run the assertion with the following settings:

Stop searching after first successful match = **disabled**

Include matched substring in capture = **disabled**

Context variable = **p**

This will be the result:

$\{p[0]\}$ = "604-555-1234" (1st capture group, for the 1st successful match)

$\{p[1]\}$ = "604-555-4332" (1st capture group, for the 2nd successful match)

Notice that when you disable "Include matched substring", the matched substrings are no longer saved.

Scenario 4: Use the same settings as Scenario 3, but introduce a new capture group by enclosing the *entire* regular expression within parentheses (remember, capture groups are indicated by parentheses):

(phone=(\d\d\d-\d\d\d-\d\d\d\d))

This will be the result:

`${p[0]}` = "phone=604-555-1234" (1st capture group, for the 1st successful match)

`${p[1]}` = "604-555-1234" (2nd capture group, for the 1st successful match)

`${p[2]}` = "phone=604-555-4332" (1st capture group, for the 2nd successful match)

`${p[3]}` = "604-555-4332" (2nd capture group, for the 2nd successful match)

Notice that this is identical to the output from Scenario 2. Thus, the "Include matched substring" option is the same as disabling the option and enclosing the entire expression within parentheses. **Note:** Prior to version 6.2, matched substrings were always included and could not be disabled.

5. Select the [Test] tab to test your regular expression and to determine whether the replacement string was entered correctly.
 - In the **Test Input** box, type some sample text that includes the value or format from the **Regular Expression** field. As you type, the assertion attempts to match your input against the Regular Expression that was entered.
 - The **Test Result** box shows the results of the match (see Figure 155). Examine the results carefully to see if this is what you intended. Figure 155 illustrates how the assertion interprets the test input given the sample regular expression and replacement string shown:
 - For test input "888-555-1234", the test result is "<phone country="" area="888" num="555-1234"/>". This means the assertion was able to locate the area code and phone number, but not the country code because it was not specified.
 - For test input "1-888-687-2234", the test result was able to match all three groups successfully.
 - For test input "some test", no replacement was made because this is not a phone number.
 - For test input "879-1234", no replacement was made because the phone number is missing the area code, which is a required element.
 - For test input, "604-681-9387", the area code and phone number was matched.

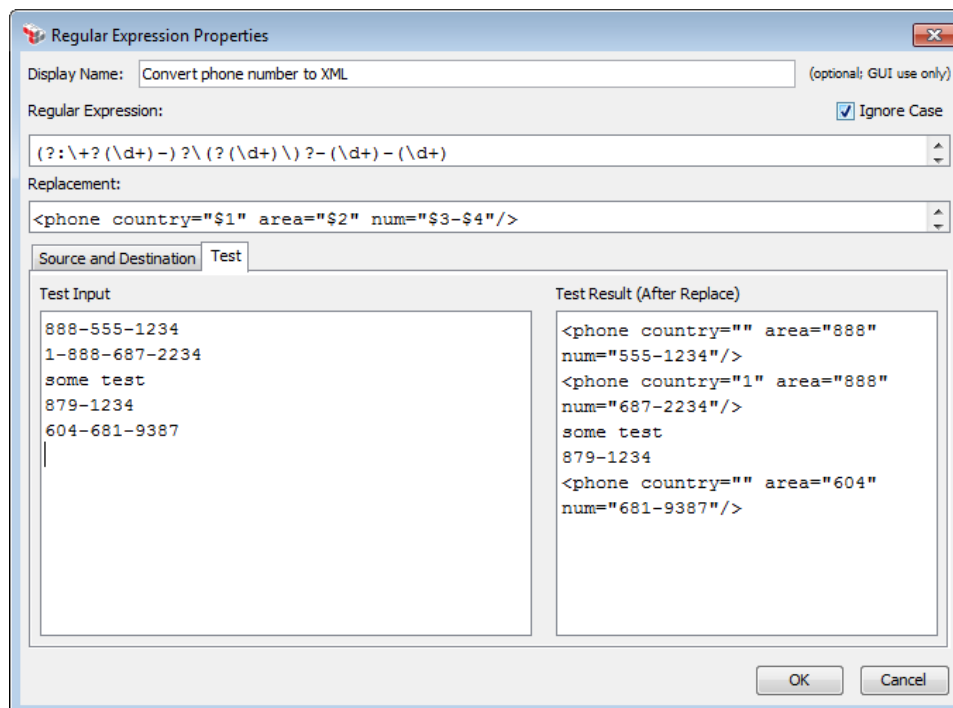


Figure 155: Evaluate Regular Expression assertion - [Test] tab

6. Click **[OK]** when done.

Evaluate Request XPath Assertion

The *Evaluate Request XPath* assertion is used to configure a specific XPath query pattern for incoming XML request messages. This assertion can be used, for example, to break a policy into different paths for different types of operations and/or to provide preferential routing to special customers or high dollar value transactions.

Note: Ensure that you are familiar with XPath patterns and specifications before you configure the Evaluate Request XPath assertion.

Context Variables Created by This Assertion

The XPath query targets specific parts of the message, evaluating its document structure and/or XML data. When the assertion executes, it runs the XPath pattern against the request, setting six context variables according to the processing result of the query. The following table describes the default context variables set by the Evaluate Request XPath assertion.

Tip: When the Evaluate Request XPath assertion is used in a [policy fragment](#) and you need the context variables in Table 125 available to the parent policy, be sure to include the [Export Variables from Fragment](#) assertion within the policy fragment to ensure that the variables are available to the parent policy.

Table 125: Context variables created by Evaluate Request XPath assertion

Variable	Description
requestXPath.result	<p>The content of this variable depends on the match found:</p> <ul style="list-style-type: none"> If the XPath expression matches an element, this variable contains the contents of the first element that matched the XPath expression. If the XPath expression locates any other match, this variable contains the first match. If the XPath expression does not locate a match, the assertion fails and this variable is not set. <p><i>Example:</i></p> <p>Consider the following sample XML document:</p> <pre><test> <data>hello</data> <data>world</data> </test></pre> <p>The <i>requestXPath.result</i> variable will be set as follows:</p> <ul style="list-style-type: none"> The XPath expression <code>/test/data</code> matches both elements <code><data>hello</data></code> and <code><data>world</data></code>. In this case, the variable will contain <code>"hello"</code>, which is the contents of the first element that was found. The XPath expression <code>/test/data[2]/text()</code> does not match any element but does match <code>"world"</code>. In this case, the variable will contain <code>"world"</code>, which is the first match found.
requestXPath.results	<p>Similar to <i>requestXPath.result</i> except that it contains all values matched rather than just the first one. Using the above example, the XPath expression <code>/test/data</code> will yield <code>"hello"</code> and <code>"world"</code> in the variable.</p>
requestXPath.element	<p>Contains the resulting text of the query. Similar to the <i>requestXPath.result</i> variable, except this includes the entire element (including start and end tags).</p> <p>This variable contains values of type String.</p>
requestXPath.elements	<p>Similar to <i>requestXPath.element</i> except that it contains all values matched rather than just the first one.</p> <p>This variable contains values of type Element.</p> <p>Note: You cannot use <i>requestXPath.element</i> where</p>

Variable	Description
	<i>requestXPath.elements</i> is required, as assertions that expect an 'Element' value will not work with a "String" value.
requestXPath.count	Contains the number of nodes found, which will be ≥ 1 if the expression matched.
requestXPath.found	Either "true" or "false", depending on whether the XPath expression matches the request.

The values stored in the context variables can be used in subsequent [Evaluate Regular Expression](#) or [Compare Expression](#) assertions using the special `${assertionPrefix.variableName}` syntax. If the [Audit Messages in Policy](#) assertion is encountered during message processing, then the XPath query results can be viewed in the Gateway Audit Events window.

If the variables from several Evaluate Request XPath assertions need to be available simultaneously (for example, an upcoming [Compare Expression](#) assertion will be used to compare the results), then each assertion must use a different prefix. For example, you define the new variable prefix *newPrefix* in the Evaluate Request XPath assertion. Now instead of the default names shown in Table 125, this assertion will create variables named *newPrefix.result*, *newPrefix.count*, *newPrefix.found*, and *newPrefix.element*.

Note: Since a single variable [namespace](#) is shared during the entire processing of a message, the prefixes used in both the Evaluate Request XPath and [Evaluate Response XPath](#) assertions must be unique. If the same variable prefix is used in both, then one assertion will overwrite the other's variables.

If the XPath is a Boolean expression, the assertion succeeds only if the Boolean expression evaluates to "true". If the XPath selects nodes, the assertion succeeds only if the list of matching nodes is non-empty. In any other case, including an XPath error, the assertion fails.

The Evaluate Request XPath assertion supports the XPath 1.0 standard.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Follow the appropriate section below to complete the properties dialog.

Web Service Policy

When using the Evaluate Request XPath assertion in a web service policy:

- a. Specify the XPath and select the target element to be evaluated from the code box. For detailed instructions on using the interface to build your XPath, see "Selecting an XPath" on page 154.
- b. Select the appropriate XPATH version from the drop down.
- c. For an explanation of the validation messages displayed, see Context Variable Validation in the *Layer 7 Policy Manager User Manual*.
- d. Make sure that the variable prefix entered here is different from those used in other Evaluate Request XPath and [Evaluate Response XPath](#) assertions. See the description at the beginning of this topic for more information about variable prefixes.
- e. Click **[OK]**.

XML Application Policy

When using the Evaluate Request XPath assertion in an XML application policy:

- a. Enter an XPath expression corresponding to the target request element in the field.
- b. Optionally add namespaces to the [namespace map](#).
- c. Click **[OK]**.

Evaluate Response XPath Assertion

The Evaluate Response XPath assertion is used to configure a specific XPath query pattern for outgoing XML response messages. This assertion can be used, for example, to break a policy into different paths for different types of operations and/or to provide preferential routing to special customers or high dollar value transactions.

Note: Ensure that you are familiar with XPath patterns and specifications before you configure the Evaluate Response XPath assertion. The Evaluate Response XPath assertion must be placed anywhere after the [routing assertion](#) in a policy.

Context Variables Created by This Assertion

The XPath query targets specific parts of the message, evaluating its document structure and/or XML data. When the assertion executes, it runs the XPath pattern against the response, setting six context variables according to the processing result of the query. The following table describes the default context variables sset by the Evaluate Response XPath assertion.

Tip: When the Evaluate Response XPath assertion is used in a [policy fragment](#) *and* you need the context variables in Table 126 available to the parent policy, be sure to include the [Export Variables from Fragment](#) assertion within the policy fragment to ensure that the variables are available to the parent policy.

Table 126: Context variables created by Evaluate Response XPath assertion

Variable	Description
responseXPath.result	<p>The content of this variable depends on the match found:</p> <ul style="list-style-type: none"> • If the XPath expression matches an element, this variable contains the contents of the first element that matched the XPath expression. • If the XPath expression locates any other match, this variable contains the first match. • if the XPath expression does not locate a match, the assertion fails and this variable is not set. <p><i>Example:</i></p> <p>Consider the following sample XML document:</p> <pre><test> <data>hello</data> <data>world</data> </test></pre> <p>The <i>responseXPath.result</i> variable will be set as follows:</p> <ul style="list-style-type: none"> • The XPath expression <code>/test/data</code> matches both elements <code><data>hello</data></code> and <code><data>world</data></code>. In this case, the variable will contain "hello", which is the contents of the first element that was found. • The XPath expression <code>/test/data[2]/text()</code> does not match any element but does match "world". In this case, the variable will contain "world", which is the first match found.
responseXPath.results	<p>Similar to <i>responseXPath.result</i> except that it contains all values matched rather than just the first one. Using the above example, the XPath expression <code>/test/data</code> will yield "hello" and "world" in the variable.</p>
responseXPath.element	<p>Contains the resulting text of the query. Similar to the <i>responseXPath.result</i> variable, except this includes the entire</p>

Variable	Description
	<p>element (including start and end tags).</p> <p>This variable contains values of type String.</p>
responseXPath.elements	<p>Similar to <i>responseXPath.element</i> except that it contains all values matched rather than just the first one.</p> <p>This variable contains values of type Element.</p> <p>Note: You cannot use <i>requestXPath.element</i> where <i>requestXPath.elements</i> is required, as assertions that expect an 'Element' value will not work with a 'String' value.</p>
responseXPath.count	<p>Contains the number of nodes found, which will be ≥ 1 if the expression matched.</p>
responseXPath.found	<p>Either "true" or "false", depending on whether the XPath expression matches the response.</p>

The values stored in context variables can be used in subsequent [Evaluate Regular Expression](#) or [Compare Expression](#) assertions using the special `${assertionPrefix.variableName}` syntax. If the [Audit Messages in Policy](#) assertion is encountered during message processing, then XPath query results can be viewed in the Gateway Audit Events window.

If the variables from several Evaluate Response XPath assertions need to be available simultaneously (for example, an upcoming [Compare Expression](#) assertion will be used to compare the results), then each assertion must use a different prefix. For example: You define the new variable prefix *newPrefix* in the Evaluate Response XPath assertion. Now instead of the default names shown in Table 126, this assertion will create variables named *newPrefix.result*, *newPrefix.count*, *newPrefix.found*, and *newPrefix.element*.

Note: Since a single variable [namespace](#) is shared during the entire processing of a message, the prefixes used in both the Evaluate Response XPath and [Evaluate Request XPath](#) assertions must be unique. If the same variable prefix is used in both, then one assertion will overwrite the other's variables.

If the XPath is a Boolean expression, the assertion succeeds only if the Boolean expression evaluates to "true". If the XPath selects nodes, the assertion succeeds only if the list of matching nodes is non-empty. In any other case, including an XPath error, the assertion fails.

The Evaluate Response XPath assertion supports the XPath 1.0 standard.

Using the Assertion


1. Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
- To change the configuration of an existing assertion, proceed to step 2 below.

Follow the appropriate section below to complete the properties dialog.

Web Service Policy

When using the Evaluate Response XPath assertion in a web service policy:

-  Select the **XML message source** from the drop-down list. The default is to use the **Default Response** message source, but you can also retrieve the message from an eligible context variable. An eligible context variable is one that meets either of the following conditions:
 - The context variable was created using the [Set Context Variable](#) assertion and is of type Message.
 - The context variable was created from the [Response HTTP Rules] tab of the "Route via HTTP(S) Assertion" on page 529.
- Specify the XPath and select the target element to be evaluated from the code box. For detailed instructions on using the interface to build your XPath, see "Selecting an XPath" on page 154.
- Select the appropriate XPATH version from the drop down.
- Enter a prefix that will be added to the [context variables created by this assertion](#). This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.

For an explanation of the validation messages displayed, see Context Variable Validation in the *Layer 7 Policy Manager User Manual*.

- Make sure that the variable prefix entered here is different from those used in other Evaluate Response XPath and [Evaluate Request XPath](#) assertions. See the description at the beginning of this topic for more information about variable prefixes.
- Click **[OK]**.

XML Application Policy

When using the Evaluate Response XPath assertion in an XML application policy:

1. Optionally set the **XML message source**, if necessary.
2. Enter an XPath expression corresponding to the target response element in the field.
3. Optionally add namespaces to the [namespace map](#).
4. Click **[OK]**.

Evaluate WSDL Operation Assertion

The *Evaluate WSDL Operation* assertion determines which operation is being invoked based on the information in the service's WSDL and matches it against a pre-selected one. This assertion succeeds if the operation matches the one set in the assertion, otherwise it fails. For example, a user can be granted rights to view a product list but that same user cannot view the product details nor place an order.

Example:

Consider the following sample policy:

```

    "At least one assertion must evaluate to true"
    "All assertions must evaluate to true"
        "At least one assertion must evaluate to true"
            Evaluate WSDL Operation 'listProducts'
            Evaluate WSDL Operation 'getProductDetails'
        Authenticate User: 'Bob'
    "All assertions must evaluate to true"
        Evaluate WSDL Operation 'listProducts'
        Authenticate User: 'Sue'
  
```

The results will be as follows:

- If Bob attempts to either list products or get product details, the assertion will succeed because these are included in his of legal operations.
- If Sue attempts to get product details, the assertion will fail because her operation only allows 'list products'.

For more information about organizing policies, see "Policy Organization" on page 2.

Using the Assertion

1. Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Evaluate WSDL Operation** in the policy window and select **WSDL Operation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

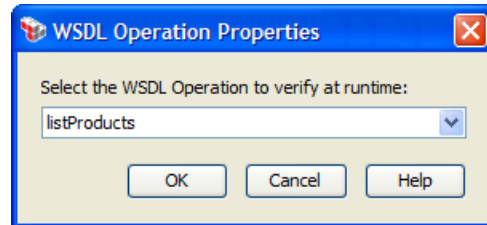


Figure 156: WSDL Operation Properties

3. From the drop-down list, select the WSDL operation to be verified at run time. The Gateway will compare the WSDL's requested operation against this selection to determine whether the assertion passes or fails.
4. Click [**OK**] when done.

Process SAML Attribute Query Request Assertion

The *Process SAML Attribute Query Request* assertion validates AttributeQuery requests based on user configuration. It also makes values and elements from an AttributeQuery available as context variables.

This assertion only supports SAML 2.0.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Context Variables Created by This Assertion

The Process SAML Attribute Query Request assertion sets the following context variables.

Note: The default *<prefix>* is "attrQuery" and can be changed in the assertion properties (Figure 157).

Note: The 'subject' context variables in Table 127 below (except for *subject.format*) will not be set if the NameID was encrypted and decryption was not configured.

Table 127: Context variables created by Process SAML Attribute Query Request assertion

Context variable	Type	Notes
<code><prefix>.attributes</code>	Element (multivalued)	All Attribute elements contained in the AttributeQuery.
<code><prefix>.subject</code>	String	Value of the Subject's NameID.
<code><prefix>.subject.nameQualifier</code>	String	Subject's NameID's NameQualifier attribute value, if provided.
<code><prefix>.subject.spNameQualifier</code>	String	Subject's NameID's SPNameQualifier attribute value, if provided.
<code><prefix>.subject.format</code>	String	Subject's NameID's Format attribute value, if provided. Never empty; if not supplied, value will be <i>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</i> .
<code><prefix>.subject.spProvidedId</code>	String	Subject's NameID's SPProvidedID attribute value, if present.
<code><prefix>.id</code>	String	AttributeQuery's ID attribute, if present.
<code><prefix>.version</code>	String	AttributeQuery's Version attribute, if present.
<code><prefix>.issueInstant</code>	String	AttributeQuery's IssueInstant attribute, if present.
<code><prefix>.destination</code>	String	AttributeQuery's Destination attribute, if present.
<code><prefix>.consent</code>	String	AttributeQuery's Consent attribute. If not supplied, the value will be <i>urn:oasis:names:tc:SAML:2.0:consent:unspecified</i> .
<code><prefix>.issuer</code>	String	AttributeQuery's Issuer element's value, if present.
<code><prefix>.issuer.nameQualifier</code>	String	Issuer's NameQualifier attribute value, if present.
<code><prefix>.issuer.spNameQualifier</code>	String	Issuer's SPNameQualifier attribute value, if present.
<code><prefix>.issuer.format</code>	String	Issuer's Format attribute value, if present.
<code><prefix>.issuer.spProvidedId</code>	String	Issuer's SPProvidedID attribute value, if present.

The following variables may also be set:

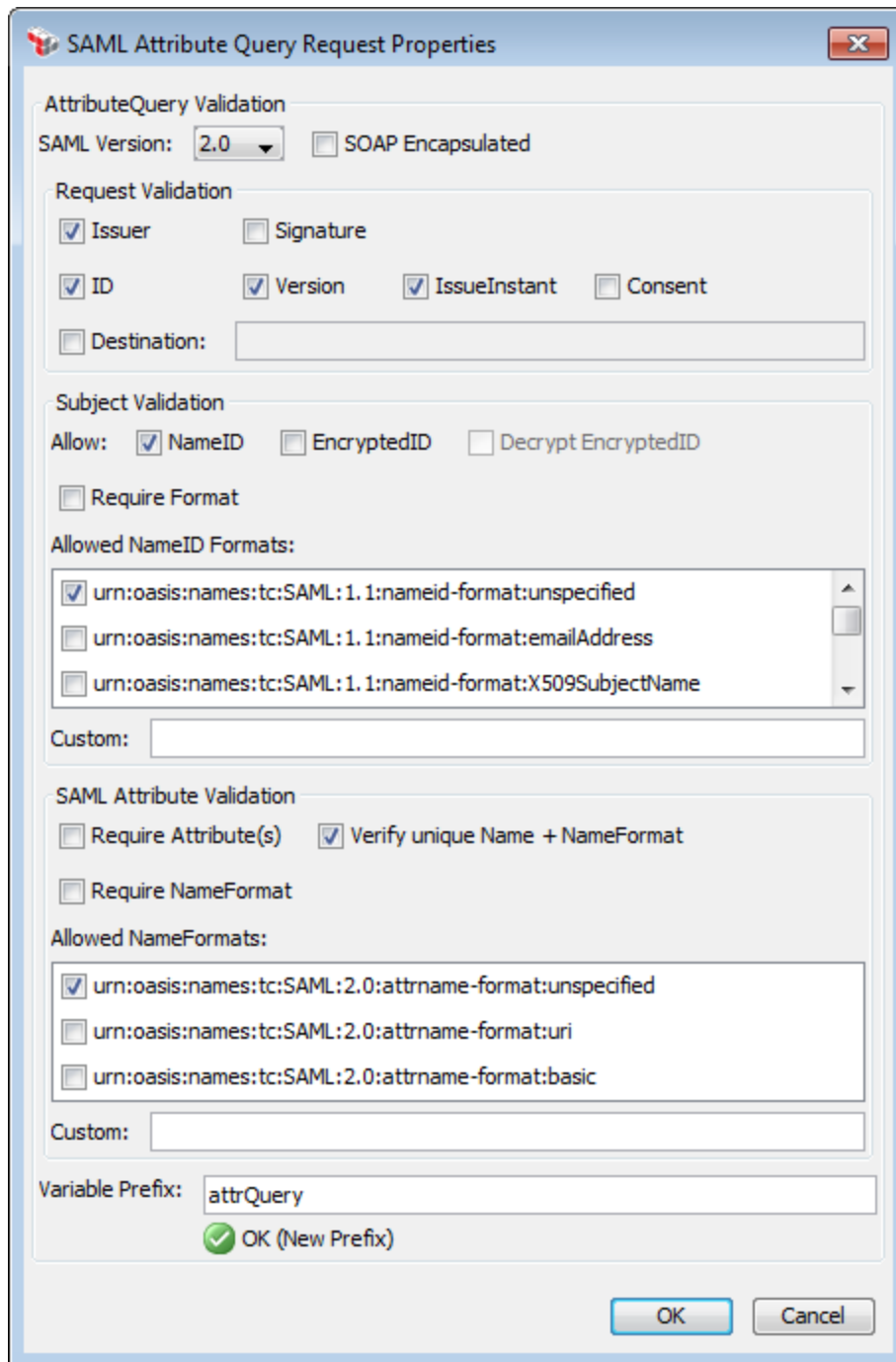
- If decryption is configured and was performed (Decrypt EncryptedID check box in Figure 157), then all the context variables from the [\(Non-SOAP\) Decrypt XML Element](#) assertion will also be set. These variables include:

`<prefix>.elementsDecrypted`
`<prefix>.encryptionMethodUri`
`<prefix>.recipientCertificates`

The prefix used for those variables is the prefix specified in Figure 157 For more information, see "(Non-SOAP) Decrypt XML Element Assertion" on page 374.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Process SAML Attribute Query Request** in the policy window and select **SAML Attribute Query Request Properties** or double-click the assertion in the policy window. The assertion properties are displayed.



SAML Attribute Query Request Properties

AttributeQuery Validation
SAML Version: ☐ SOAP Encapsulated

Request Validation
☒ Issuer ☐ Signature
☒ ID ☒ Version ☒ IssueInstant ☐ Consent
☐ Destination:

Subject Validation
Allow: ☒ NameID ☐ EncryptedID ☐ Decrypt EncryptedID
☐ Require Format
Allowed NameID Formats:
☒ urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified
☐ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
☐ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
Custom:

SAML Attribute Validation
☐ Require Attribute(s) ☒ Verify unique Name + NameFormat
☐ Require NameFormat
Allowed NameFormats:
☒ urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified
☐ urn:oasis:names:tc:SAML:2.0:attrname-format:uri
☐ urn:oasis:names:tc:SAML:2.0:attrname-format:basic
Custom:


Variable Prefix:
☒ OK (New Prefix)



OK Cancel

Figure 157: SAML Attribute Query Request Properties

3. Configure the dialog as follows:

Table 128: SAML Attribute Query Request settings

Setting	Description
AttributeQuery Validation	<ul style="list-style-type: none"> • SAML Version: Only SAML 2.0 is supported. • SOAP Encapsulated: Select this check box if the AttributeQuery is encapsulated within a SOAP envelope.
Request Validation	<p>Select the appropriate check boxes to indicate which attribute or element must be present in an AttributeQuery request:</p> <p>Issuer Signature* ID Version IssueInstant Consent Destination** </p> <p>*This assertion does not validate or verify the signature. To validate the signature, use the (Non-SOAP) Verify XML Element assertion. To remove the signature, use the Add or Remove XML Element(s) assertion.</p> <p>**Select the Destination check box to indicate that a destination attribute is required. If the destination attribute must have an allowed value, enter all allowed values in the adjacent text box. Enter in as many values as needed separated by a space. You may specify URIs or context variables of type String (variables that resolve to an empty string or non-string are ignored and will not cause assertion failure, but a 'Warning' audit is logged). Context variables may contain space-separated URI strings.</p> <p>Note: If an attribute/element has been configured but is missing, the assertion will fail.</p>
Subject Validation	<ul style="list-style-type: none"> • Allow: Select the supported Subject identifiers: NameID EncryptedID <p>If [EncryptedID] is permitted, select the Decrypt EncryptedID check box to decrypt the EncryptedID and update the message with the result of the decryption.</p> <p>Tip: The "Require Format" and "Allowed NameID Format" validation are applied only when either a NameID was included in the AttributeQuery or if an EncryptedID was received and decrypted. If decryption was not selected, then this validation cannot be performed. Additionally, context variables related to the NameID will not be set.</p> • Require format: Select this check box to require the Format attribute to be present on the NameID, otherwise the assertion will fail. Clear this check box if the Format attribute is not

Setting	Description
	<p>required. If no format attribute is supplied, it will have the following default value: urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified.</p> <ul style="list-style-type: none"> • Allowed NameID formats: Select the supported NameID formats from the list. By default, urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified is selected as this is the default value of this attribute when no value is supplied. • Custom: If the NameID format you need is not listed, enter a set of custom Format URI values here.  <p>Enter in as many values as needed separated by a space. You may specify URIs or context variables of type String (variables that resolve to an empty string are ignored and will not cause assertion failure, but a "Warning" audit is logged). Context variables may be single or multivalued. Single-valued variables may contain space-separated URI strings.</p>
SAML Attribute Validation	<p>This section configures the rules for the saml:Attributes contained in the Attribute Query.</p> <ul style="list-style-type: none"> • Require Attributes: Select this check box to fail the assertion if an empty AttributeQuery is received. Clear the check box if attributes are not required. • Verify unique Name + NameFormat: Select this check box to fail the assertion if there are any logical duplicate attributes. Note that the AttributeValue (if any) is not considered in this check. • Require NameFormat: Select this check box to fail the assertion if the NameFormat attribute is not present. Clear this check box if the NameFormat attribute is not required. • Allowed NameFormats: Select the supported NameFormats from the list. By default, urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified is selected as this is the default value of this attribute when no value is supplied. • Custom: If the NameFormat you need is not listed, enter a set of custom NameFormat URI values here.  <p>Enter in as many values as needed separated by a space. You may specify URIs or context variables of type String (variables that resolve to an empty string are ignored and will not cause assertion failure, but a "Warning" audit is logged). Context variables may be single or multivalued. Single-valued variables may contain space-separated URI strings.</p>
Variable Prefix	Enter a prefix that will be added to the context variables created by this

Setting	Description
	<p>assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>The default variable prefix is attrQuery.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

4. Click **[OK]** when done.

Process SAML Authentication Request Assertion

The *Process SAML Authentication Request* assertion helps to simplify policies used to create a single sign-on service. This assertion can perform the following:

- (Optional) Extract the SAML Request from a form or URL parameter and then decode it.
- Validate that the incoming Authentication Request is valid, according to the SAML profile specifications.

Validation Details

The assertion will validate the following and will fail if these rules are not met:

- `<Issuer>` is present; if `<Format>` is supplied, it must be `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`
- No SubjectConfirmation elements should be present
- Extract key information from the Authentication Request and place them into context variables.

This assertion only supports SAML 2.0.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Context Variables Created by This Assertion

The Process Authentication Request assertion sets the following context variables. **Note:** The default `<prefix>` is "authnRequest" and can be changed in the assertion properties (Figure 158).

Table 129: Context variables created by Process Authentication Request assertion

Context variable	Description
<code><prefix>.subject</code>	Returns the Subject of the AuthnRequest.
<code><prefix>.subject.nameQualifier</code>	Returns the domain to qualify the Subject name.
<code><prefix>.subject.spNameQualifier</code>	Returns the name of a Subject SP, which is used to qualify a name.
<code><prefix>.subject.format</code>	Returns the URI of the Subject format.
<code><prefix>.subject.spProvidedId</code>	Returns the identifier of the Subject SP.
<code><prefix>.x509CertBase64</code>	Returns the Base64-encoded X.509 Certificate, if present in the AuthnRequest.
<code><prefix>.x509Cert</code>	<p>Returns the X.509 Certificate, if present in the AuthnRequest, and if it is convertible into an X.509 Certificate.</p> <p>This variable can be input into the Retrieve Credentials from Context Variable assertion.</p>
<code><prefix>.acsUrl</code>	Returns the URL of the Assertion Consumer Service.
<code><prefix>.Id</code>	Returns the ID of the AuthnRequest.
<code><prefix>.version</code>	Returns the version of the request.
<code><prefix>.issueInstant</code>	Returns the time the request was issued.
<code><prefix>.destination</code>	Returns the destination to which this AuthnRequest was sent.
<code><prefix>.consent</code>	<p>Returns the consent of the AuthnRequest. If one is not available, the following value will be used:</p> <p><i>urn:oasis:names:tc:SAML:2.0:consent:unspecified</i></p>
<code><prefix>.issuer</code>	Returns the entity which issued the AuthnRequest.
<code><prefix>.issuer.nameQualifier</code>	Returns the domain used to qualify the Issuer name.
<code><prefix>.issuer.spNameQualifier</code>	Returns the name of an IssuerSP, which is used to qualify a name.
<code><prefix>.issuer.format</code>	Returns the URI of the Issuer format.
<code><prefix>.issuer.spProvidedId</code>	Returns the identifier of the Issuer SP.
<code><prefix>.request</code>	<p>Returns the contents of the AuthnRequest. This is only set for HTTP bindings to allow XPath of extensions or other values.</p> <p>Tip: To access the main part of this context variable as text, you must</p>

Context variable	Description
	append the ".mainpart" suffix; for example: \${authnRequest.request.mainpart}. For more information about the ".mainpart" suffix, see Transport Layer Variables in the <i>Layer 7 Policy Manager User Manual</i> .

Notes:

The variables `<prefix>.x509CertBase64` and `<prefix>.x509Cert` may contain values if the `<AuthnRequest>` is signed. If this is the case then:

- `<prefix>.x509CertBase64` will contain the Base64 encoded certificate from the request (if any)
- `<prefix>.x509Cert` will contain the X.509 Certificate used to validate the request signature

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Process SAML Authentication Request** in the policy window and select **SAML Authentication Request Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

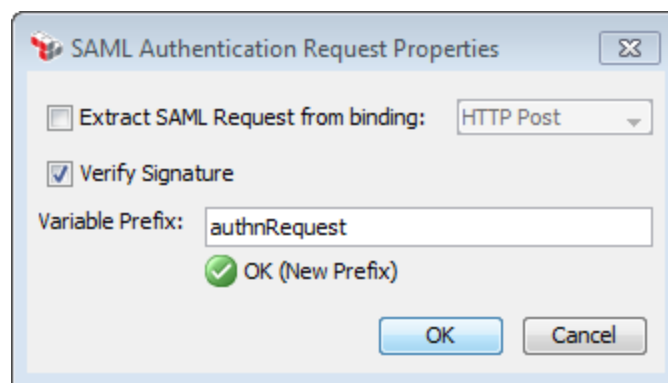


Figure 158: SAML Authentication Request Properties

3. Configure the dialog as follows:

Table 130: SAML Authentication Request settings

Setting	Description
Extract SAML Request from binding	<p>Select this check box to have the assertion extract the SAML Request from the incoming HTTP URL or Form parameters, based on the chosen binding (HTTP Post or HTTP Redirect).</p> <p>Clear this check box to use the SAML Request from the body of the target message selected for this assertion. For more information, see "Selecting a Target Message" on page 153.</p>
Verify Signature	<p>Select this check box to have the assertion validate any signature that is present. Signature validation may use an enclosed X.509 Certificate and may attempt to look up the certificate in the Gateway's trust store.</p> <p>This check box is unavailable if HTTP Redirect is selected for Extract SAML Request from binding.</p>
Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>The default variable prefix is authnRequest.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

4. Click **[OK]** when done.

Replace Tag Content Assertion

The *Replace Tag Content* assertion provides search-and-replace functionality for content within tags, in HTML or XML files (even where the HTML/XML is not properly formed).

This assertion is especially useful in reverse web proxy scenarios, where the Gateway needs to manipulate HTML content in a route response before it is returned to the client.

Tip: The "Evaluate Regular Expression Assertion" on page 449 can also be used to search and replace content in the response, but it is difficult to create regular expressions that search and modify only select areas of the HTML code.

Example use case: Some HTML tags (anchor, script, etc.) may contain content or attributes that references the web application host which is unknown to the client. Use the Replace Tag Content assertion to change the references to the Gateway host, which is serving as a reverse proxy to the web application.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the Replace Tag Content Properties automatically appear; when modifying the assertion, right-click **<target>: Replace...** in the policy window and select **Replace Tag Content Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

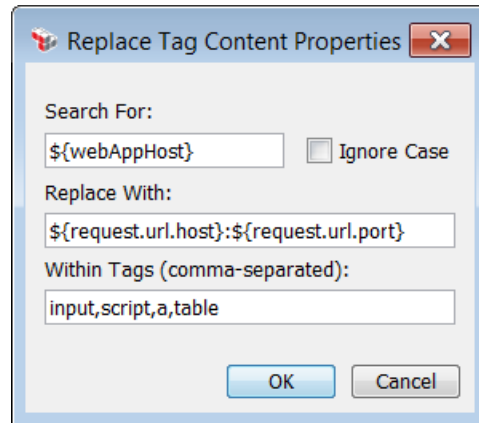





Figure 159: Replace Tag Content Properties

3. Configure the properties as follows:

Table 131: Replace Tag Content settings

Setting	Description
Search For 	Enter the string to search for. You may reference context variables. By default, the search is case sensitive. To make it case insensitive, select the Ignore Case check box.
Replace With 	Enter the replacement text. You may reference context variables.
Within Tags 	Specify the tag(s) to search (case insensitive). Separate multiple tags with commas. You may reference context variables.

4. Click **[OK]** when done.

Require WS-Addressing Assertion

The *Require WS-Addressing* assertion lets you specify which versions of WS-Addressing should be present in the message and whether the addressing headers must be signed. The Gateway uses the WS-Addressing version to select the namespace.

This assertion can use namespaces from the following versions:

WS-Addressing 1.0
WS-Addressing 08/2004

In addition, you can enter any other configurable namespace URI in the assertion properties.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about selecting the target identity for this assertion, see "Selecting a Target Identity" on page 152.

Sample WS-Addressing Message

The following is a sample message with WS-Addressing headers:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:MessageID>http://example.com/6B29FC40-CA47-1067-B31D-
      00DD010662DA</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://example.com/business/client1</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>http://example.com/fabrikam/Purchasing</wsa:To>
    <wsa:Action>http://example.com/fabrikam/SubmitPO</wsa:Action>
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

Context Variables Created by This Assertion

The *Require WS-Addressing* assertion sets the following context variables based on the WS-Addressing headers. **Note:** The *<prefix>* is set in the assertion properties (Figure 160) and is optional. There is no default.

Table 132: Context variables created by *Require WS-Addressing* assertion

Variable	Description
$\$(\text{<prefix>}.to)$	Contains the value from the "to" addressing header.

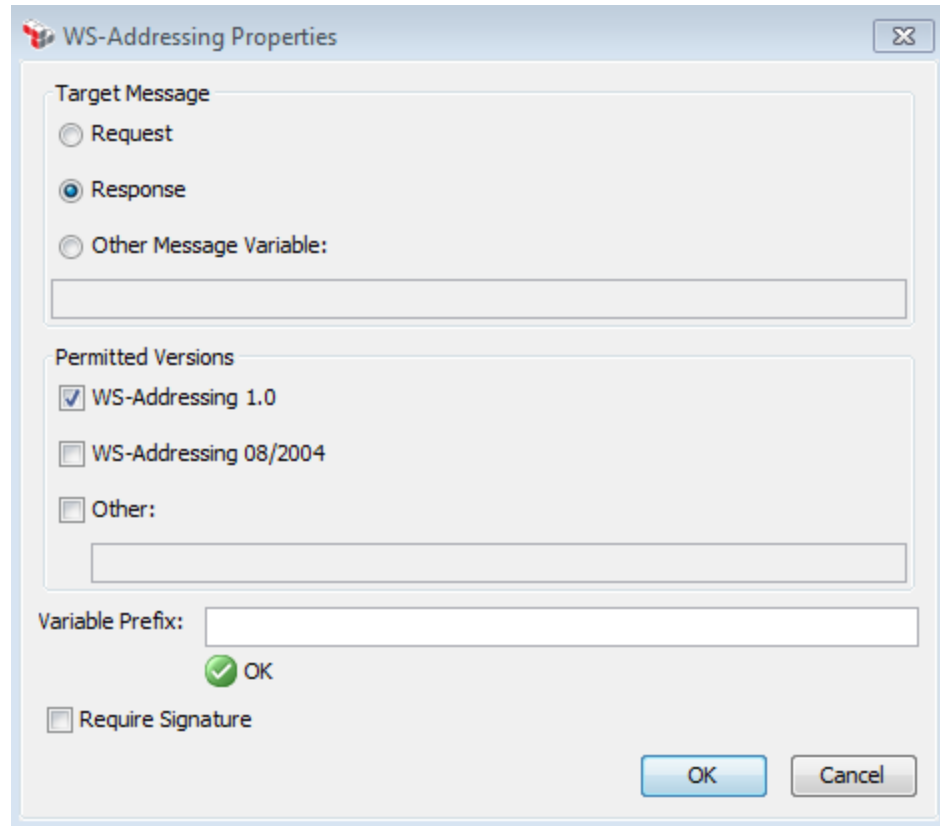
Variable	Description
	Value from sample message: <i>http://example.com/fabrikam/Purchasing</i> .
<code>\${<prefix>.action}</code>	Contains the value from the "action" addressing header. Value from sample message: <i>http://example.com/fabrikam/SubmitPO</i> .
<code>\${<prefix>.elements}</code>	Contains all the addressing headers, as a multivalued context variable. From the sample message, these are all the lines with "<wsa>", except for "<wsa:Address>".
<code>\${<prefix>.messageid}</code>	Contains the value from the "messageid" addressing header. Value from sample message: <i>http://example.com/6B29FC40-CA47-1067-B31D-00DD010662DA</i> .
<code>\${<prefix>.from}</code>	Contains the value from the "from" addressing header. Relates to endpoints and will contain the address of the endpoint reference. This header is not present in the sample message.
<code>\${<prefix>.replyto}</code>	Contains the value from the "replyto" addressing header. Relates to endpoints and will contain the address of the endpoint reference. Value from sample message: <i>http://example.com/business/client1</i> .
<code>\${<prefix>.faultto}</code>	Contains the value from the "faultto" addressing header. Relates to endpoints and will contain the address of the endpoint reference. This header is not present in the sample message.
<code>\${<prefix>.namespace}</code>	Contains the value from the "namespace" addressing header. Will be set to the namespace of the processed WS-Addressing header(s). Value from sample message: <i>http://www.w3.org/2005/08/addressing</i> .

Note: Not every addressing property will be present in all messages. If a property is not present, the context variable will contain an empty string. If a message contains repeated addressing headers, the first acceptable set of headers is used (note that this may be the first referenced by a signature, not necessarily the first in document order).

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the **WS-Addressing Properties** automatically appear; when modifying the assertion, right-click **<target>: Require WS-Addressing** in

the policy window and select **WS-Addressing Properties** or double-click the assertion in the policy window. The assertion properties are displayed.



The dialog box titled "WS-Addressing Properties" contains the following sections:

- Target Message:** Three radio buttons: "Request", "Response" (selected), and "Other Message Variable:" followed by a text input field.
- Permitted Versions:** Three checkboxes: "WS-Addressing 1.0" (checked), "WS-Addressing 08/2004", and "Other:" followed by a text input field.
- Variable Prefix:** A text input field.
- Buttons:** A green checkmark icon with the text "OK" next to it, and a "Require Signature" checkbox.
- Footer:** "OK" and "Cancel" buttons.

Figure 160: WS-Addressing Properties

3. Configure the properties as follows:

Table 133: WS-Addressing settings

Setting	Description
Target Message	<p>Select the message to check for WS-Addressing:</p> <ul style="list-style-type: none"> • Request: The request message will be checked. • Response: The response message will be checked. • Other Context Variable: A context variable will be checked. This context variable must be of type Message and must be predefined or has been set in the policy prior to the Require WS-Addressing assertion. For more information on Message variables, see Context Variables in the <i>Layer 7 Policy Manager User Manual</i>. <p>Tip: The message target can also be set outside of the assertion properties. For more information, see "Selecting a Target Message" on page 153.</p>

Setting	Description
Permitted Versions	<p>Specify which versions of WS-Addressing are permitted in the message. The Gateway will use the namespace associated with the selected version.</p> <p>If any another WS-Addressing namespace URI should be permitted, select Other and enter the additional permitted URI in the adjacent text field.</p> <p>The assertion will fail if a supported version of WS-Addressing is not found in the message.</p>
Variable Prefix	<p>Optionally, enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>
Require Signature	<p>Select this check box if the addressing headers must be signed. The assertion will fail if a signature is not present or is invalid.</p>

- Click **[OK]** when done.

Set SAML Response Status Code Assertion

The *Set SAML Response Status Code* assertion lets you choose a SAML response status and place it into in a context variable. This variable can be used in the "Customize Error Response Assertion" on page 430 to help you troubleshoot possible errors.

Tip: The first four codes in the lists below are top-level codes. The rest are second-level/subordinate codes that can be used to provide more information on an error.

For SAML 2.0, the following response statuses are available:

```
urn:oasis:names:tc:SAML:2.0:status:Success
urn:oasis:names:tc:SAML:2.0:status:Requester
urn:oasis:names:tc:SAML:2.0:status:Responder
urn:oasis:names:tc:SAML:2.0:status:VersionMismatch
urn:oasis:names:tc:SAML:2.0:status:AuthnFailed
urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue
urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy
urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext
urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP
urn:oasis:names:tc:SAML:2.0:status:NoPassive
urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP
urn:oasis:names:tc:SAML:2.0:status:PartialLogout
urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded
urn:oasis:names:tc:SAML:2.0:status:RequestDenied
urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported
```

urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated
urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh
urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow
urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized
urn:oasis:names:tc:SAML:2.0:status:TooManyResponses
urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal
urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding

For SAML 1.1, the following response statuses are available:

Success
VersionMismatch
Requester
Responder
RequestVersionTooHigh
RequestVersionTooLow
RequestVersionDeprecated
TooManyResponses
RequestDenied
ResourceNotRecognized

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Set SAML Response Status Code** in the policy window and select **SAML Response Status Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

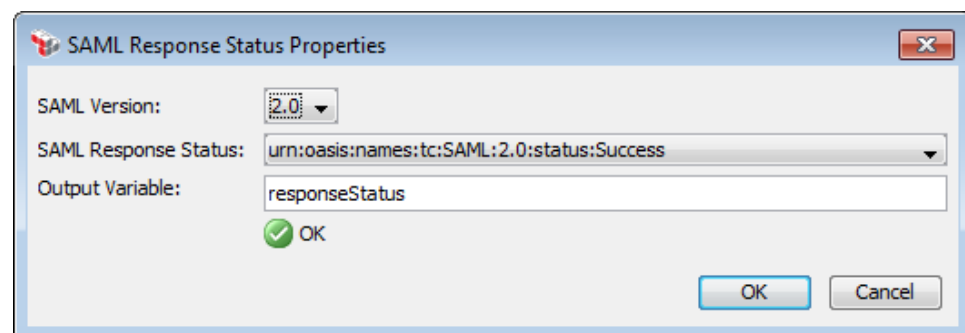


Figure 161: SAML Response Status Properties dialog

3. Configure the dialog as follows:

Table 134: SAML Response Status settings

Setting	Description
SAML Version	Choose the SAML version from the drop-down list: 1.1 or 2.0 .
SAML Response Status	Choose the SAML response status from the drop-down list.
Output Variable	Specify a context variable to hold the SAML response status. Default: responseStatus

Note: For SAML 1.1, the response status values are QNames associated with the namespace of the SAML protocol, where the output variable holds the local part of a QName. The local parts of these QNames are: *Success*, *VersionMismatch*, *Requester*, and *Responder*. For more information, see <http://saml.xml.org/saml-specifications>.

4. Click **[OK]** when done.

Translate HTTP Form to MIME Assertion

The *Translate HTTP Form to MIME* assertion allows you to parse the content submitted in HTTP form format by a browser-like client and turn it into a more standards-compliant multi-part MIME message. The Gateway converts incoming content to MIME in accordance with an ordered list of configured field names and Content-Types. If an incoming message does not contain a field configured for conversion in the Translate HTTP Form to MIME assertion, then the assertion will fail.

For information on parsing and converting a standard multi-part MIME message into HTTP form submission format, see the "Translate MIME to HTTP Form Assertion" on page 484.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **HTTP Form to MIME Translation Properties** automatically appear; when modifying the assertion, right-click **Translate HTTP Form to MIME** in the policy window and select **HTTP Form to MIME Translation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

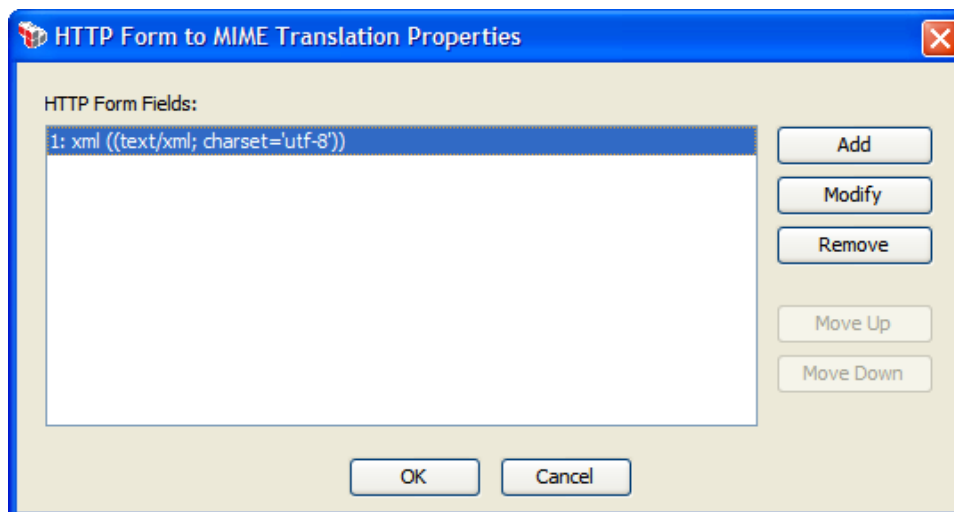


Figure 162: HTTP Form to MIME Translation Properties

3. Select an action to perform.

Table 135: HTTP Form to MIME Translation settings

Action	Description
Add	Click [Add] to configure a new HTTP form field.
Modify	Select an existing form field and then click [Modify] to edit the field information.
Remove	Select a form field to remove and then click [Remove] to delete it. The field is removed without further confirmation.
Move Up Move Down	The order of the form fields in the table must reflect the order of the MIME parts in the resulting MIME message. To change the order of the entries in the table, select an entry to be moved and then click [Move Up] or [Move Down] .

4. When adding or modifying a field, the Configure Field Information form appears.

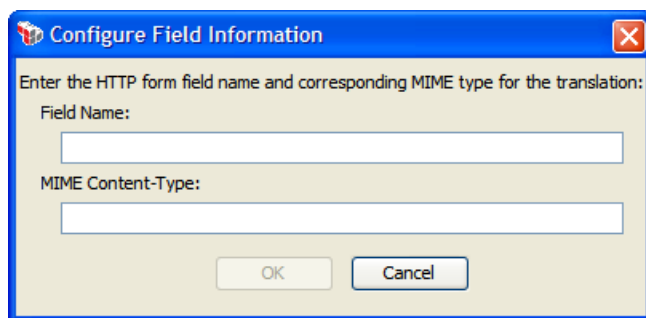


Figure 163: Configure Field Information form

Configure the form as follows:

Table 136: Configure Field Information

Setting	Description
Field Name	Enter the name of the HTTP form field. The field name must exactly match the name and case of the element in the incoming message; for example, <i>xml</i>
MIME Content-Type	Enter the MIME type for the form field, including a character set if the field contains textual data; for example, <i>text/xml; charset='utf-8'</i> In accordance with the SOAP with Attachments standard, the first MIME part should always be the SOAP message.

- Click **[OK]** to close each form when done.

Translate MIME to HTTP Form Assertion

The *Translate MIME to HTTP Form* assertion allows you to parse and convert an incoming multi-part MIME message into an HTTP form submission such as would be submitted by a browser-like client. The Gateway converts incoming MIME content to HTTP form format in accordance with an ordered list of configured field names. If an incoming message does not contain sufficient MIME parts for the field names configured for conversion in the Translate MIME to HTTP Form assertion, then the assertion will fail.

For information on parsing and converting incoming HTML form content into a multi-part MIME message, see the "Translate HTTP Form to MIME Assertion" on page 482.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the **MIME to HTTP Form Translation Properties** automatically appear; when modifying the assertion, right-click **Translate MIME to HTTP Form** in the policy window and select **MIME to HTTP Form Translation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

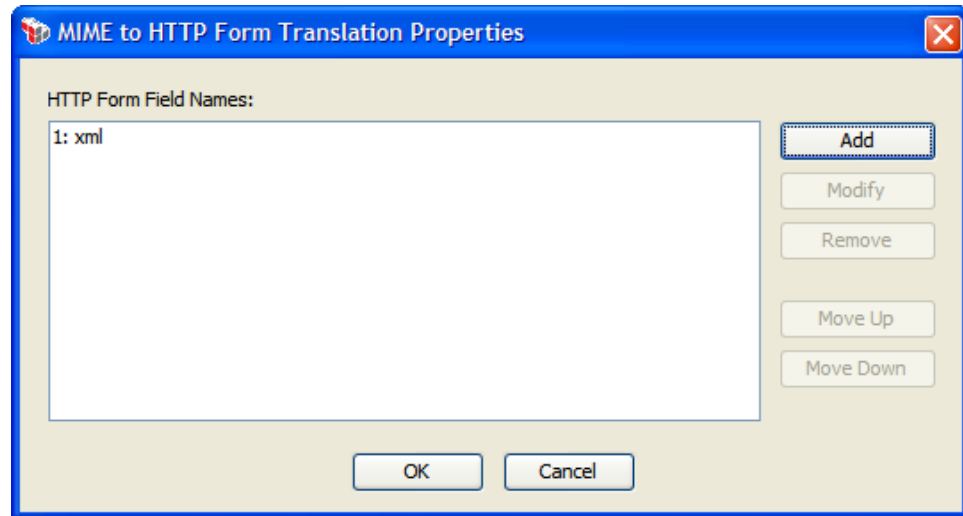


Figure 164: MIME to HTTP Form Translation Properties

3. Select an action to perform.

Table 137: MIME to HTTP Form Translation actions

Action	Description
Add	Click [Add] to add an HTTP form field name.
Modify	Select an existing form field and then click [Modify] to edit the name.
Remove	Select a form field name to remove and then click [Remove] to delete it. The name is removed without further confirmation.
Move Up Move Down	Since the Gateway converts the parts of each MIME message sequentially, the order of the form fields in the HTTP Form Field Names table must reflect the order of the MIME parts in the message. To change the order of the entries in the table, select an entry to be moved and then click [Move Up] or [Move Down].

4. When adding or modifying a field, the Configure Field Name dialog appears.

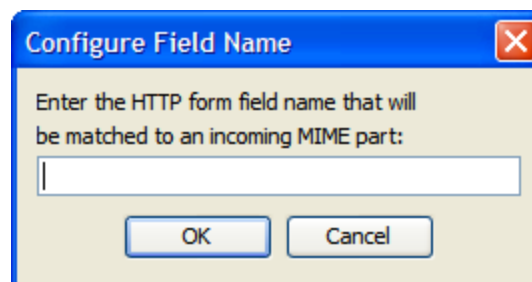


Figure 165: Configure Field Name form

Enter or modify the HTTP form field name. When the MIME message is transformed into an HTTP form post, the MIME part matching this field name's position will be rendered as "fieldname=<data>...".

5. Click **[OK]** to close each dialog when done.

Validate Certificate Assertion

The *Validate Certificate* assertion is used to validate an X.509 certificate context variable. Specifically, this assertion can validate that a certificate is not expired nor revoked, and that it has a valid chain.

IMPORTANT: A valid certificate does not ensure authentication. In other words, the Gateway does not check to ensure that the user possesses a private key.

Context Variables Created by This Assertion

The Validate Certificate assertion sets the following context variables with details of the validation. **Note:** The default <prefix> is "certificateValidation" and can be changed in the assertion properties (Figure 166).

Table 138: Context variables created by the Validate Certificate assertion

Variable	Description
<code>\${<prefix>.passed}</code>	Returns either True or False.
<code>\${<prefix>.error}</code>	Returns error message if validation fails.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Validate Certificate** in the policy window and select **Validate Certificate Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

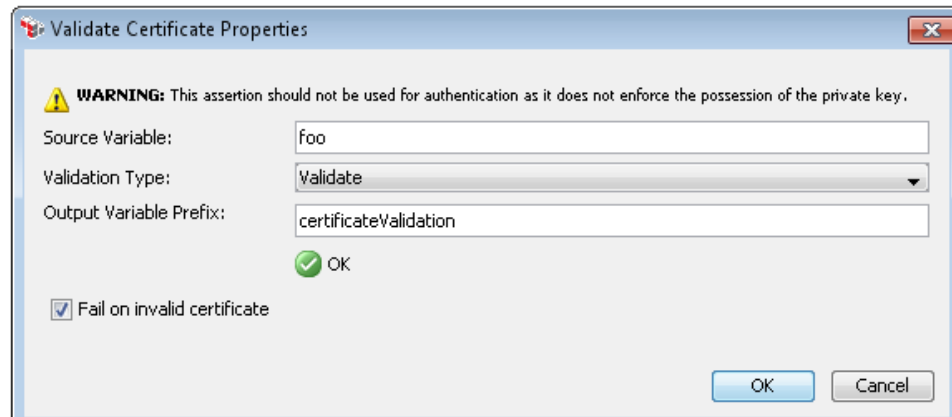


Figure 166: Validate Certificate Properties

3. Configure the properties as follows.

Table 139: Validate Certificate settings

Setting	Description
Source Variable	Enter the name of the context variable containing the X.509 certificate.
Validation Type	<p>Choose the level of validation from the Validation Type drop-down list</p> <ul style="list-style-type: none"> • Validate: Select this option to validate the expiration and format of the given certificate only. • Validate Certificate Path: Select this option to validate the certificate and build a path to a trust anchor. • Revocation Checking: Select this option to validate the certificate, build a path to a trust anchor, and perform a revocation check.
Output Variable Prefix	<p>Specify a prefix that will be added to the context variables created by this assertion. The prefix will prevent the context variable from being overwritten if the assertion appears more than once in a policy.</p> <p>Default: certificateValidation</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>
Fail on invalid certificate	<p>Select this check box to cause the assertion to fail and log an error when an invalid certificate is entered.</p> <p>Clear this check box to log an error but not fail the assertion upon an invalid certificate.</p>

4. Click **[OK]** when done.

Validate HTML Form Data Assertion

The *Validate HTML Form Data* assertion is used to validate the data set within an HTML form—for example, to require that a certain field must appear a minimum number of times or cannot appear more than once. You can specify which fields (i.e., form controls) are allowed, their data types, and their location in the request.

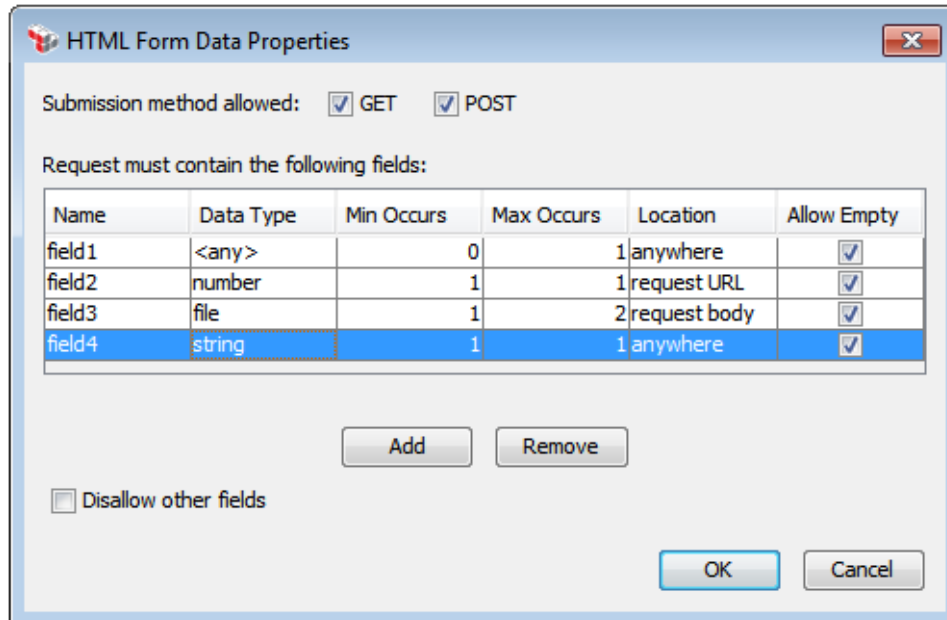
This assertion only works on HTTP requests; it is skipped if the request is not HTTP.

Ensure that this assertion appears before the routing assertion in the policy.

Tip: To further refine the allowable fields, include the [Compare Expression](#) assertion in the policy. For example, you are permitting only fields named "widget" with values over 100. To do this, define field *widget* with data type *number* in the *Validate HTML Form Data* assertion. In the *Compare Expression* assertion, add "*widget* > 100". The *Compare Expression* assertion can precede or follow the *Validate HTML Form Data* assertion. If you need to access the HTTP form parameters, use the `${request.http.parameter}` context variable.

Using the Assertions

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **HTML Form Data Properties** automatically appear; when modifying the assertion, right-click **Validate HTML Form Data** in the policy window and select **HTML Form Data Properties** or double-click the assertion in the policy window. The assertion properties are displayed.



HTML Form Data Properties

Submission method allowed: ☒ GET ☒ POST

Request must contain the following fields:

Name	Data Type	Min Occurs	Max Occurs	Location	Allow Empty
field1	<any>	0	1	anywhere	<input checked="" type="checkbox"/>
field2	number	1	1	request URL	<input checked="" type="checkbox"/>
field3	file	1	2	request body	<input checked="" type="checkbox"/>
field4	string	1	1	anywhere	<input checked="" type="checkbox"/>

Add Remove

☐ Disallow other fields

OK Cancel

Figure 167: HTML Form Data Properties

3. Configure the properties as follows:

Table 140: HTML Form Data settings

Setting	Description
Submission method allowed	<p>Select which submission methods are allowed: GET, POST. Requests made using other HTTP methods will cause the assertion to fail.</p> <p>You must select at least one method .</p>
Request must contain the following fields:	<p>Define the fields that are permitted in the request. The assertion succeeds only when a message contains <u>all</u> the listed fields, with the appropriate details.</p> <ul style="list-style-type: none"> To add a field, click [Add] and then enter the field information as described below. To remove a field, click anywhere in the row to select it, then click [Remove]. The field is removed immediately. <p>Complete the field details as follows:</p> <ul style="list-style-type: none"> Name: Type the name of the field. All names must be unique. The name is case sensitive. Data Type: Double-click and select which data type to allow: number, file, string, or <any>. (Note: The data type <i>file</i> requires the submission method <i>POST</i>.) Min Occurs: Enter the minimum number of times the field must appear in the request. To indicate that the field is optional (i.e., may or may not be present), enter a value of 0 (zero).

Setting	Description
	<ul style="list-style-type: none"> • Max Occurs: Enter the maximum number of times the field is allowed to appear in the request. The maximum may be the same as the minimum if you wish to enforce a specific number of occurrences. • Location: Double-click and specify where the field must be located in the request: within the request URL, request body, or anywhere in the request. (Note: The location <i>request body</i> requires the submission method <i>POST</i>.) • Allow Empty: Select this check box to allow the field to have an empty value. (Note: By default, when a policy using the Number data type is imported from a previous version, this check box will be deselected by default.)
Disallow other fields	<p>Indicate how you want to treat all other fields not specified in the table:</p> <ul style="list-style-type: none"> • Select this check box to allow <i>only</i> the listed fields in the request. The presence of any other fields will cause the assertion to fail. This makes the assertion more restrictive. • Clear this check box to allow any other field in the request <i>in addition</i> to the fields listed in the table. This makes the assertion more broad.

4. Click **[OK]** when done.

Validate JSON Schema Assertion

The *Validate JSON Schema* assertion is used to validate JSON (JavaScript Object Notation) data against a JSON schema. Specifically it will:

- validate JSON data structure
- validate JSON data property types
- validate JSON data property values

The JSON schema can either be defined within the assertion, or the assertion can monitor a URL or extracted the URL from a Content-Type or Link header.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To adjust how JSON schemas are cached, refer to these cluster properties:

```
json.schemaCache.maxAge
json.schemaCache.maxDownloadSize
json.schemaCache.maxEntries
```

Tips: (1) If the JSON schema validation fails, the reason is stored in the `$(jsonschema.failure)` context variable. (2) Place a "Protect Against JSON Document Structure Threats Assertion" on page 678 before this assertion to protect against DOS attacks

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the JSON Schema Validation Properties automatically appear; when modifying the assertion, right-click **<target>: Validate JSON Schema** in the policy window and select **JSON Schema Validation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

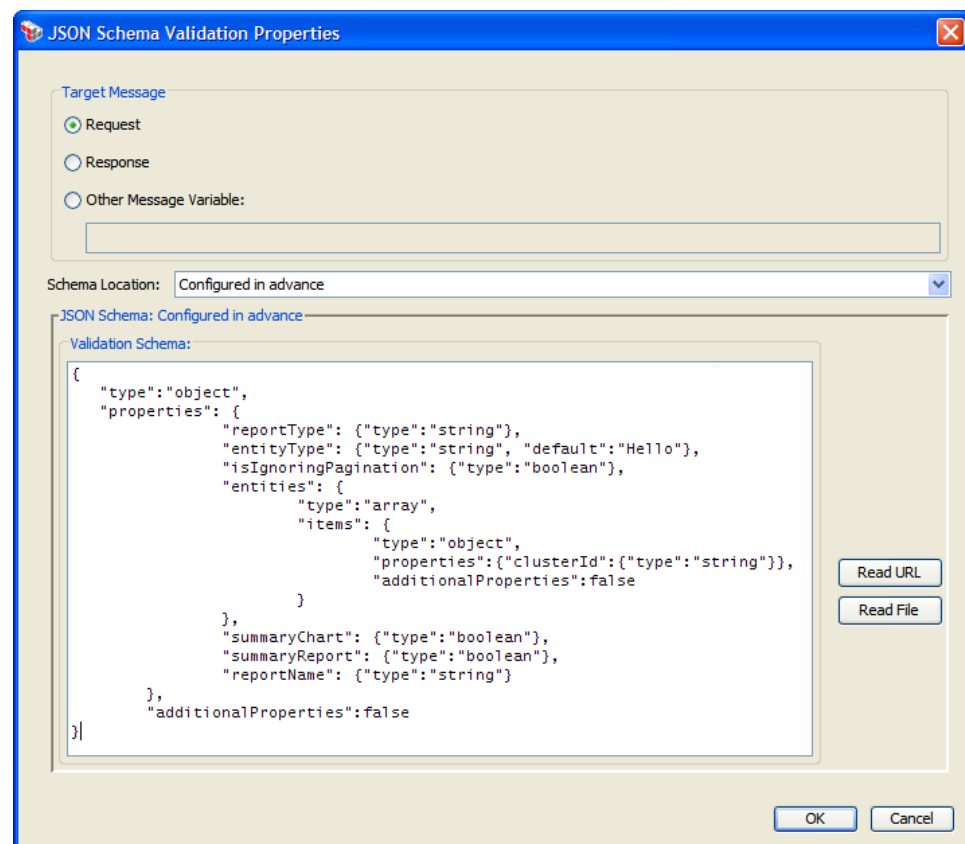


Figure 168: JSON Schema Validation Properties


3. Specify the target message to be validated:


- **Request:** Select this to validate the request message. This is the default setting if the assertion is positioned before the routing assertion in the policy.
- **Response:** Select this to validate the response message. This is the default setting if the assertion is positioned after the routing assertion in the policy.
- **Other Variable:** Select this to validate JSON content stored in a context variable. This variable will normally be either a String or a Message variable with Content-Type 'application/json' or another Content-Type that allows text. This variable must be predefined or has been set in the policy prior to the Validate JSON Schema assertion. For more information on Message variables, see Context Variables in the *Layer 7 Policy Manager User Manual*.

Tip: The message target can also be set outside of the assertion properties. For more information, see "Selecting a Target Message" on page 153.

4. From the **Schema location** drop-down list, specify where the schema is coming from:

Table 141: Configuring the JSON schema based on location

Setting	Description
Configure in advance	<p>Select this option to define a JSON schema directly.</p> <ol style="list-style-type: none"> 1. Specify the JSON schema using any of the following methods: <ul style="list-style-type: none"> • Manually type the code into the Validation Schema box or copy and paste the code from another source. Variables may be used. The assertion will check the input for correct JSON structure, but it will not validate any variables entered. <p> Tip: You can use the ".mainpart" suffix on variables of type Message and with Content-Type 'application/json'. For more information about this suffix, see "Context Variable Data Types" under Context Variables in the <i>Layer 7 Policy Manager User Manual</i>.</p> • Load the schema from a URL by clicking [Read URL] and then specifying the URL. <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> <ul style="list-style-type: none"> • Load the schema from a local file by clicking [Read File] and then browsing to the appropriate file. <p>Tip: The schema maximum size is controlled by the <code>schemacache.maxSchemaSize</code> cluster property.</p> 2. Review the content of the Validation Schema box and edit if necessary.

Setting	Description
Monitor URL for latest value 	<p>Select this option to specify a URL for the JSON schema. The Gateway monitors the external resources for changes over time.</p> <p>Type the address in the URL to monitor field. The URL may contain context variables that will be resolved at run time. By default, Gateway will issue an <i>If-Modified-Since</i>: HTTP request for this URL approximately every 5 minutes while the schema is in use.</p> <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> <p>Tip: The monitor time interval is controlled by the <i>json.schemaCache.maxAge</i> cluster property.</p>
Retrieve Schema URL from Content-Type or Link Header	<p>Select this option to retrieve the JSON schema URL from either a Content-Type profile parameter in the header or from a Link header.</p> <p>Example of a MIME type parameter: 'profile':</p> <p><i>Content-Type: application/json; profile=http://json.com/my-hyper-schema</i></p> <p>Example of a "describedby" HTTP header:</p> <p><i>Link: <http://json.com/my-hyper-schema>; rel="describedby"</i></p> <ul style="list-style-type: none"> • Use [Add] to add as many regular expressions as necessary to determine if a URL belongs to the set of white-listed URLs. • Use [Edit] to modify any of the regular expressions. • Use [Delete] to remove a regular expression from the list. • Select the Skip validation... check box to allow the assertion to succeed if there is no schema URL in the message. Clear this check box to always check for a schema URL (the assertion will fail if not found). <p>Note: The Content-Type parameter is checked first; if a URL is not found, then the header values are checked next.</p>

5. Click **[OK]** when done.

Validate MTOM Message Assertion

The *Validate MTOM Message* assertion is used to validate MTOM-optimized messages. MTOM-optimized messages are based on the Message Transmission Optimization Mechanism (MTOM) specification.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

MIME Multipart Messages

A SOAP service normally does not permit MIME multipart messages unless it is explicitly specified in the service's WSDL document or the policy contains an that processes MIME multipart. Adding a Validate MTOM Message assertion that targets a request message in a policy will mean that the service(s) for that policy will permit MIME multipart messages.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click **<target>: Validate MTOM Message** in the policy window and select **MTOM Validate Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

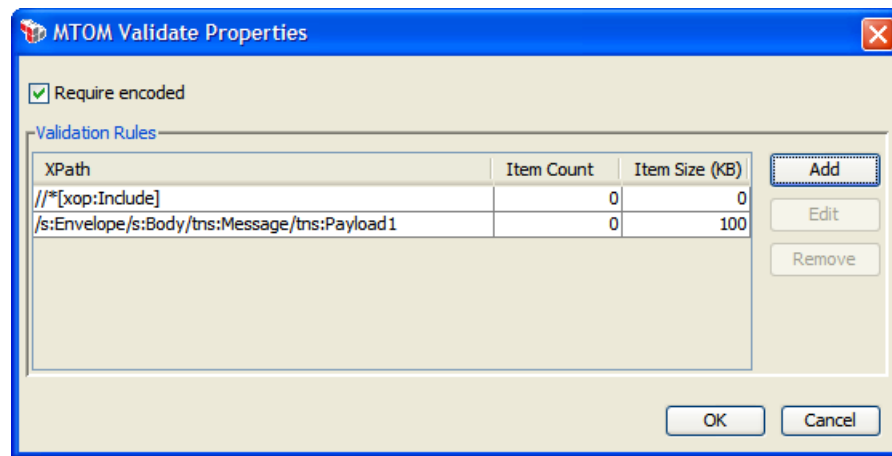


Figure 169: MTOM Validate Properties

- Configure the properties as follows:

Table 142: MTOM Validate Settings

Setting	Description
Require encoded	<p>Select this check box to require that the source message be MTOM-encoded, otherwise the assertion will fail.</p> <p>Clear this check box to not fail the assertion if the source message is not MTOM-encoded.</p>
Validation Rules	The Validation Rules table is used to control which element's content

Setting	Description
	<p>will be validated. For each XPath, you can specify a maximum Item Count and maximum Item Size.</p> <ul style="list-style-type: none"> To add a validation rule, click [Add]. To modify a validation rule on the list, select it and then click [Edit]. To remove a validation rule from the list, select it and then click [Remove].

- When adding or modifying a validation rule, the MTOM Validate Properties - Rule dialog appears:

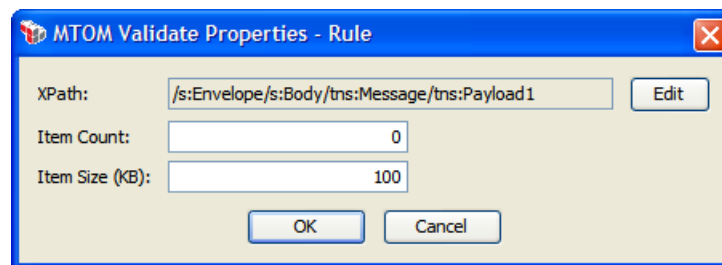


Figure 170: MTOM Validate Properties - Rule

Complete the fields as follows:

- XPath:** Click **[Edit]** to select the XPath containing the elements to validate. For more information, see "Selecting an XPath" on page 154.
- Item Count:** Enter the maximum number of items permitted at the XPath. A count of '0' (zero) means an unlimited number of items.
- Item Size:** Enter the maximum size permitted for an item, in kilobytes.

- Click **[OK]** when done.

Validate or Change Content Type Assertion

The *Validate or Change Content Type* assertion can be used to validate or change the Content-Type of any target message. You can target specific parts of a multi-part MIME message.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

- Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Validate Content Type** or **<target>: Change Content Type to <ContentType>** in the policy window and select **Content Type Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

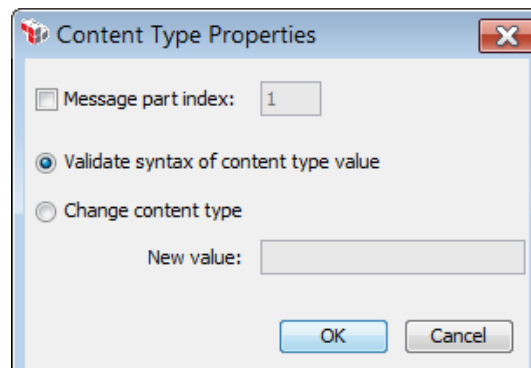



Figure 171: Content Type Properties

3. Configure the properties as follows:

Table 143: Rate Limit settings

Setting	Description
Message part index	<p>Select this check box to target the Content-Type in a specific MIME part in the message.</p> <p>Clear this check box to target the Content-Type in the main/root MIME part of the message (i.e., MIME part '1').</p>
Validate syntax of content type	<p>Select this option to validate the syntax of the value of the Content-Type of the targeted message or message part.</p> <ul style="list-style-type: none"> • If the validation succeeds, the assertion succeeds. • If the validation fails, the assertion fails and returns the assertion status code 601 ("Error in assertion processing.")
Change content type	<p>Select this option to modify the Content-Type value of the targeted message or message part.</p> <p>Note: The modification takes effect immediately, but any previous message processing based on the old Content-Type will not be undone.</p>
New value 	<p>When changing the Content-Type, enter the new value here. You can also enter a context variable to set the value at the time of policy execution.</p>

Setting	Description
	Note: The value must be a valid and complete MIME type as defined by RFC 2045 and 2046.

- Click **[OK]** when done.

Validate SOAP Attachments Assertion

The *Validate SOAP Attachments* assertion allows you to validate the size and MIME type of incoming SOAP attachments. The assertion will fail under any of the following conditions:

- the request message does not contain an attachment
- the request message contains an attachment that was not declared in the WSDL
- the attachment is too large
- the attachment is declared an MIME Content-Type different from the expected type
- a signature is required but not present for the attachment.

You can optionally require that the attachment be signed (not available in the XML Datascreen version of the Gateway).

The Validate SOAP Attachments assertion supports the W3C *SOAP Messages with Attachment* standard as outlined in www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211, and the *OASIS Web Services Security SOAP Messages with Attachment (SwA) Profile 1.0 (Committee Draft)* for signed attachments.

Notes: (1) You cannot use the Validate SOAP Attachments assertion with XML applications. (2) If a signature is required for an attachment, one of the following assertions must precede the SOAP Request with Attachment assertion: [Require WS-Security Signature Credentials](#), [Require WS-Secure Conversation](#), [Require SAML Token Profile](#), [Require Encrypted UsernameToken Profile Credentials](#), or [Require WS-Security Kerberos Token Profile Credentials](#).

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.

2. Right-click **Validate SOAP Attachments** in the policy window and select **SOAP Attachment Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

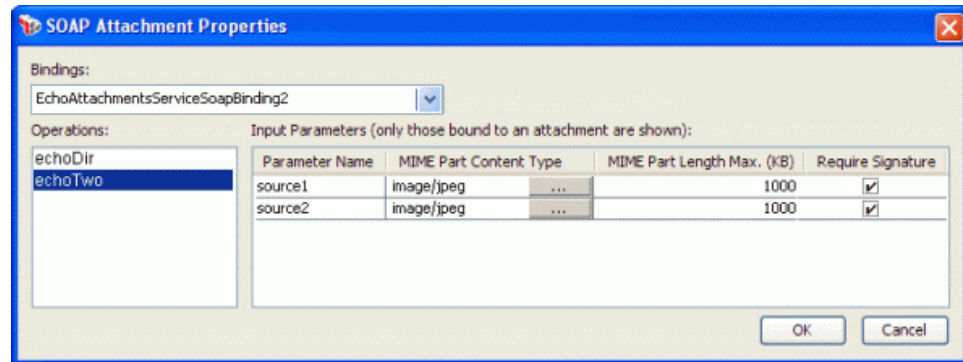


Figure 172: SOAP Attachment Properties

The **Binding**, **Operations**, and **Input Parameters** corresponding to one or more attachments are automatically populated from the web service WSDL document.

Note: If no attachments are present or your service is not capable of supporting attachments, then nothing will appear in the properties dialog.

3. From the **Binding** drop-down list, select the binding that contains the attachment. The operation(s) belonging to the binding appear under **Operations**.
4. From the **Operations** list, select the operation that contains the attachment(s). The following information is displayed in the **Input Parameters** grid.

Table 144: SOAP Attachment Properties settings

Column	Description
Parameter Name	Name of the input parameter for the attachment.
MIME Part Content Type	The Content-Type is retrieved from the WSDL document. If it is not correct, click [...] to change it.
MIME Part Length Max.	Set to default size of 1000 KB by the Gateway. Modify as necessary.
Require Signature	Select this check box to require that the attachments be signed. This option does not apply to the XML Datascreen version of the Gateway. WARNING: Signatures with attachments cannot be verified when the message is save as part of auditing, as the signed attachment is not saved. Modifying an attachment will most likely break the signature of the attachment.

Multiple attachments per input parameter are also supported. In this case, the total size of the attachments being referred to by the input parameter cannot exceed the value of the MIME Part Length Max column value corresponding to the input parameter.

5. Click **[OK]** when done.

Validate XML Schema Assertion

The *Validate XML Schema* assertion allows you to specify a schema for validating a web service or XML application request or response messages. This assertion can be used to protect backend web services against the following threats:

- **XML Parameter Tampering:** All XML parameters in the request are validated to ensure conformance with the XML schema specifications. This is to prevent injection of malicious scripts or content into the request parameters.
- **XDoS Attacks:** The message structure and content are examined to ensure that they are correct.

A message schema is provided by the Gateway administrator. If the service's WSDL contains a schema, then that schema can be extracted to serve as the starting point for the schema used in the Validate XML Schema assertion. This WSDL schema can be extracted in whole or in request or response message-specific parts.

If the schema contains import statements that refer to external schemas, the Policy Manager will attempt to fetch all unresolved schemas in an import tree (for example, a schema referencing another schema) and add them to the global schema table. You can view these imported schemas using the [Manage Global Resources](#) task. If the Policy Manager is unable to resolve a schema (for example, because of a bad URL or URI), you will be prompted to manually add the schema.

Tip: The format of the import statement can affect how it is received by the Gateway. A full URL path is most preferable and is always resolvable (e.g., "http://schema.example.com/test.xsd"). Just the file name is acceptable, provided that the exact name can be located in the Global Schemas stored in the Gateway (e.g., "test.xsd"). Not acceptable are paths containing a specific drive letter (e.g., "f:\test.xsd"), or relative paths such as "../test.xsd".

A policy can contain multiple Validate XML Schema assertions. The runtime application of a schema is determined by its placement in the policy path. If routing has already occurred when the Validate XML Schema assertion initiates, then the schema will be applied to the response message. If routing has not yet occurred, then the schema will be applied to the request message.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Schema Failure in Context Variable

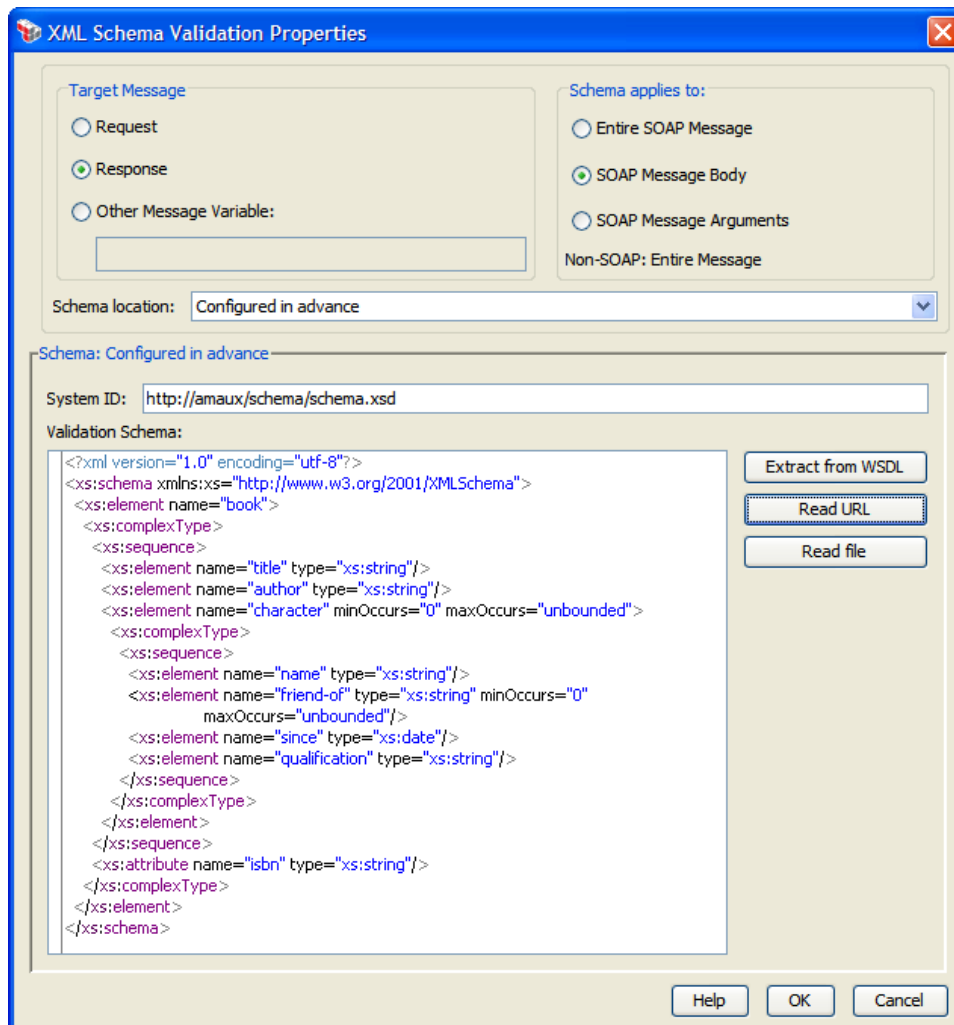
When a schema validation fails, an audit record is created and the reason for failure is placed in the context variable `${schema.failure}`. This makes it possible to reference the failure later in the policy (for example, inclusion in the [Return Template Response to Requestor](#) assertion).

Schemas with Circular References

A "circular reference" occurs when a schema references other schemas that ultimately point back to the original schema. The Policy Manager will fetch all schemas from a destination, circular or not, and add them to the global schemas table.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the Schema Validation Properties automatically appear; when modifying the assertion, right-click **<target>: Validate XML Schema** in the policy window and select **XML Schema Validation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.



The dialog box is titled "XML Schema Validation Properties". It contains two main sections: "Target Message" and "Schema applies to:".

Target Message:

- ☐ Request
- ☒ Response
- ☐ Other Message Variable:

Schema applies to:

- ☐ Entire SOAP Message
- ☒ SOAP Message Body
- ☐ SOAP Message Arguments
- ☐ Non-SOAP: Entire Message

Schema location: Configured in advance

Schema: Configured in advance

System ID:

Validation Schema:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="author" type="xs:string"/>
        <xs:element name="character" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name" type="xs:string"/>
              <xs:element name="friend-of" type="xs:string" minOccurs="0"
                maxOccurs="unbounded"/>
              <xs:element name="since" type="xs:date"/>
              <xs:element name="qualification" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="isbn" type="xs:string"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Buttons on the right: Extract from WSDL, Read URL, Read file.

Buttons at the bottom: Help, OK, Cancel.

Figure 173: XML Schema Validation Properties

3. Specify the target message to be validated:

- **Request:** Select this to validate the request message. This is the default setting if the assertion is positioned before the routing assertion in the policy.
- **Response:** Select this to validate the response message. This is the default setting if the assertion is positioned after the routing assertion in the policy.
- **Other Message Variable:** Select this to validate a message stored in a context variable of type 'Message'. This variable must be predefined or has been set in the policy prior to the Validate XML Schema assertion. For more information on Message variables, see Context Variables in the *Layer 7 Policy Manager User Manual*.

Tip: The message target can also be set outside of the assertion properties. For more information, see "Selecting a Target Message" on page 153.

4. For SOAP messages, specify the portion of the message that will be validated by the schema. For non-SOAP messages, the schema will be applied to the entire message.

- **Entire SOAP Message**

Schema validation is performed on the entire SOAP envelope.

The schema configured by the policy author in this case should be based on the SOAP envelope schema. It may optionally include definitions that cover the payload of the SOAP headers and/or the SOAP body.

If you need to validate a schema against the SOAP message including any security elements in the header (for example, signature element), you should additionally import the WS-Security schema in your custom schema (for example, <http://schemas.xmlsoap.org/ws/2002/04/secext/secext.xsd>).

- **SOAP Message Body**

Apply the schema to each element under the *soap:Body* element in a SOAP message. This setting is the default.

Note: When importing an RPC/literal-style WSDL using this option, the system will prompt you with: "The WSDL style seems to indicate that the schema validation should be applied to the body 'arguments' rather than the entire body. Would you like to change the setting accordingly?" Answer 'Yes' only if you are certain that the web service is RPC/literal-style.



- **SOAP Message Arguments**

Apply the schema to the children elements under the first child element under the *soap:Body*. This is typically used in RPC/literal-style web services where the argument elements themselves are not declared in the schema.

5. From the **Schema location** drop-down list, specify where the schema is coming from:

Table 145: Configuring the schema based on location

Setting	Description
Configure in advance	<p>Select this option to define a root schema and all dependencies directly.</p> <ol style="list-style-type: none"> 1. Specify the schema using any of the following methods: <ul style="list-style-type: none"> • Manually type the code into the Validation Schema box or copy and paste the code from another source.

Setting	Description
	<ul style="list-style-type: none"> If the Gateway can detect a schema in the WSDL document, you can click [Extract Schema from WSDL] to import the schema from the WSDL document. A WSDL-based schema is typically only included in document-style web services. Complete the Extract Schema from WSDL dialog in step 6 below. Load the schema from a URL by clicking [Read URL] and then specifying the URL.  <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> <ul style="list-style-type: none"> Load the schema from a local file by clicking [Read File] and then browsing to the appropriate file. <p>The System ID field is automatically populated when opening a resource (from the WSDL, a URL, or a file).</p> <p>Notes: (1) The schema maximum size is controlled by the <i>schemacache.maxSchemaSize</i> cluster property. (2) If the cluster property <i>schema.allowDoctype</i> is set to "true", then the "Configure in advance" XML schema may contain a document type definition (DTD); otherwise, DTDs are not permitted (default).</p> <p>2. Review the content of the Validation Schema box and edit if necessary. You can right-click within the box for some useful tools to help you edit. For more information, see Using the XML Editor in the <i>Layer 7 Policy Manager User Manual</i>.</p>
Monitor URL for latest value 	<p>Select this option to specify a URL for the root schema. The Gateway loads all the dependencies and then monitors the external resources for changes over time.</p> <p>Type the address in the URL to monitor field. The URL may contain context variables that will be resolved at run time. By default, Gateway will issue an <i>If-Modified-Since</i>: HTTP request for this URL approximately every minute while the schema is in use.</p> <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> <p>Note: The schema maximum size is controlled by the <i>schemacache.maxSchemaSize</i> cluster property.</p>
Pick XML Schema from global resources	<p>Select this option to pick the validation schema from the global resources table. Choose the global schema to use from the Selected schema drop-down list. If the schema you require is not listed, click [Manage Global Resources] to define it, or to modify or remove other global resources defined in the system. For more information, see "Managing Global Resources" on page 72.</p>

6. If you chose to extract the schema from a WSDL document, the following dialog appears:

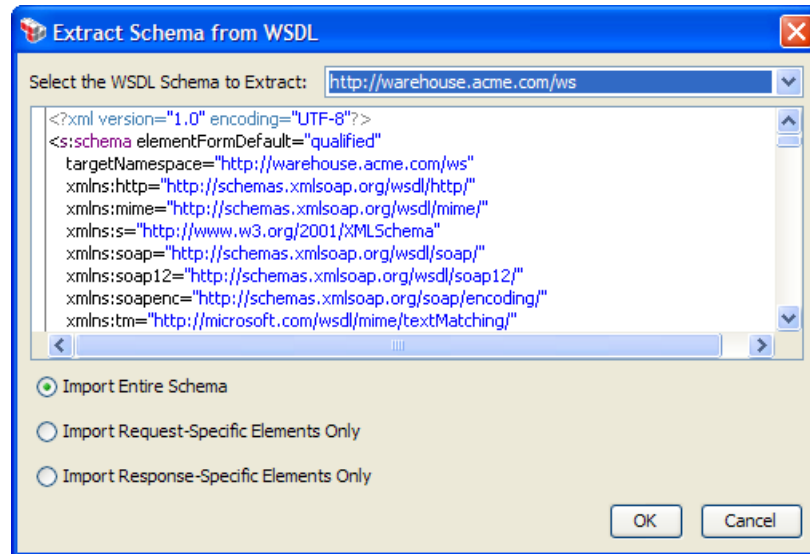


Figure 174: Extract Schema from WSDL dialog

Configure the dialog as follows and then click **[OK]** when done:

Table 146: Extract Schema from WSDL settings

Setting	Description
Select the WSDL Schema to Extract	<p>If the WSDL document contains more than one schema, select the schema to use from the drop-down list. The schema code is displayed in the box below.</p> <p>Note: The Validate XML Schema assertion only takes a single schema as input. If the WSDL contains multiple schemas, it may be necessary to reorganize those schemas into one root schema that references other schemas through import statements. The Policy Manager attempts to retrieve the schemas referenced by the import statements and add them to the global schema table. To view these schemas, see "Managing Global Resources" on page 72.</p>
Import Entire Schema	Extract the entire schema. This setting is the default.
Import Request-Specific Elements Only	Extract only the schema elements particular to request messages.
Import Response-Specific Elements Only	Extract only the schema elements particular to response messages.

7. When a resource with dependencies is opened, you are prompted to confirm whether to import the schema's dependencies as global resources. Select **[Import]** to import the dependencies or **[Skip]** to exclude the dependencies. Select **[Cancel]** to cancel the loading of the resource (whether from the WSDL, a URL, or a file).
8. If you chose **[Import]** in the previous step, all the schema dependencies that will be processed and potentially added as global resources (Figure 175) are listed. Review the list carefully and note the *Action* column for each resource:
 - *Ignore*: The resource will not be imported.
 - *Update*: The resource will update an existing global resource.
 - *Create*: A new global resource will be created for the resource.

Select **[Import]** to update the global resources or **[Skip]** to not update the global resources. Select **[Cancel]** to cancel the import.

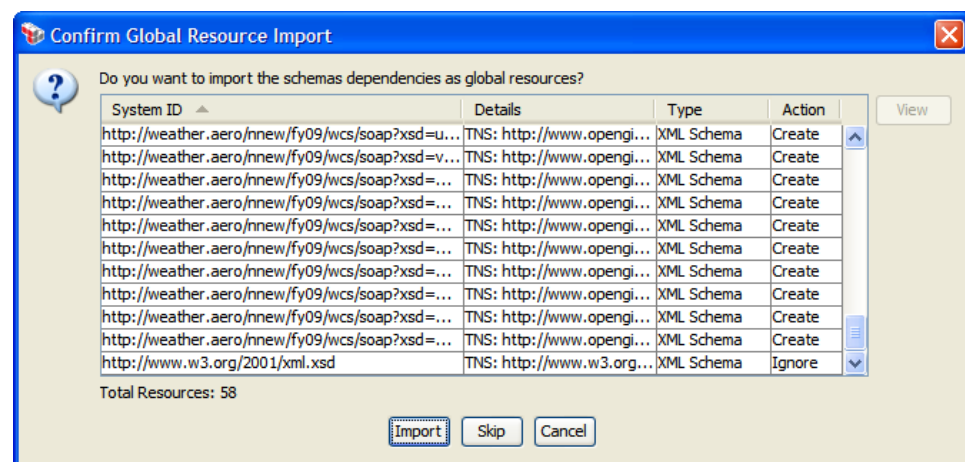


Figure 175: Confirming importing schema dependencies

9. During import, if there are issues that require manual intervention, you will be prompted with a dialog similar to Figure 176

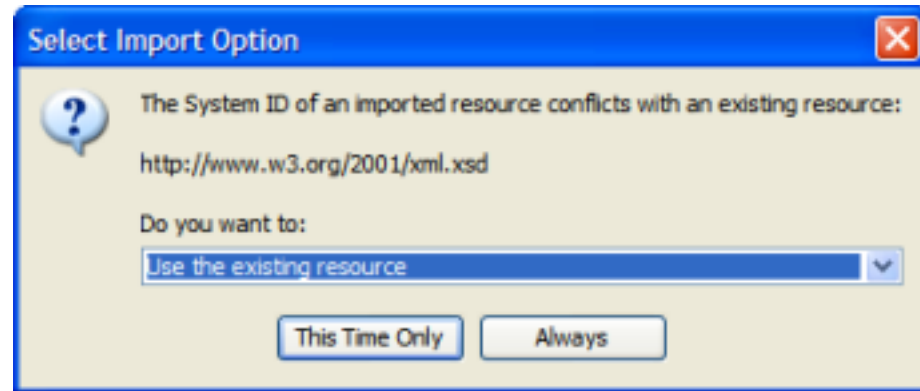


Figure 176: Select Import Option dialog

Select a resolution from the drop-down list, then specify whether:

- **[This Time Only]**: Use the selected action only for this occurrence of the conflict. When another similar conflict occurs, you will be asked again how to resolve it.
 - **[Always]**: Use the selected action for all the conflicts of this type. You will not be prompted for a resolution if another similar conflict occurs during this import.
10. On the XML Schema Validate Properties, click **[OK]** when done. If the dependencies of a configured in advance XML Schema are found then the assertion is added to the policy development window. If the Policy Manager is unable to validate the dependencies, you are prompted to manually add the unresolved schema(s).

Chapter 8: Message Routing Assertions

Notes: (1) Depending on which Gateway product you have installed, not all the assertions shown below may be available. See Features by Product in the *Layer 7 Policy Manager User Manual* for a list of which features are available for each product. (2) This category may also include custom-created encapsulated assertions. For more information, see "Working with Encapsulated Assertions" on page 126.

In the Policy Manager, the following assertions are available in the Message Routing category of the [Assertions] tab:

Configure Message Streaming Assertion	508
Copy Request Message to Response Assertion	510
Manage Cookie Assertion	512
Manage Transport Properties/Headers Assertion	515
Return Template Response to Requestor Assertion	518
Route via FTP(S) Assertion	520
FTP Cluster Properties for This Assertion	522
Configuring the [Connection] Tab	523
Configuring the [Authentication] Tab	526
Configuring the [Advanced] Tab	528
Route via HTTP(S) Assertion	529
Configuring the [Authentication] Tab	531
Configuring the [Headers] Tab	533
Configuring the [Connection] Tab	536
Configuring the [HTTP] Tab	538
Configuring the [Proxy] Tab	539
Configuring the [Other] Tab	539
Route via JMS Assertion	541
Context Variables Created by This Assertion	542
Configuring the [Target] Tab	544
Configuring the [Security] Tab	546
Configuring the [Request] Tab	548
Configuring the [Response] Tab	550
Route via MQ Native Assertion	551
Context Variables Created by This Assertion	552
Defined MQ Header Prefixes	553
Configuring the [Target] Tab	554
Configuring the [Request] Tab	556
Configuring the [Response] Tab	558

Route via Raw TCP Assertion	560
Route via SSH2 Assertion	563
Performing SFTP Partial Downloads/Uploads	563

The Message Routing assertions define where service messages are sent and what access credentials are required by the back-end service. The Message Routing assertions are not concerned with the credential requirements of the Gateway. Use them to define the credential requirements of the target back-end system that will receive Gateway-routed service messages.

Configure Message Streaming Assertion

The *Configure Message Streaming* assertion allows large message requests to stream without being read and buffered by the Gateway. This assertion can reduce the latency of requests passing through the Gateway, but note that this will limit the types of validation and processing the Gateway is able to perform on these requests. An example of such a limitation is illustrated by the following policy:

```
Request: Configure Message Streaming: buffer immediately
Request: route to <URL of external validation or other service>
Request: route to <URL of protected service>
```

If this policy is changed to "enable streaming (no buffering)", the request body is streamed by the first routing assertion and will not be available to the second routing assertion.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click *<target>*: Configure Message Streaming in the policy window and select **Configure Message Streaming** or double-click the assertion in the policy window.

The assertion properties are displayed.

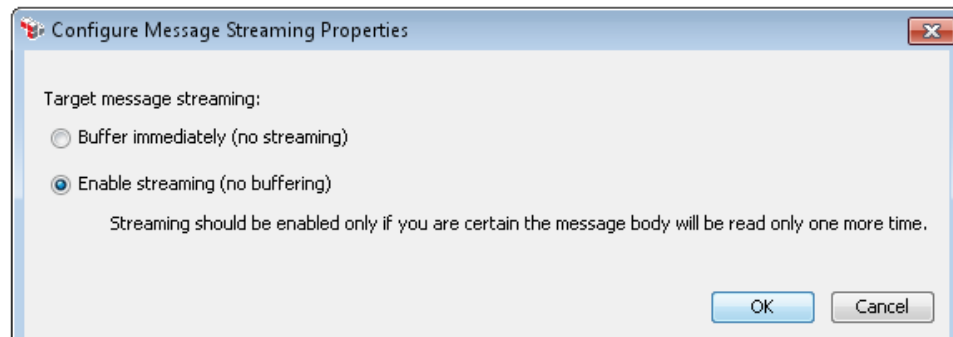


Figure 177: Configure Message Streaming Properties

3. Configure the properties as follows.

Table 147: Configure Message Streaming settings

Setting	Description
Buffer immediately (no streaming)	<p>Select this option to have the entire message read immediately and saved to the Gateway's internal buffer. If the total size exceeds the value of the <i>attachment.diskThreshold</i> cluster property, it will be buffered to disk. The target message will be buffered by the Gateway and will be available for repeated inspection or modification by subsequent operations.</p> <p>The assertion will fail if the target message has not yet been initialized (<code>\${<message variable>.buffer.status} = "uninitialized"</code>), or, if the target message has already been streamed without being buffered (<code>\${<message variable>.buffer.status} = "gone"</code>).</p> <p>For more information about the <code>\${<target>.buffer.status}</code> variable, see "Message Layer Variables" in Context Variables.</p>
Enable streaming (no buffering)	<p>Select this option to prevent this target message from being buffered by the Gateway.</p> <p>Note: Routine use of this option is recommended only if it is necessary to stream large requests through the Gateway without buffering.</p> <ol style="list-style-type: none"> The assertion will fail if the target message has not yet been initialized (<code>\${<message variable>.buffer.status} = "uninitialized"</code>), or, if it has already been buffered (<code>\${<message variable>.buffer.status} = "buffered"</code>). <p>The next operation that accesses the body of the target message will consume it in streaming mode. Any further attempt to access the body of this message will result in an error because it has already been streamed away.</p> <ol style="list-style-type: none"> When enabling streaming in a SOAP service, the following settings must be used in the Service Properties to ensure that

Setting	Description
	<p>streaming occurs correctly:</p> <ul style="list-style-type: none"> In the [General] tab, clear the Perform WS-Security processing for this service check box. In the [WSDL] tab, select the Allow requests intended for operations not supported by the WSDL check box.

- Click **[OK]** when done.

Copy Request Message to Response Assertion

The *Copy Request Message to Response* assertion copies the inbound request exactly as it appears at the current point in the policy, to the response. It provides a convenient method to review the results of a request without requiring a web service or an http destination.

The following are some examples on how you can use this assertion:

- When added after the [Apply XSL Transformation](#) or [Evaluate Regular Expression](#) assertions, the Copy Request Message to Response assertion will send the resulting message back to the client.
- When added after any of the [Threat Protection](#) assertions, the Copy Request Message to Response assertion will return the message to the client if the request passes all the threat protection assertions. If the request fails any assertion, a SOAP fault is generated and no message is returned.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the Request to Response Properties automatically appear; when modifying the assertion, right-click **Copy Request Message to Response** in the policy window and select **Request to Response Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

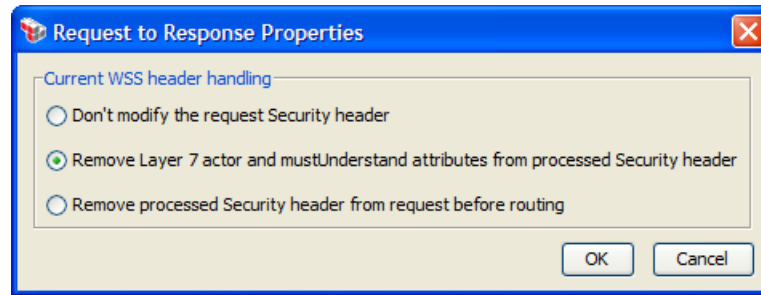


Figure 178: Request to Response Properties

3. Specify how to handle WSS Security headers in the request messages:

Table 148: Request to Response Properties settings

Option	Description
Don't modify the request Security header	<p>Instructs the Gateway to leave the security header in the outgoing SOAP request message as-is. The security header in the request may still have been modified if the Gateway needed to decrypt any encrypted material during message processing.</p> <p>Use this setting if the protected service needs to do its own checking of the request's original security header, or if the protected service does not care whether its request messages have a security header.</p> <p>For best performance, use this setting whenever possible to minimize the amount of per-request message modification.</p> <p>Note: Do not modify the Security header if the policy uses WS-Security. For more information, see the "Add or Remove WS-Security Assertion" on page 273.</p>
Remove Layer 7 actor and mustUnderstand attributes from processed Security header	<p>Instructs the Gateway to remove the 'mustUnderstand' attribute and 'Layer 7' actor from the security header in the outgoing SOAP message.</p> <p>Use this setting if the presence of the Layer 7 actor causes issues with the backend service. In certain cases, this actor may cause the backend service to ignore the Security headers because it believes it is addressed to someone else. You will also use this setting if the backend service does not support Security and would reject a request with 'mustUnderstand' asserted on the Security header.</p> <p>An alternative might be to remove the Security header completely, however this will incur a performance penalty for the extra processing required to remove these from the messages. You may want to keep the Security headers intact for logging purposes.</p>
Remove processed Security header from request before routing	<p>Instructs the Gateway to remove any security header that was processed by the gateway before forwarding the request to the protected service.</p> <p>Use this setting when the protected service is not expecting security headers in the forwarded SOAP requests.</p>

4. Click **[OK]** when done.

Manage Cookie Assertion

The *Manage Cookie* assertion is used to manipulate the cookies in a policy. This assertion is designed specifically for cookie configuration and provides more flexibility than the basic cookie handling capabilities offered in the [Manage Transport Properties/Headers](#) or the [Route via HTTP\(S\)](#) assertions.

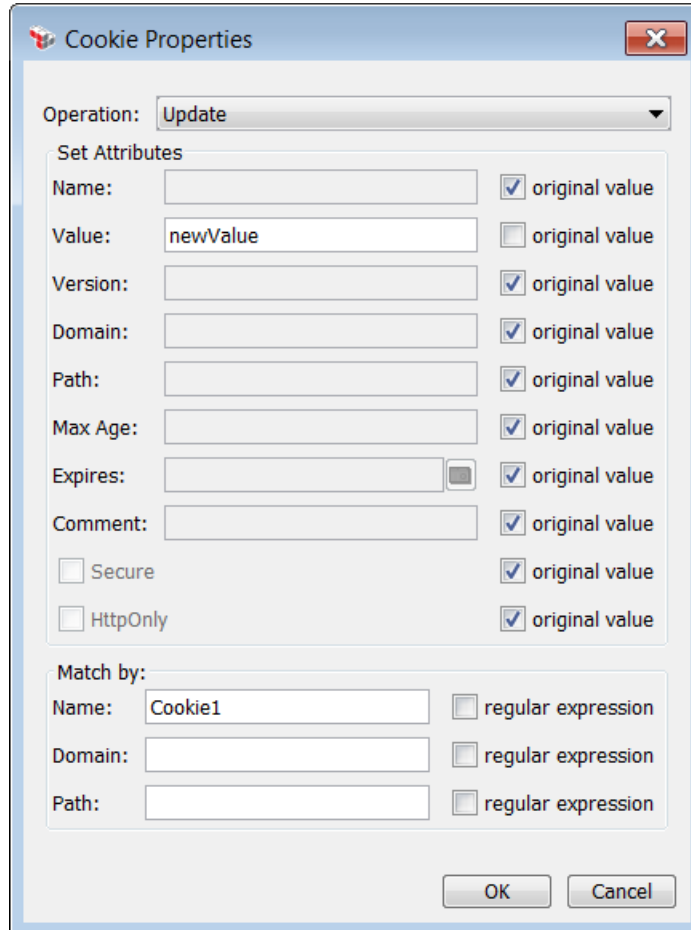
This assertion supports the original Netscape cookie specifications (http://curl.haxx.se/rfc/cookie_spec.html), as well as RFC 2109 (<http://tools.ietf.org/html/rfc2109>).

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Note: The Gateway may rewrite cookie attributes in order to track cookies origins or to ensure that the cookies will be sent back to the Gateway in subsequent requests. It is recommended that this automatic rewriting be maintained, but advanced users may disable the rewriting for troubleshooting purposes by setting the following context variables to 'false': `${response.cookie.overridePath}` and `${response.cookie.overrideDomain}`.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the Cookie Properties automatically appear; when modifying the assertion, right-click the assertion in the policy window (the name differs according to the assertion configuration) and select **Cookie Properties** or double-click the assertion in the policy window. The assertion properties are displayed.






The image shows a 'Cookie Properties' dialog box with a title bar containing a close button. The 'Operation' dropdown is set to 'Update'. Under the 'Set Attributes' section, there are input fields for Name, Value (containing 'newValue'), Version, Domain, Path, Max Age, Expires, and Comment. Each field has a checkbox to its right, all of which are checked and labeled 'original value'. Below this section are checkboxes for 'Secure' and 'HttpOnly', both of which are unchecked. The 'Match by' section has input fields for Name (containing 'Cookie1'), Domain, and Path, each with an unchecked checkbox labeled 'regular expression'. At the bottom are 'OK' and 'Cancel' buttons.

Figure 179: Cookie Properties, with "Update" operation example

3. Select a task to perform:

Table 149: Cookie tasks

To...	Do this...
Add a new cookie 	<ol style="list-style-type: none"> Choose the appropriate add operation: <ul style="list-style-type: none"> Add: Add a new cookie. Note: The assertion will fail if the cookie already exists. Add or Replace: Add a new cookie, replacing any existing cook with the same name, domain, and path. Enter as many attributes about the cookie as necessary. You may reference context variables in any of the attribute fields. <ul style="list-style-type: none"> Name and Value are required at a minimum. Version: Enter a value that specifies to which version the state management specification the cookie conforms. Domain: Enter a value that specifies the domain for which the

To...	Do this...
	<p>cookie is valid. Domains that are explicitly specified must always begin with a period.</p> <ul style="list-style-type: none"> • Path: Enter a value that specifies the subset of URLs to which this cookie applies. • Max Age: Enter the lifetime of the cookie, in seconds. When the Max Age is reached, the cookie is discarded by the client. A value of "0" (zero) indicates that the cookie will be discarded immediately, effectively ending the session. <p>Notes: (1) The agent may not retain the cookie for the specified duration; the cookie may be evicted due to memory pressure or privacy concerns. (2) Some agents do not support the Max Age attribute and will ignore it.</p> <ul style="list-style-type: none"> • Expires: Use the calendar control to choose an expiration date for the cookie. Tips: (1) If you choose an invalid date (for example, a date in the past), the date is highlighted in red but you will still be able to close the dialog box. (2) If you manually change the date, be sure to update the day of the week as well, otherwise the date will be flagged as erroneous. • Comment: Use this field to document the intended use of the cookie. <p>3. Optionally select these attributes:</p> <ul style="list-style-type: none"> • Secure: The cookie can only be used in secure/encrypted connections. • HttpOnly: The cookie is not exposed through channels other than HTTP or HTTPS requests. These cookies are not accessible via non-HTTP methods, such as calls via JavaScript.
<p>Remove cookies</p> 	<ol style="list-style-type: none"> 1. Choose the Remove operation. 2. Under "Match by", enter one or more attributes to locate the cookie(s) you wish to remove. You may also specify context variable or regular expression to delete several cookies at once. If regular expression is entered, select the regular expression check box.
<p>Update the attributes of existing cookies</p> 	<ol style="list-style-type: none"> 1. Choose the Update operation. 2. Clear the original value check box for any attribute that you wish to update, then enter the updated value in the adjacent text box. As with adding, you may reference context variables. Note: The "original value" check boxes mean to leave the original value of the attribute unchanged. 3. Under "Match by", specify one or more attributes to locate the cookie(s) you wish to update. You may also specify a context variable or regular expression to update multiple cookies at once. If a regular expression is entered, select the regular expression check box.

4. Click **[OK]** when done.

Manage Transport Properties/Headers Assertion

The *Manage Transport Properties/Headers* assertion allows you to add, replace, or remove custom HTTP header or JMS property in a message.

This assertion will always succeed provided that the target message exists.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Notes: (1) Use this assertion carefully. The Gateway does not validate the headers in a message. Conflicting or malformed headers may produce unexpected results. (2) If two or more JMS properties are added with the same name, only the last one added will be used by the [Route via JMS](#) assertion and the incoming JMS request listener.

Modifying Content-Type Headers

If you need to modify Content-Type headers, use the "Validate or Change Content Type Assertion" on page 694. Attempting to change these headers via the Manage Transport Properties/Headers assertion may result in duplicate Content-Type headers and incorrect interpretation of the message Content-Type.

Note that when removing Content-Type headers using this assertion, there is no effect on the actual Content-Type header coming out of the Gateway. Context variables (*http.allheadervalues*, *http.header/headervalues.Content-Type*) are affected as follows:

- For request messages, these context variables are removed.
- For response messages, these variables will remain.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the Manage Transport Properties/Headers Properties automatically appear; when modifying the assertion, right-click *<target>*: **[Add|Remove] [HTTP Header|JMS Property]...** in the policy window and select

Transport Properties/Headers Properties or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 180: Transport Properties/Headers Properties

3. Configure the dialog as follows:

Table 150: Transport Properties/Headers tasks



To...	Do this...
<p>Add a new HTTP header or JMS property</p> <p>(possibly duplicating an existing item with the same name)</p> 	<ol style="list-style-type: none"> 1. Choose the type of metadata to add: HTTP Header or JMS Property. 2. Choose the Add or Add or Replace operation: <ul style="list-style-type: none"> • "Add" may possibly duplicate an existing item with the same name. • "Add or Replace" replaces any existing header or property with the same name. 3. Enter the name of the header or property to add. You may reference context variables. 4. Optionally enter the value of the header or property. You may reference context variables. Tip: To use special characters, enclose them within double quotes. <p>A new HTTP header or JMS property with the specified information is added to the target message.</p>
<p>Remove one or more HTTP header or JMS property</p> 	<ol style="list-style-type: none"> 1. Choose the type of metadata to remove: HTTP Header or JMS Property. 2. Choose the Remove operation. 3. Specify the Header/Property Name to remove using any of the following methods (case insensitive unless a Regular Expression is specified): <ul style="list-style-type: none"> • Enter the exact name of the header or property to remove. • Specify a context variable that will resolve to the name of the header(s) or property(ies) to remove.

Table 150: Transport Properties/Headers tasks

To...	Do this...
	<ul style="list-style-type: none"> Specify a regular expression that matches the header(s) or property(ies) to remove, and then select the Regular Expression check box. <p>Tips: (1) To remove all headers names with a matching value, use a regular expression that matches all header names (for example, ".*"). (2) You cannot use context variables when the Regular Expression check box is selected. If you require this functionality, use the Evaluate Regular Expression assertion inside of a Run Assertions for Each Item assertion.</p> <p>4. Optionally, specify a corresponding Header Value using any of the following methods (case insensitive unless a Regular Expression is specified):</p> <ul style="list-style-type: none"> Enter the exact value to match. Specify a context variable that will resolve to the value to match. Specify a regular expression that matches the value(s) that you want to remove, and then select the Regular Expression check box. Note: The Policy Manager does not validate the regular expression pattern. <p>If you do not specify a value, then all matching HTTP header names or JMS properties are removed, regardless of value. However if you specify a context variable that resolves to empty, then only the matching items that have an empty value are removed.</p> <p>Note: The assertion does not fail if no header matches the given name and/or value. However, the assertion fails if the context variable for the header/property name does not exist or if it resolves to empty.</p>

4. Click **[OK]** when done.

Return Template Response to Requestor Assertion

The *Return Template Response to Requestor* assertion lets you define a message to be returned to the requestor. This allows you (for example) to create a more descriptive message for a SOAP fault or to elaborate an error condition to aid troubleshooting. For example, you place this assertion in an "[At least one assertion must evaluate to true](#)" assertion folder after an [Evaluate Response XPath](#) assertion. If the Evaluate Response XPath assertion fails, then the template response message will be sent back to the requestor.

Example:

A Return Template Response to Requestor assertion is configured as follows:

Response HTTP Status: 200
Response Content Type: text/plain
Response Body: Hello

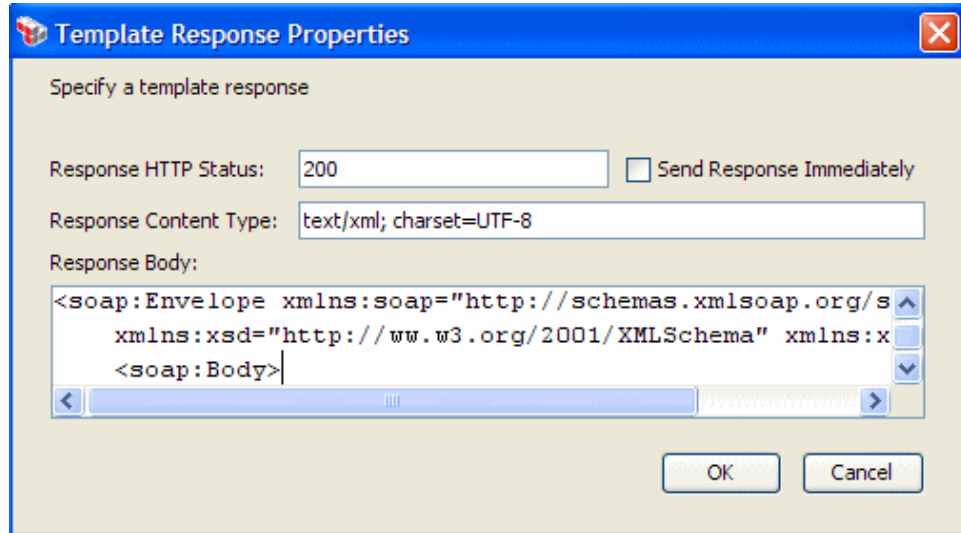
This will return the following message to the client:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain
Content-Length: 5
Date: Mon, 17 Apr 2006 17:38:28 GMT
Hello
```

Notes: (1) The template response is always sent at the end of the policy processing, regardless of its actual position within the policy. (2) Use of increasing number of context variables in a policy may impact the Return Template Response to Requestor assertion and cause an overall decrease in policy performance.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **Template Response Properties** automatically appear; when modifying the assertion, right-click **Return Template Response to Requestor** in the policy window and select **Template Response Properties** or double-click the assertion in the policy window. The assertion properties are displayed.



Specify a template response

Response HTTP Status: ☐ Send Response Immediately

Response Content Type:

Response Body:




```
<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>
```

OK Cancel

Figure 181: Template Response Properties

3. Configure the response as follows:

Table 151: Template Response settings

Setting	Description
Response HTTP Status 	Enter a valid numeric HTTP status code (for example, 200, 401, etc.). You may reference context variables.
Send Response immediately	Select this check box to send the template response message immediately, with or without a payload. The transmission of a payload depends on the status code (for example, a '204 (No Content)' response status indicates to skip payload transmission). Clear this check box to send the template response at the completion of the policy. Note: If the response is sent immediately, no WS-Security decoration is applied to the response message and the connection is closed (there is no persistent connections/keep-alive).
Response Content Type 	Enter any Content-Type in the format "first/last". A sample response Content-Type might be: <i>text/xml; charset=utf-8</i>
Response Body 	Type the message for the template response. You may include context variables within the message body, if necessary. However note the following if context variables are used: <ul style="list-style-type: none"> When a context variable is used in the template response, the Gateway does not check whether the XML response is well formed. This may result in invalid body content.

Setting	Description
	<ul style="list-style-type: none"> If the context variable is of type Message, you may need to refer to the <code>\${variableName.mainpart}</code> part of the context variable to avoid problems. Using ".mainpart" turns the contents into a String. <p>Note: For context variables of types other than Message, the <i>mainpart</i> part is not required. For more information about the data types, see Context Variables in the <i>Layer 7 Policy Manager User Manual</i>.</p>

- Click **[OK]** when done.

Route via FTP(S) Assertion

The *Route via FTP(S)* assertion is used to route requests from the Gateway to a backend FTP(S) server, using passive mode FTP. You can configure the port number to use and which directory to use on the remote FTP server.

Notes: (1) The Gateway can be configured as an FTP(S) server in order to support FTP proxying. For more information, see *Working with FTP Requests* in the *Layer 7 Policy Manager User Manual*. (2) The Gateway does not support the use of elliptic curve certificates (ECC) as the client certificate for an outbound TLS connection.

If the routing is successful, the response message will contain the reply from the remote FTP server. For uploads, the response body will be empty; for lists or download requests, the body will contain the listing/file contents. The reply code and message from the remote FTP server will be set in the target message and made available in these context variables:

```
${<prefix>.ftp.replycode}
${<prefix>.ftp.replytext}
```

Where "<prefix>" is:

- "response" if the Message Target is "Response"
- the name of the target message variable if the Message Target is "Message Variable"

The Message Target is set in the [Connection] tab of the FTP(S) Routing Properties.

Example: Requesting the last modified time of a file

This simple example shows how you can use the Route via FTP(S) assertion to retrieve the modified timestamp of a file on a remote server using a HTTP GET Request and how the reply code and message are populated into context variables.

Precondition:

- There is a remote FTP server "ftp.example.com" hosting the file `/log_files/log.txt`.

➤ *To request the modified timestamp from a remote server:*

1. Create the following simple policy fragment (properties settings to follow):

Route via FTPS Server ftp.example.com

[Return Template Response to Requestor](#)

2. Configure the [Connection] tab in the FTP(S) Routing Properties as follows:

- *Protocol:* **FTPS with Explicit SSL (AUTH TLS/SSL)**
- *Host name:* **ftp.example.com**
- Select **"Verify server certificate"**
- *Command:* **MDTM**
- *Message Target:* **Message Variable**
- *Target Message Variable:* **output**
- *Directory:* **/log_files**
- *Arguments:* **log.txt**
- *Assertion Outcome:* **Fail on Transient or Permanent Negative Completion
reply code**

3. Configure the Template Response Properties as follows:

- *Response HTTP Status:* **200**
- *Response Content Type:* **text/html; charset=UTF-8**
- *Response Body:*

```
<html><body>
RESPONSE: ${output.ftp.replycode} ${output.ftp.replytext}
</body></html>
```

This is the example in action:

1. The service policy receives and processes an HTTP GET request.
2. The Route via FTP(S) assertion retrieves the last modified date (MDTM command) of the file of specified file ('Directory' and 'Arguments') from the remote FTP server ('Host name').

3. The success reply code and last modified date of the file are made available through the context variables `${output.ftp.replycode}` and `${output.ftp.replytext}`.
4. The "Return Template Response to Requestor Assertion" on page 518 returns an HTML message with the reply details to the client; for example:

```
<html><body>
RESPONSE: 213 20140224231131.616
</body></html>
```

For a list of the supported FTP commands, see Table 5 in Listen Port Properties on page 1. For an example of how this assertion can be used with the listen ports to configure an extended FTP command support proxy, see Configuring the [FTP Settings] Tab on page 1.

FTP Cluster Properties for This Assertion

The cluster properties described in FTP Cluster PropertiesLayer 7 Policy Manager User Manual define the default FTP(S) listen port behavior. The following cluster properties are specific to the Route via FTP(S) assertion and have no effect on any listen ports.

Table 152: FTP Cluster Properties that only affect the Route via FTP(S) assertion

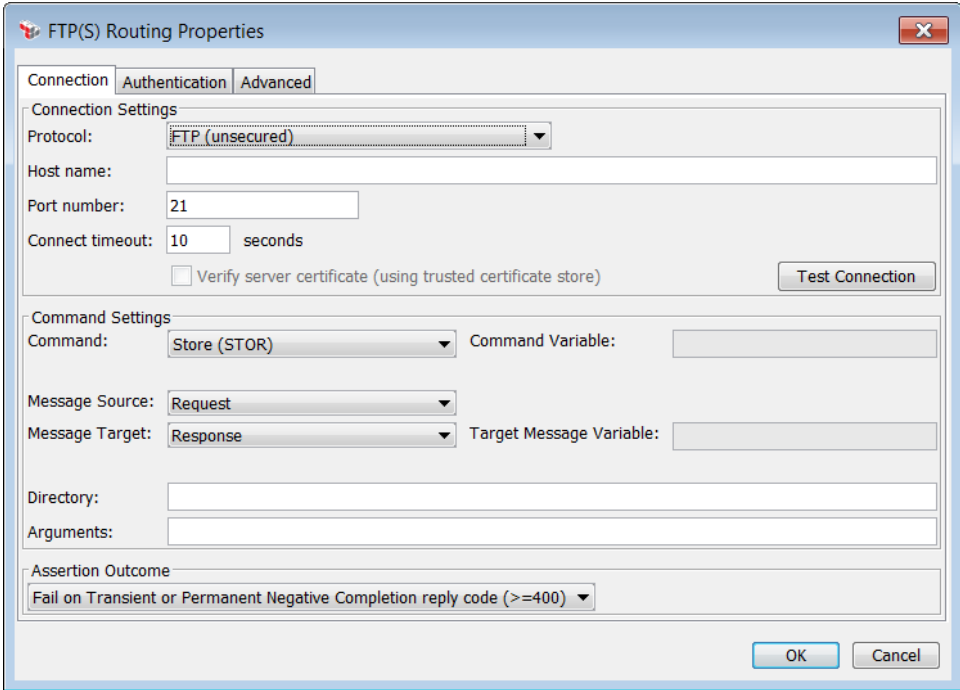
Property	Description
ftp.globalMaxDownloadConcurrency	The maximum number of FTP downloads that may be executed concurrently by Route via FTP(S) assertions. This is a global limit across all such assertions. Default: 64
ftp.globalMinDownloadConcurrency	The core number of FTP downloads that may be executed concurrently by Route via FTP(S) assertions. This is a global limit across all such assertions. Note: This is the number of threads in the pool under normal circumstances. Higher workloads will cause more threads to be created, up to the maximum defined by <code>ftp.globalMaxDownloadConcurrency</code> . Default: 32
ftp.globalMaxDownloadQueue	The maximum number of FTP downloads that may be waiting to execute concurrently by Route via FTP(S) assertions. This is a global limit across all such assertions. Default: 64

Using the Assertion

1. Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **FTP(S) Routing Properties** automatically appear; when modifying the assertion, right-click **Route via FTP(S) Server** in the policy window and select **FTP(S) Routing Properties** or double-click the assertion in the policy window. The assertion properties are displayed. These properties are organized across the following tabs:
 - Connection*
 - Authentication*
 - Advanced*
 3. Configure each tab as necessary. Refer to the appropriate section below for a complete description of each tab. **Tip:** If you are unsure of the settings to use, consult with the FTP server administrator.
 4. Click **[OK]** when done.

Configuring the [Connection] Tab



The screenshot shows the "FTP(S) Routing Properties" dialog box with the "Connection" tab selected. The dialog has three tabs: "Connection", "Authentication", and "Advanced". The "Connection" tab contains the following sections:

- Connection Settings:**
 - Protocol: **FTP (unsecured)** (dropdown menu)
 - Host name:
 - Port number:
 - Connect timeout: seconds
 - ☐ Verify server certificate (using trusted certificate store)
 - Test Connection** button
- Command Settings:**
 - Command: **Store (STOR)** (dropdown menu)
 - Command Variable:
 - Message Source: **Request** (dropdown menu)
 - Message Target: **Response** (dropdown menu)
 - Target Message Variable:
 - Directory:
 - Arguments:
- Assertion Outcome:**
 - Fail on Transient or Permanent Negative Completion reply code (>=400)** (dropdown menu)




At the bottom right are **OK** and **Cancel** buttons.

Figure 182: FTP(S) Routing Properties - [Connection] tab

The [Connection] tab is used to configure the FTP connection.

1. Configure the **Connection Settings** section as follows:





Table 153: FTP(S) connection settings

Setting	Description
Protocol 	Choose the protocol to use: <ul style="list-style-type: none"> FTP (unsecured): Information is submitted unencrypted. FTPS with Explicit SSL (AUTH TLS/SSL): Information is encrypted using explicit SSL (RFC2228). FTPS with Implicit SSL: Information is encrypted using implicit SSL.
Host name 	Enter the hostname of the FTP(S) server machine or a context variable that will contain the hostname. This name is verified against the X.509 certificate
Port number 	Specify the port number or a context variable to use for the security method chosen. These defaults are used: <ul style="list-style-type: none"> <i>FTP (unsecured)</i> and <i>FTPS with Explicit SSL</i>: port 21 <i>FTPS with Implicit SSL</i>: port 990
Connect timeout	Specify the connection timeout period. The default is 10 seconds.
Verify server certificate	If encryption is used, select this check box to verify the server's certificate against the trust store in the Gateway. For more information, see Managing Certificates in the <i>Layer 7 Policy Manager User Manual</i>
Test Connection	Click this button to test your FTP connection. This button is available only when all required information is entered in the properties dialog. Note: The [Test Connection] button cannot properly verify the connection if a context variable was used in the password, directory, or command.

2. Configure the Command Settings section as follows:

Table 154: FTP(S) command settings

Setting	Description
Command	Choose an FTP command to send to the FTP server, or choose From Variable to retrieve the command from the "Command Variable" field. For a list of the supported FTP commands, see Table 5 in Listen Port Properties on page 1. Note: The FTP command handling setting in the associated FTP listen port may impact how the command is interpreted. For more information, see Listen Port Properties in the <i>Layer 7 Policy Manager User Manual</i> .
Command Variable	If the raw FTP command is being retrieved from a context variable,

Setting	Description
	enter the variable here.
Message Source	From the drop-down list, select the source message of any data that is expected to accompany the selected command. Choose from Request , Response , or any Message context variables that have been defined so far.
Message Target	Specify the target message where reply data from the FTP server should be stored, including the reply code, reply text, and any file or list data. Choose from Response or any Message context variables that will receive this information.
Target Message Variable 	If the message target is a variable, enter the variable name. If this variable does not exist, it will be created.
Directory 	Specify the remote directory on the FTP server to use or a context variable that will contain the directory name. You will be warned if the context variable specified does not exist at that particular point in the policy. Leave blank to use the default root directory defined by the FTP server.
Arguments 	Enter any arguments or parameters required by the FTP command. For most common commands, the argument will be a file or directory name, but other commands may support other arguments. Note: The Arguments field replaces both the "Auto-generate file name" and "Specify pattern" options found in the Route via FTP(S) assertion prior to version 8.2.0. Examples of arguments: <ul style="list-style-type: none"> To ensure a unique file name for an upload request, you can use the context variable <code>\${requestID}</code> as the argument. This is the behavior of the "Auto-generate file name" option available in versions prior to 8.2.0. To define a specific custom file name, enter the file name here. For example if you are using an upload-type command, entering myFile.txt will cause a file named "myFile.txt" to be created on the remote FTP server that contains the contents of the message body. This is the same as selecting the "Specify pattern" option prior to version 8.2.0. <p>The arguments can be created dynamically by including context variables within the pattern. For example, the pattern <code>"fromGateway-\${requestID}"</code> will name all the uploaded files <code>"fromGateway-"</code> followed by the request ID. You will be warned if the context variable specified does not exist at that</p>

Setting	Description
	<p>particular point in the policy.</p> <ul style="list-style-type: none"> To list the contents of a specific directory (other than the current working directory), enter the name of that directory as the argument. For example, the Directory field contains "/users/jsmith". To list the contents of the "downloads" directory (assuming "/users/jsmith/downloads"), enter "downloads" in the Argument field.

- Indicate how the assertion will respond to error reply codes from the target FTP server. Choose the appropriate **Assertion Outcome** from the drop-down list:
 - Fail on all Permanent Negative Completion reply codes (result: fail on all codes ≥ 500)
 - Fail on Transient OR Permanent Negative Completion reply codes (result: fail on all codes ≥ 400)
 - Never fail as long as target replies. This option allows FTP clients in a proxy scenario to receive useful responses that will, in most cases, reveal the reasons for failures (for example, insufficient privileges, incorrectly formatted arguments, etc.).

For more details on the FTP reply codes, see https://en.wikipedia.org/wiki/List_of_FTP_server_return_codes.

Configuring the [Authentication] Tab

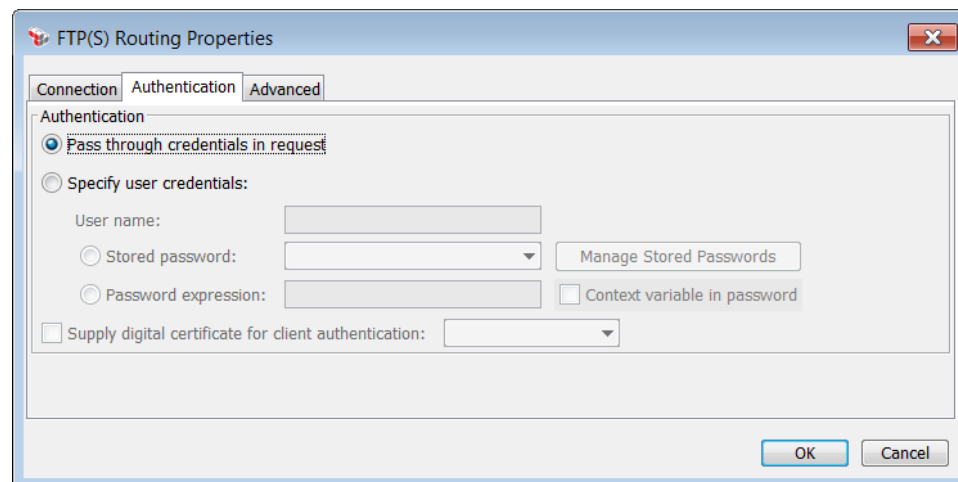


Figure 183: FTP(S) Routing Properties - [Authentication] tab

The [Authentication] tab is used to configure authentication to the remote FTP server.

1. Choose **Pass through credentials in request** to use the credentials contained in the request.

Choose **Specify user credentials** to manually enter the credentials to use, and then enter the user name and password:

- **User name**
- **Stored password:** Use a password from the secure password store on the Gateway. Choose the password to use from the drop-down list. **Note:** Only stored passwords may be used here—you cannot type in a password (to do this, use the "Password expression" option instead). To define a stored password, click [**Managed Stored Passwords**]. For more information, see Managing Stored Passwords in the *Layer 7 Policy Manager User Manual*.
- **Password expression:** Use the password in the specified expression. You may specify context variables or embed context variables within the expression. **Note:** Entering a password expression is not recommended, as it is stored in plaintext form and is less secure. For maximum security, use the "Stored password" option instead.
 - **Context variable in password:** Select this check box to allow the assertion to correctly recognize context variables used in the Password expression field; for example, you will be using the `${secpass.*}` context variables. For more information, see Stored Password Properties in the *Layer 7 Policy Manager User Manual*.

Tip: For security purposes, the user name and password are automatically deleted when you close the properties and "Specify" is not selected. You do not need to manually clear these fields.

2. **Supply digital certificate for client authentication:** Select this check box to use a client certificate from a private keystore for FTPS authentication, then choose the certificate from the dropdown list. **Note:** This option is available only if a private keystore has been defined.

Configuring the [Advanced] Tab

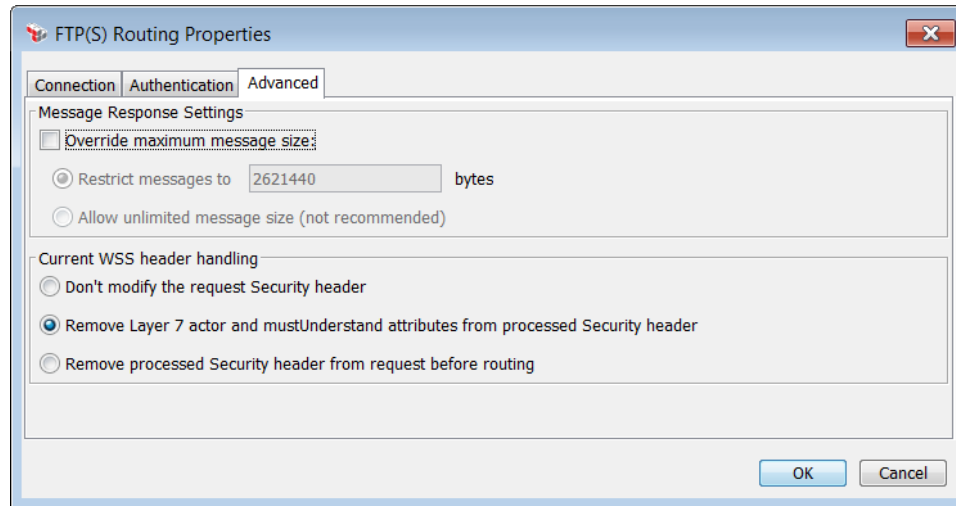


Figure 184: FTP(S) Routing Properties - [Advanced] tab

The [Advanced] tab is used to configure additional properties for the FTP routing.

- **Override maximum message size:** By default, the maximum response message size is 2GB. Select this check box to download messages larger than this limit or to impose a specific limit. You can also allow message of unlimited size, however this is not recommended as this may adversely affect Gateway performance.
- **Current WSS header handling:** Specify how to handle the security header.

Table 155: WSS header handling

Option	Description
Don't modify the request Security header	<p>Instructs the Gateway to leave the security header in the outgoing SOAP request message as-is. The security header in the request may still have been modified if the Gateway needed to decrypt any encrypted material during message processing.</p> <p>Use this setting if the protected service needs to do its own checking of the request's original security header, or if the protected service does not care whether its request messages have a security header.</p> <p>For best performance, use this setting whenever possible to minimize the amount of per-request message modification.</p> <p>Note: Do not modify the Security header if the policy uses WS-Security. For more information, see the "Add or Remove WS-Security Assertion" on page 273.</p>
Remove Layer 7 actor and mustUnderstand	<p>Instructs the Gateway to remove the 'mustUnderstand' attribute and 'Layer 7' actor from the security header in the outgoing SOAP message.</p>

Option	Description
attributes from processed Security header	<p>Use this setting if the presence of the Layer 7 actor causes issues with the backend service. In certain cases, this actor may cause the backend service to ignore the Security headers because it believes it is addressed to someone else. You will also use this setting if the backend service does not support Security and would reject a request with 'mustUnderstand' asserted on the Security header.</p> <p>An alternative might be to remove the Security header completely, however this will incur a performance penalty for the extra processing required to remove these from the messages. You may want to keep the Security headers intact for logging purposes.</p>
Remove processed Security header from request before routing	<p>Instructs the Gateway to remove any security header that was processed by the gateway before forwarding the request to the protected service.</p> <p>Use this setting when the protected service is not expecting security headers in the forwarded SOAP requests.</p>

Route via HTTP(S) Assertion

The *Route via HTTP(S)* assertion defines where a Web service or XML application message is sent and what authentication credentials it uses. If the service requests client authentication, the Gateway can be configured to respond in any number of ways:

- By default, it will use the subject certificate from the default SSL to respond to the SSL-TLS handshake.
- You can specify a custom private key to use. The Gateway will use the subject certificate from this private key to respond to outbound TLS client certificate challenges from the server.
- You can configure the Gateway to decline all certificate challenges by selecting the "Use no private key" option when selecting a private key in this assertion. **Tip:** This option is unique to the Route via HTTP(S) assertion.

To learn more about selecting a private key for this assertion, see [Selecting a Custom Private Key](#) in the *Layer 7 Policy Manager User Manual*.



A message routing assertion is an essential policy element. When you publish a service using the Publish SOAP Web Service Wizard, Create WSDL Wizard, or Publish Web API Wizard (with a target URL), the Policy Manager automatically adds the service URL specified during the publication process as a Route via HTTP(S) assertion in the published service's initial [policy](#).

Notes: (1) By default, the outbound HTTP method is passed through from the inbound request HTTP method. Where there is no inbound request method (for example, a context variable is used), then the POST action is used. (2) The Gateway does not support the use of elliptic curve certificates (ECC) as the client certificate for an outbound TLS connection.

The Route via HTTP(S) assertion supports the HTTP 1.0 and 1.1 standards. It should be present in policies that consume an external REST or HTTP-SOAP based API.


Using the Assertion

By default, the Policy Manager automatically adds a Route via HTTP(S) assertion to a new service policy created by one of the Publish Service wizards. If the assertion was removed or you need to add another one, refer to [Adding an Assertion](#) for instructions on adding this assertion.

1. Right-click "**Route via HTTP(S) to...**" in the policy window and then select **HTTP(S) Routing Properties** or double-click the assertion in the policy window. The assertion properties are displayed.
2. Review the address in the **URL** box to ensure that it is the correct URL for the service; make any changes if necessary. The Policy Manager will verify that the URL is well formed and that the hostname is valid in the DNS. **Tip:** For SOAP services published from a WSDL, you can click  to reset the URL to the one specified during the service publication process. 

Note: If the URL contains a valid host but invalid path, routing attempts will be recorded as a Policy Violation in the service statistics. However if the host is unknown, the routing attempts will be recorded as Routing Failures. For more information about service statistics, see [Dashboard - Service Metrics](#) in the *Layer 7 Policy Manager User Manual*.

For greater flexibility in specifying the path, you can embed context variables within the URL. Be sure the context variables resolve to a valid URL.

3. Choose the **HTTP Method** to use from the drop-down list. The list includes the well known methods, but you can enter your own method if necessary (including specifying a context variable). The default setting of **<Automatic>** uses the HTTP method from the request (if present), otherwise it uses the POST method. **Tip:** If a custom HTTP method is present in the message being routed, it will be passed through. 

For more information about HTTP Methods, refer to the "HTTP/FTP" tab of the Service Properties.

4. Choose the **Request Source** from the drop-down list. You may use the default request message or any other Message context variable that has been defined.
5. Specify the **Response Destination**. You may choose a destination from the drop-down list or type the name of a Message variable that will hold the response. **Tip:** The default variable name of "httpResponse" is just a suggestion; ensure this name is unique if you opt to use it. Refer to the context variable naming rules if you receive syntax errors.

Note: When saving the route response to a Message context variable, the response body and headers are saved to the variable, not the default response. The response returned back to the client is the default response, not the Message variable. The routing header rules should affect the headers saved to the Message variable in this case, not the headers returned to the client.

6. Configure each tab as necessary. Refer to the appropriate section below for a complete description of each tab.
7. Click **[OK]** when done.



Note: As of version 8.2.00, a *gzip* request or response with zero content-length will be treated the same as a non-gzipped request/response; the Route via HTTP(S) assertion no longer fails. As a result, previously "invalid" empty gzipped requests/responses that caused an error will no longer trigger an error, unless the policy is specifically constructed to fail if an empty gzip request/response is received.

Configuring the [Authentication] Tab

In the [Authentication] tab, select an authentication method.

Table 156: Route via HTTP(S): Authentication methods

Authentication	Description
None (Anonymous)	Select this option for anonymous services. No credentials are required.

Authentication	Description
Use OAuth Authorization 	<p>Select this option to use OAuth Authorization for credentials. Choose the OAuth Version and then specify the Token Variable.</p> <p>The OAuth version determines what is prepended to the contents of the token variable in the Authorization header value.</p> <ul style="list-style-type: none"> For OAuth 1.0, this is the equivalent of adding an Authorization header with the value "OAuth \${var}". Otherwise, it is equivalent to adding an Authorization header with the value "Bearer \${var}". <p>Note: Ensure that the OAuth token has already been obtained and is present in the specified context variable.</p>
Specify HTTP Credentials 	<p>Select this option for basic HTTP authentication. You are prompted to enter your User Name, Password, NTLM Domain, and NTLM Host. You may specify context variables in the User Name and Password fields.</p> <p>Notes: (1) If no credentials are entered, the authentication option will revert to "None (Anonymous)" the next time the [Authentication] tab is opened. (2) NTLM authentication is not supported when a proxy server is configured (see "Configuring the [Proxy] Tab" on page 539). (3) Both NTLM v1 and v2 authentication are supported.</p>
Use HTTP Credentials from Request	<p>Select this option to use the HTTP basic or NTLM authentication headers in the request.</p>
Attach SAML Sender-Vouches	<p>Select this option to attach a SAML sender-vouches ticket to each outgoing back-end request that was authenticated by the Gateway. This ticket contains the user name of the authenticated user along with an expiration time, and is signed by the Gateway using the SSL certificate. You are prompted to specify the SAML Version and Ticket expiry time, in minutes (whole number only).</p> <p>Note: The "Attach SAML Sender-Vouches" option is enabled only for SOAP Web service policies. It differs from the "Require SAML Token Profile Assertion" on page 228 as follows:</p> <ul style="list-style-type: none"> The Attach SAML Sender-Vouches option is being added to the outgoing message from the Gateway to the protected service. The Require SAML Token Profile Assertion requires that SAML security already be present in an incoming message from a client application to the Gateway.
Send TAI Header	<p>Select this option to require a Trust Association Interceptor (TAI) third-party authentication pass. TAI credential chaining can be used with or without a static user name and password. With TAI, if the Gateway authenticated a user, then the user name of that authenticated user will be included in the IV_USER HTTP header in the outgoing request.</p>

Authentication	Description
Use Windows Integrated	<p>Select this option to enable outbound Kerberos Delegation via Windows Integrated Authentication. Specify how to proceed:</p> <ul style="list-style-type: none"> • Use Delegated Credentials: Select this option to use the credentials extracted from the request Kerberos token to request a service ticket for routing. If using this option, one of the following assertions must be present in the policy: Require Windows Integrated Authentication Credentials or Require WS-Security Kerberos Token Profile Credentials. • Use Gateway Keytab: Select this option to use the <i>kerberos.keytab</i> file on the Gateway. For more information about this file, see "Using Windows Domain Login" in the <i>Layer 7 Installation and Maintenance Manual</i>. • Use Configured Credentials: Select this option to have the assertion use the specified account to authenticate with the KDC and obtain a service ticket for routing. <p>Notes: (1) The "Use Gateway Keytab" and "Use Configured Credentials" options do not require Kerberos access control (in other words, the Require Windows Integrated Authentication Credentials or WSS Kerberos assertions are not required). Using the Require HTTP Basic Credentials assertion is sufficient. (2) Kerberos authentication is not supported when a proxy server is configured (see "Configuring the [Proxy] Tab" on page 539).</p>

Configuring the [Headers] Tab

The [Headers] tab is used to define which HTTP headers should be passed through. It contains separate sections for request and response headers.

By default, all request and response headers are passed through in their original form.

IMPORTANT: There may be potential security ramifications to allowing all applications header to be passed through. If in doubt, restrict the pass-through to only specific headers.

When passing through only specific headers, define these headers in their respective tables. You can choose to pass the original value of the header or a custom value (context variables acceptable).

Some tips for constructing a list of headers to be passed through:

- You may repeat header names if you are constructing multiple rules on handling a particular header. See "[Working with Multiple Headers](#)" below for more details.

- When passing the original value, if the header is present multiple times in the incoming request, then it is passed multiple times as they are in the original request.
- When passing through only specific request or response headers, if no headers are specified in the accompanying table, then the Gateway will revert to passing through all headers.

Note that the following request headers are not automatically passed through even if you deselect the check box:

connection
content-encoding
content-length
content-type
date
keep-alive
server
transfer-encoding

Note: Header customizations made here may be overridden by changes made through the "Manage Transport Properties/Headers Assertion" on page 515 (or vice versa), depending on which assertion appears later in the policy.

Header Handling with Kerberos

Prior to version 8.2, an Authorization header set by the Add Header assertion (now renamed to "Manage Transport Properties/Headers Assertion") would override the Kerberos Authorization header set by the Route via HTTP(S) assertion. In version 8.2, the Kerberos Authorization header set by the Route via HTTP(S) assertion will not be overridden.

The practical implications of this change:

- Policies created prior to version 8.2 that are configured to pass through only specific headers are not affected by this change.
- Policies created in version 8.2 that have the "Pass through only certain request headers" check box selected are also not affected by this change.
- If you need to pass through all headers, you can now use the "Manage Transport Properties/Headers Assertion" on page 515 to remove unnecessary Authorization headers from the request.

Working with Multiple Headers

It may be necessary to construct multiple rules to describe how you want to handle a particular header. For example, you wish to forward several custom values for a particular request or response header.

The following table summarizes the possible scenarios when you are passing through only certain headers.

Table 157: Scenarios for header rules

Scenario	What will happen
Define a header "ABC" with value = <original value>	All headers with the name "ABC" will be forwarded, with their original values intact.
Define a header "ABC" with value = "XYZ"	A new header with name "ABC" with value "XYZ" is inserted.
Define the headers "ABC" = "123" and "ABC" = '456'.	Two headers with name "ABC" are inserted: One with value "123" and another one with value "456".
Define the headers "ABC" = <original value> and "ABC" = "123".	All headers with the name "ABC" will be forwarded, with their original values intact. An additional header with the custom value "123" is inserted.

Working with HTTP Host Headers

The HTTP Host Header can be set a number of different ways. By default, this header is set to the URL hostname specified at the top of the properties dialog. You can enhance the flexibility of the Host Header by doing the following:

1. Ensure you are passing through only certain request headers.
2. Add a new request header with the name **"Host"**.
3. Select **Customize value**, and then choose one of the following depending on how you wish to populate the Host Header:
 - *leave the value blank*: The HOST header in the HTTP request will be populated with the Host name from the target URL.
 - *enter a context variable*: The HOST header in the HTTP request will be populated with the value contained in the context variable.
 - *any other non-blank value*: The HOST header in the HTTP request will be populated with the value entered here.

Configuring the [Connection] Tab

The [Connection] tab is used to configure failover strategies, timeouts, and TLS settings.

1. Specify how IP addresses should be retrieved:

Table 158: Retrieving IP addresses during HTTP routing

IP address option	Description
Look Up IP Addresses in DNS	Select this option to have the Gateway retrieve the IP addresses from the Domain Name Server (DNS). This setting is the default and it does not use a failover strategy.
Use the following IP addresses	Select this option to have the Gateway only use IP addresses from the list that follows.
Use multiple URLs	Select this option to have the Gateway sequentially use URLs from the list that follows. This option is useful if, for example, multiple instances of a service reside at different URLs rather than just different IP addresses.


Tip: You may specify context variables when constructing a list of IP addresses or URLs.

2. Choose a **Failover strategy** to use in case an IP address or URL fails to respond:


Table 159: Failover Strategies during HTTP routing

Failover Strategy	Description
Ordered Sticky with Failover	<p>The Gateway sends service messages to the first IP/URL in the list until that IP/URL does not respond (fails). When this occurs, the next IP/URL in the list is used.</p> <p>Tip: The cluster property <code>io.failoverServerRetryDelay</code> controls the delay before the Gateway retries a failed server. The default is to wait 15 minutes when using the "Ordered Sticky with Failover" strategy.</p>
Random Sticky with Failover	<p>The Gateway chooses an IP/URL randomly at the beginning of each session and uses it for the duration of the session. If the chosen IP/URL address fails, another IP/URL is chosen at random.</p>
Round Robin	<p>The Gateway rotates through the IP/URL list on a request-by-request basis (round-robin) from the first, to the second, and so on. When the end of the list is reached, the cycle continues from the top of the list.</p> <p>Tip: The cluster property <code>io.failoverServerRetryDelay</code> controls the delay before the Gateway retries a failed server. The default is to wait 5 minutes when using the "Round Robin" strategy.</p>

3. If you wish to override any of the following timeout values for this routing assertion only, do the following:

- The **Connection Timeout** defines the maximum time (in milliseconds) the Gateway will attempt to establish a TCP connection. If exceeded, the routing will fail (or failover). To override the system default, clear the **Use System Default** check box and then enter a different value. You may reference context variables. 

Tip: The system default for this timeout is defined by the *io.outConnectTimeout* cluster property, which defaults to 30 seconds if the property is not explicitly set.

- The **Read Timeout** defines the maximum time (in milliseconds) allowed for response data (not necessarily the complete response) to be read for the outbound request. If exceeded, the routing will fail (or failover). To override the system default, clear the **Use System Default** check box and enter a value. You may reference context variables. 

Tip: The system default for this timeout is defined by the *io.outTimeout* cluster property, which defaults to 60 seconds if the property is not explicitly set.

- The **Maximum Retries** defines the maximum number of attempts, in addition to the initial attempt, to establish a TCP connection. For example, Maximum Retries = 3 means there will be 4 attempts: the initial attempt and 3 retry attempts. To override the system default, clear the **Use System Default** check box and enter a value between **0** and **100** (where "0" will prevent retries). The default is **3** retries.

4. Choose which **TLS Version** to allow when connecting via HTTPS.

To use a specific set of TLS cipher suites for this HTTP connection, click [**Cipher Suites**]. For more information, see *Selecting Cipher Suites in the Layer 7 Policy Manager User Manual*.

To allow a subset of trusted certificates during the outbound TLS handshake, click [**Trusted Server Certificates**] and then select:

- Trust all Trusted Certificate: Trust all trusted certificates presently in the Gateway trust store. For more information, see *Managing Certificates in the Layer 7 Policy Manager User Manual*.
- Trust only the specified Trusted Certificates: Trust only the trusted certificates in the table below. Only the certificates that you define here will be trusted during the outbound TLS handshake from this routing assertion

Note: As with all trusted certificates, the certificates in this list will be trusted only if their settings are compatible (for example, if it has been configured to be "trusted for outbound SSL").

Configuring the [HTTP] Tab

The [HTTP] tab is used to further refine the settings for your HTTP protocol.

1. Choose the **Version** from the drop-down list. The **Default** setting will use the version defined by the `io.httpVersion` cluster property. The other settings will override the cluster property.
2. Select the **Compress Output** check box if you want to compress the request payload. This can improve performance and transfer times, especially if the payloads are large.

Note: The compression option is valid only when the service endpoint is another Gateway.

3. Select the **Use Keep-Alive** check box to use persistent connections. These connections are more efficient, as they allow reuse of TCP connections for multiple messages. Clear this check box to not use persistent connections on this routing.

Note: You can enable "keep-alive" only when the `io.httpDisableKeepAlive` cluster property is at its default "false" setting. If that property was set to "true", then "keep-alives" are disabled globally and cannot be enabled for an individual HTTP routing.

4. Select the **Follow Redirects** check box to instruct the assertion to follow HTTP redirect responses from the downstream target. Otherwise, redirect responses are sent back to the requestor.
5. Select the **Transmit body regardless of request method** check box to include the request body with the outbound request, even if the HTTP request method is one that normally would not include a body (for example, GET, HEAD, DELETE, or OPTIONS). **Note:** The following of redirects is disabled for the request when a request body is forcibly included, even if the request method (such as GET) would otherwise have followed them

Clear the check box to not forcibly include the request body with the outbound request. In this case, the request body is include only with HTTP request methods (such as POST, PUT) that normally include them.

6. Click **Customize Request Form POST Parameters** if you need to change how these parameters work. You will be able to:
 - Specify whether all request form Post parameters received from the requestor are passed through, or only certain ones.

- Define a list of specific parameters to pass through. **Tips:** (1) You may repeat parameter names if you are constructing multiple rules on handling a particular parameter. See "Working with Multiple Headers " on page 535 for more details. (2) If the parameter is present multiple times in the incoming request, then it is passed multiple times as they are in the original request.

Configuring the [Proxy] Tab

The [Proxy] tab is used to configure an HTTP proxy host, if required.

When configuring a proxy host, enter the following:

Proxy Host
Proxy Port
Proxy Username
Proxy Password

Note: When a proxy server is configured, the following authentication methods cannot be used: *Specify HTTP Credentials, Use Windows Integrated.*

Configuring the [Other] Tab

The [Other] tab is used to configure miscellaneous HTTP routing settings.

1. In the **Request WSS Header Handling** section, specify how to handle the security header:

Table 160: WSS Header Handling during HTTP routing

Option	Description
Don't modify the request Security header	<p>Instructs the Gateway to leave the security header in the outgoing request message as-is. The security header in the request may still have been modified if the Gateway needed to decrypt any encrypted material during message processing.</p> <p>Use this setting if the protected service needs to do its own checking of the request's original security header, or if the protected service does not care whether its request messages have a security header.</p> <p>For best performance, use this setting whenever possible to minimize the amount of per-request message modification.</p> <p>Note: Do not modify the Security header if the policy uses WS-Security. For more information, see the "Add or Remove WS-Security Assertion" on page 273.</p>
Remove Layer 7 actor and mustUnderstand	<p>Instructs the Gateway to remove the "mustUnderstand" attribute and 'Layer 7' actor from the security header in the outgoing message.</p> <p>Use this setting if the presence of the Layer 7 actor causes issues with</p>

Option	Description
attributes from processed Security header	<p>the backend service. In certain cases, this actor may cause the backend service to ignore the Security headers because it believes it is addressed to someone else. You will also use this setting if the backend service does not support Security and would reject a request with "mustUnderstand" asserted on the Security header.</p> <p>An alternative might be to remove the Security header completely, however this will incur a performance penalty for the extra processing required to remove these from the messages. You may want to keep the Security headers intact for logging purposes.</p>
Remove processed Security header from request before routing	<p>Instructs the Gateway to remove any security header that was processed by the gateway before forwarding the request to the protected service.</p> <p>Use this setting when the protected service is not expecting security headers in the forwarded requests.</p>
Promote other security header as default before routing	<p>Instructs the Gateway to promote one of the downstream WSS recipients as the next default WSS header. Select the recipient from the drop-down list.</p> <p>This option is used primarily when the intended recipient of a WSS assertion does not accept or recognize security headers that contain Actor attributes.</p> <p>For more information about changing the recipient of the available WSS (WS-Security) message-level assertions, see "Changing the WSS Assertion Recipient" on page 146.</p>

- Under **Response Size Limit**, you may override the permitted maximum size of the routing message if necessary. By default, the maximum size is defined by the *io.xmlPartMaxBytes* cluster property. You may reference a context variable when restricting to a specific size. Note that allowing response messages of unlimited size is not recommended and should be used only under the direction of CA Technical Support.

Note: This setting only limits the size of the message when it is processed by the Gateway; to limit the size of the message sent to the client, use the [Limit Message Size](#) assertion. Also note that this setting (as well as the *io.xmlPartMaxBytes* cluster property) is not used when responses are streamed back to the client with no processing required by the Gateway. (Response streaming is controlled by the *io.HttpResponseStreaming* cluster property.)

- Choose an **Assertion Outcome** based on the response from the downstream endpoint:

Table 161: Assertion Outcome during HTTP routing

Option	Description
Fail if target returns error status (≥ 400)	<p>The assertion will fail when the response read from the target contains an error status (≥ 400).</p> <p>There is an exception to this rule: If the service for which the policy is associated with is published as a Web Service (not an XML application) and the target returns a response with the status 500 and the content is a SOAP fault, then the SOAP fault is accepted as a valid response to be propagated to the requestor.</p> <p>Note: If an error occurs while getting a response from the target, the assertion will fail.</p>
Pass through SOAP faults with error status 500	<p>When failing on an error status, you can give special attention to error status 500 (internal error).</p> <ul style="list-style-type: none"> Select this check box to allow a status 500 error to be returned from the backend server as a response, complete with the 500 HTTP error code. Clear this check box to treat the 500 error as a SOAP fault, which may trigger a customized fault response.
Never fail as long as target returns an answer	<p>The assertion will succeed if the endpoint returns any response read from the target. With this option, the assertion can fail only if it is not possible to read a response from the target.</p> <p>Exception: The assertion may still fail if both of the following conditions are true:</p> <ul style="list-style-type: none"> In the [Authentication] tab, the Service Authentication is "Use HTTP Credentials from Request". The backend services returns a 401 HTTP error. <p>To prevent assertion failure in this scenario, use "Specify HTTP Credentials" as the Service Authentication method instead. Specify the username and password as context variables from the request (i.e., <code>\${request.username}</code>, <code>\${request.password}</code>).</p>

Route via JMS Assertion

The *Route via JMS* assertion allows you to configure the JMS transportation of outbound service messages from the Gateway. In order to use this assertion in a policy, ensure that the JMS destinations have been:

1. Configured in the appropriate server (TIBCO EMS, IBM WebSphere over LDAP, or any other custom server).
2. Referenced in the JNDI directory.
3. Registered in the Policy Manager.

For more information, see Managing JMS Destinations in the *Layer 7 Policy Manager User Manual*.

Note: If multiple JMS properties with the same name exist in the message, only the last one added will be used by the Route via JMS assertion and the incoming request listener.

The Administrator is responsible for installing and configuring the items required for JMS routing. If you encounter errors during the execution of a JMS policy, contact your Administrator.

The Route via JMS assertion and destinations support the JMS 1.0 standard.

Context Variables Created by This Assertion

The Route via JMS assertion sets the following context variables with the header information from the JMS response message.

Note: The response context variables are not set if the JMS Destination is configured for “No replies (one-way)” (JMS Destination Properties > [Outbound Options] tab > Outbound Reply Behavior). For more information , see Managing JMS Destinations in the *Layer 7 Policy Manager User Manual*.

Table 162: Context variables created by the Route via JMS assertion

Variable	Description
<code>\${response.jms.header.<name>}</code>	Returns the value of the JMS response header, where <code><name></code> is the header name.
<code>\${response.jms.headernames}</code>	This is a multivalued context variable that returns the names of all headers that are present.
<code>\${response.jms.allheadervalues}</code>	<p>This is a multivalued context variable that returns all the header names and values that are present, in the format <code>headername:headervalue..</code></p> <p>The following are possible headers:</p> <ul style="list-style-type: none"> JMSDestination JMSDeliveryMode JMSExpiration JMSPriority JMSMessageID JMSTimestamp JMSCorrelationID JMSReplyTo JMSType JMSRedelivered

Tip: For a list of the context variables created when the Gateway receives a JMS request, see *Managing JMS Destinations in the Layer 7 Policy Manager User Manual*.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **JMS Routing Properties** automatically appear; when modifying the assertion, right-click **Route via JMS** in the policy window and select **JMS Routing Properties** or double-click the assertion in the policy window. The assertion properties are displayed. These properties are organized across the following tabs:
 - Target*
 - Security*
 - Request*
 - Response*
3. Configure each tab as necessary. Refer to the appropriate section below for a complete description of each tab.
4. Click **[OK]** when done.

Configuring the [Target] Tab

The screenshot shows the 'JMS Routing Properties' dialog box with the 'Target' tab selected. The 'JMS Destination' dropdown is set to 'jms destination [DestinationA on http://l7tech.com]'. A 'New Destination' button is to its right. Below is a 'Dynamic Properties' section with several text input fields: 'Initial Context Factory class name:', 'JNDI URL:', 'JNDI User Name:', 'JNDI Password:', 'Connection Factory Name:', 'Destination Name:', 'Destination User Name:', 'Destination Password:', and 'Wait for Reply on specified queue:'. There are 'Show Password' checkboxes next to the 'JNDI Password:' and 'Destination Password:' fields. At the bottom, there is a 'JMS response timeout:' field followed by the text 'milliseconds (empty for global default)'. 'OK' and 'Cancel' buttons are at the bottom right.

Figure 185: JMS Routing Properties - [Target] tab







The [Target] tab is used to select the JMS queue or topic, and configure dynamic properties and timeout.


1. Select the target outbound destination to use from the **JMS Destination** drop-down list. If the target destination you need doesn't exist, click [**New Destination**] to create a new JMS destination. See Managing JMS Destinations in the *Layer 7 Policy Manager User Manual* for information on defining this destination.
2. If you selected a template outbound destination in the previous step, complete the destination configuration in the **Dynamic Properties** section. Dynamic properties are those properties that are set at runtime, rather than at design time.

For more information about template destination, see "Template Outbound Destinations" in Managing JMS Destinations in the *Layer 7 Policy Manager User Manual*.

Tips: (1) If a value was entered during destination definition then it is displayed here, but cannot be modified. (2) If many destinations are in use, you can improve performance by reducing the idle time and increasing the cache size. These are controlled using the *io.jmsConnectionCacheMaxIdleTime* and *io.jmsConnectionCacheMaxSize* cluster properties, respectively.

Table 163: Dynamic properties for template outbound destinations

Dynamic Property	For information on this property, see...
Initial Context Factory class name	JMS Destination Properties, [JNDI] tab
JNDI URL	JMS Destination Properties, [JNDI] tab
JNDI User Name 	JMS Destination Properties, [JNDI] tab Tip: You may reference a context variable for the JNDI User Name in JMS Routing Properties.
JNDI Password 	JMS Destination Properties, [JNDI] tab Tip: You may reference a context variable for the JNDI Password in JMS Routing Properties.
Connection Factory Name 	JMS Destination Properties, [Destination] tab Tip: You may reference a context variable for the Connection Factory Name in JMS Routing Properties.
Destination Name 	JMS Destination Properties, [Destination] tab Tip: You may reference a context variable for the Destination Name in JMS Routing Properties.
Destination User Name 	JMS Destination Properties, [Destination] tab Tip: You may reference a context variable for the Destination User Name in JMS Routing Properties.
Destination Password 	JMS Destination Properties, [Destination] tab Tip: You may reference a context variable for the Destination Password in JMS Routing Properties.
Wait for Reply on specified queue	JMS Destination Properties, [Outbound Options] tab Note: If the Outbound Reply Behavior in the template queue is not "Wait for Reply", then this field is blank and uneditable.

- Optionally enter a **JMS response timeout** value if you wish to override the global default (defined in the *jms.ResponseTimeout* cluster property) for this one destination. The value must be greater than **0** (zero). Enter a value in milliseconds or enter a context variable that will contain the timeout value. The assertion will wait for this period of time for a response before timing out. 

Tip: The global default is specified by the *jms.ResponseTimeout* cluster property.

IMPORTANT: Ensure that the JMS response timeout is not greater than the HTTP response timeout on the client, otherwise data may be lost. (The client response timeout may vary, but for web browsers it is usually two minutes.)

Configuring the [Security] Tab

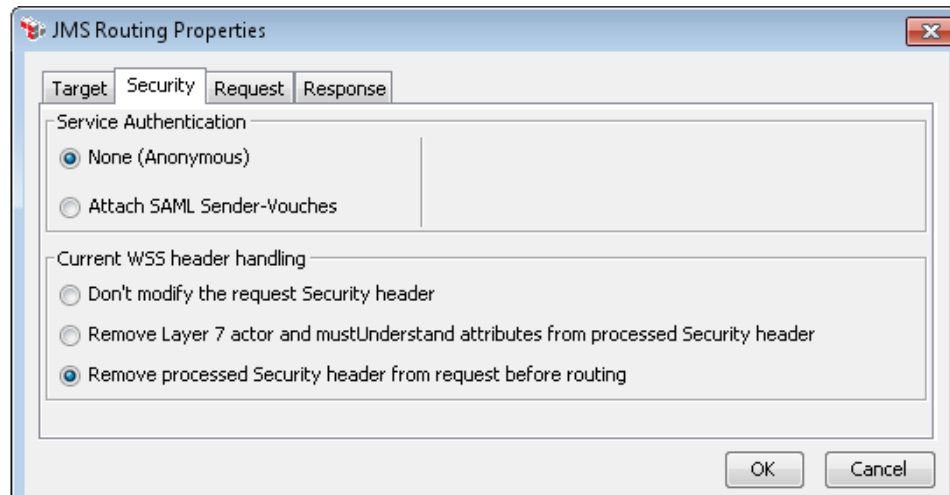


Figure 186: JMS Routing Properties - [Security] tab

The [Security] tab is used to set the service authentication and WSS header handling.

1. In the **Service Authentication** section, indicate whether authentication should be used:

Table 164: Service Authentication during JMS routing

Option	Description
None (Anonymous)	Select this option if the identity of the requestor is not being authenticated (requests anonymous).
Attach SAML Sender-Vouches	<p>Select this option to attach a SAML sender-vouches ticket to each outgoing back-end request that was authenticated by the Gateway. This ticket contains the user name of the authenticated user along with an expiration time, and is signed by the Gateway using the SSL certificate.</p> <p>When using SAML Sender-Vouches, indicate:</p> <ul style="list-style-type: none"> • SAML Version: Specify whether SAML 1.1 or 2.0 is being used. • Ticket expiry: Enter the expiry period for the ticket, in number of minutes (whole number only). The default is 5 minutes. <p>Note: This option is enabled only for SOAP web service policies. It</p>

Option	Description
	<p>differs from the Require SAML Token Profile assertion as follows:</p> <ul style="list-style-type: none"> The Attach SAML Sender-Vouches option is being added to the outgoing message from the Gateway to the protected service. The Require SAML Token Profile Assertion requires that SAML security already be present in an incoming message from a client application to the Gateway.

2. In the **Current WSS header handling** section, specify how to handle the security header:

Table 165: WSS Header Handling during JMS routing

Option	Description
Don't modify the request Security header	<p>Instructs the Gateway to leave the security header in the outgoing SOAP request message as-is. The security header in the request may still have been modified if the Gateway needed to decrypt any encrypted material during message processing.</p> <p>Use this setting if the protected service needs to do its own checking of the request's original security header, or if the protected service does not care whether its request messages have a security header.</p> <p>For best performance, use this setting whenever possible to minimize the amount of per-request message modification.</p> <p>Note: Do not modify the Security header if the policy uses WS-Security. For more information, see the "Add or Remove WS-Security Assertion" on page 273.</p>
Remove Layer 7 actor and mustUnderstand attributes from processed Security header	<p>Instructs the Gateway to remove the "mustUnderstand" attribute and "Layer 7" actor from the security header in the outgoing SOAP message.</p> <p>Use this setting if the presence of the Layer 7 actor causes issues with the backend service. In certain cases, this actor may cause the backend service to ignore the Security headers because it believes it is addressed to someone else. You will also use this setting if the backend service does not support Security and would reject a request with "mustUnderstand" asserted on the Security header.</p> <p>An alternative might be to remove the Security header completely, however this will incur a performance penalty for the extra processing required to remove these from the messages. You may want to keep the Security headers intact for logging purposes.</p>
Remove processed Security header from request before routing	<p>Instructs the Gateway to remove any security header that was processed by the gateway before forwarding the request to the protected service.</p> <p>Use this setting when the protected service is not expecting security headers in the forwarded SOAP requests.</p>

Configuring the [Request] Tab

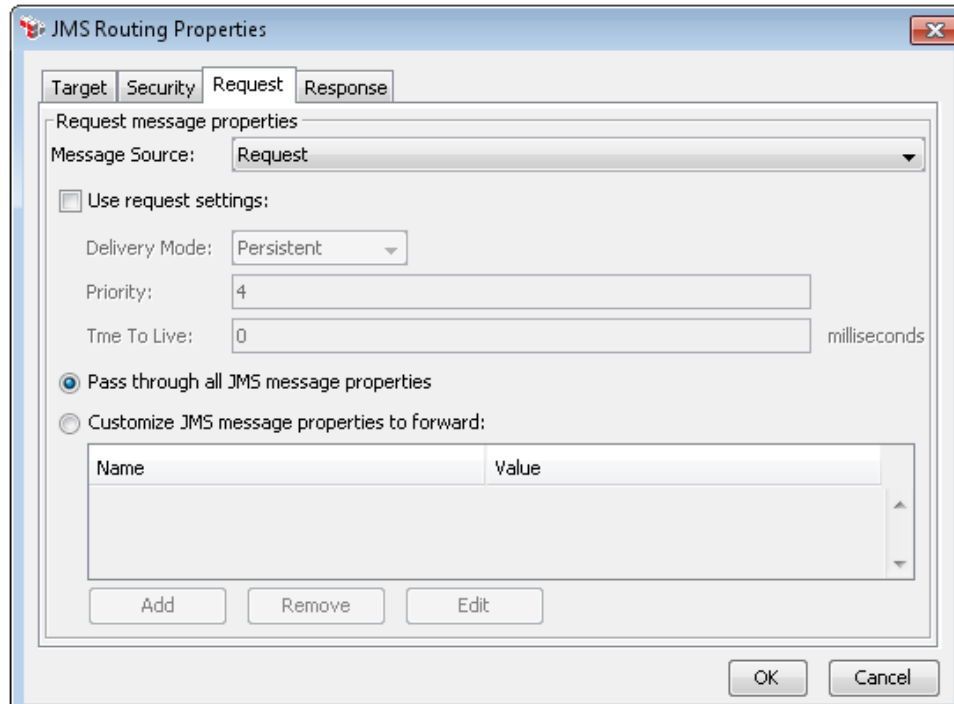




Figure 187: JMS Routing Properties - [Request] tab

The [Request] tab is used to select the message source for the request message and to configure property forwarding.


1. Specify the **Message Source** for the request message: from the drop-down list, choose from **Request**, **Response**, or any Message context variables that have been defined so far. These variables may have been created by the [Set Context Variable](#) assertion or in the **Response message properties** section of a previous Route via JMS assertion.
2. Select the **Use request settings** check box if you need to change any of the following default JMS request settings:
 - **Delivery Mode:** Choose the JMS delivery mode to use: **Persistent** or **Non-Persistent**.
 - **Priority:** Specify a priority mode for the JMS request by entering a value between **0** and **9**, where "0" is lowest priority and "9" is highest priority. The default is **4**. You may reference context variables. 

Time To Live: Enter the length of time before the JMS message expires, in milliseconds. The default value of **0** (zero) means the message never expires. You may reference context variables. 

3. Indicate how the properties from the JMS request message are handled:
 - Select **Pass through all JMS message properties** if you are allowing all JMS message properties to pass through. (Note that there will be JMS message properties to pass through only if the original request is a JMS message.)
 - Select **Customize JMS message properties to forward** if you want to do any of the following:
 - customize which or how JMS message properties are passed through
 - customize the values of the JMS message properties
 - create properties that did not exist in the original request (such as when the original request is not a JMS message).

If you are customizing the JMS message properties to pass through, define which names and values can pass through in the table below.

Table 166: Defining the JMS properties for forwarding

To...	Do this
Add a property to the list 	<ol style="list-style-type: none"> 1. Click [Add]. The JMS Message Property Setting dialog appears. 2. Enter the Property Name. (Note: Property names must obey the rules specified in the JMS Specification). 3. Specify what to do for this property: <ul style="list-style-type: none"> • Pass through original value: If the property is present in the incoming request, then pass it downstream as is. • Customize value: Insert a property with a custom string value. Enter the custom value in the adjacent box. You can either enter a fixed value or a string that contains context variables that resolves to the appropriate value during run time. For example, <i>"Hello, my ID is \${requestId} and the time is \${gateway.time}. Have a nice day."</i> <p>Note: The Policy Manager checks that the custom value entered is appropriate for the specified property. However note that if a context variable is specified, it is not possible to validate the data type at design time and an incorrect data type will cause an error during runtime. The following data types are enforced for each property:</p> <p style="margin-left: 40px;"> <i>JMSXUserID:</i> String <i>JMSXAppID:</i> String <i>JMSXDeliveryCount:</i> int </p>


To...	Do this
	<p> <i>JMSXGroupID</i>: String <i>JMSXGroupSeq</i>: int <i>JMSXProducerTXID</i>: String <i>JMSXConsumerTXID</i>: String <i>JMSXRcvTimestamp</i>: long <i>JMSXState</i>: int </p> <p>4. Click [OK]. The new property is added to the table.</p>
Modify a property in the list	<p>1. Select the property to modify and then click [Edit], or double-click a row. The JMS Message Property Setting dialog appears.</p> <p>2. Modify the information as necessary. See "Add a property to the list" above for details.</p> <p>3. Click [OK]. The modified property appears on the table.</p>
Remove a property from the list	<p>1. Select the row to delete.</p> <p>2. Click [Remove]. The property is removed immediately.</p>

Configuring the [Response] Tab

The screenshot shows the 'JMS Routing Properties' dialog box with the 'Response' tab selected. The 'Response message properties' section is expanded, showing 'Message Destination:' with two radio buttons: 'Default Response' (selected) and 'Save as context variable:' (with an empty text field). Below this is a checkbox for 'Override maximum message size:' which is unchecked. Under this checkbox are two radio buttons: 'Restrict messages to' (selected) with a text field containing '2621440' and the unit 'bytes', and 'Allow unlimited message size (not recommended)'. Below these are two more radio buttons: 'Pass through all JMS message properties' (selected) and 'Customize JMS message properties to forward:' (with an empty table). The table has two columns: 'Name' and 'Value'. At the bottom of the table are three buttons: 'Add', 'Remove', and 'Edit'. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

Figure 188: JMS Routing Properties - [Response] tab

The **[Response]** tab is used to configure the message destination and property forwarding.

1. Select the **Message Destination**:
 - Select **Default Response** to send the message to the default response destination.
 - Select **Save as context variable** to save the response message to a context variable that you specify here. You can define a new variable or an existing one. A validation message provides instant feedback on the context variable name entered. For an explanation of the validation messages displayed, see Context Variable Validation in the *Layer 7 Policy Manager User Manual*.
2. Specify the response message size limit:
 - **Override maximum message size**: Select this check box to override the permitted maximum size of the routing message. Clear this check box to use the value set in the `io.jmsMessageMaxBytes` cluster property.
 - *Restrict messages to*: Enter the maximum permitted size of the response message, in bytes. You may specify a context variable. 
 - *Allow unlimited message size (not recommended)*: Select this option to allow response messages of unlimited size. This is not recommended and should be used only under the direction of CA Technical Support.
3. Indicate how the properties from the JMS response message are handled:
 - **Pass through all JMS message properties**
 - **Customize JMS message properties to forward**

Please see "[Configuring the \[Request\] Tab](#)" above for the descriptions of these settings.

Route via MQ Native Assertion

The *Route via MQ Native* assertion allows you to configure the MQ Native transportation of outbound service messages from the CA API Gateway.

Note: MQ Native will not be operational until the appropriate .jar files have been installed.

Context Variables Created by This Assertion

The Route via MQ Native assertion sets the following context variables. **Note:** The `<prefix>` is the target of the message. For example, if the message target is a request, then the prefix is "request"; if the message target is a response, then the prefix is "response".

Table 167: Context variables created by the Route via MQ Native assertion

Context Variable	Description
<code>\${<prefix>.mqnative.md.<field>}</code>	<p>Returns the value of the case sensitive message descriptor field.</p> <p>Example: <code>\${request.mqnative.md.expiry}</code></p> <p>Note: The value for byte[] type descriptors must be entered as a Base64-encoded string. The date must be in the format <code>yyyy-MM-dd-HH.mm.ss.SSSSSS</code>.</p>
<code>\${<prefix>.mqnative.additionalheader.<folder name>.<name>}</code>	<p>Returns the value of the message header. Specifying the folder name is optional.</p> <p>Example: <code>\${request.mqnative.additionalheader.folder.rfh2Field2}</code></p>
<code>\${<prefix>.mqnative.property.<name>}</code>	<p>Returns the value of the message property.</p> <p>Example: <code>\${request.mqnative.property.folder.testStringProperty}</code></p>
<code>\${<prefix>.mqnative.additionalheadernames}</code>	<p>A multivalued context variable that retrieves all message header names. The suffix <code>.length</code> can be applied.</p> <p>Example: <code>\${request.mqnative.additionalheadernames}</code></p>
<code>\${<prefix>.mqnative.alladditionalheadervalue s}</code>	<p>A multivalued context variable that retrieves all message header values. The suffix <code>.length</code> can be applied.</p> <p>Example: <code>\${request.mqnative.alladditionalheadervalue s}</code></p>
<code>\${<prefix>.mqnative.propertynames}</code>	<p>A multivalued context variable that retrieves all message property names. The suffix <code>.length</code> can be applied.</p> <p>Example: <code>\${request.mqnative.propertynames}</code></p>
<code>\${<prefix>.mqnative.allpropertyvalues}</code>	<p>A multi-valued context variable that retrieves all message property values. The suffix <code>.length</code> can be applied.</p> <p>Example: <code>\${request.mqnative.allpropertyvalues}</code></p>

Defined MQ Header Prefixes

Header name prefixes for MQ Message Descriptor, Property, and Header are listed in [Table 2](#).

Table 168: Defined MQ Headers Prefixes

Header Name Prefix	Description
mqnative.md	This is the header name prefix for MQ Message Descriptor. Example: <i>mqnative.md.expiry</i>
mqnative.property	This is the header name prefix for MQ Message Property. Example: <i>mqnative.property.folder.propertyname</i>
mqnative.additionalheader	This is the header name prefix for MQ Message Header. Example: <i>mqnative.header.folder.headername</i>

To customize the Message Descriptors, Properties, and Additional Headers, use the "Manage Transport Properties/Headers Assertion" on page 515. Leave the Header Value empty to remove an attribute from the MQ Message Descriptor, MQ Message Property and MQ Additional Header.

Note: If a header contains the wrong prefix, it will be ignored.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the **Native MQ Routing Properties** automatically appear; when modifying the assertion, right-click **Route via MQ Native** in the policy window and select **MQ Native Routing Properties** or double-click the assertion in the policy window. The assertion properties are displayed. These properties are organized across the following tabs:

Target

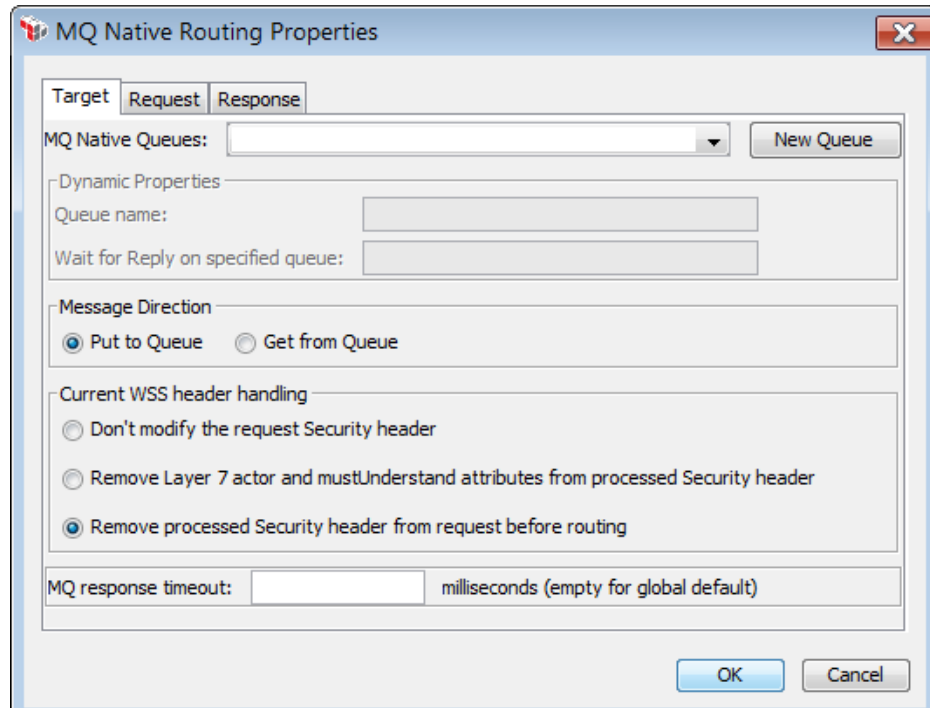
Request

Response

- Configure each tab as necessary. Refer to the appropriate section below for a complete description of each tab.

- Click **[OK]** when done.

Configuring the [Target] Tab




The image shows a dialog box titled "MQ Native Routing Properties" with a close button (X) in the top right corner. It has three tabs: "Target", "Request", and "Response", with "Target" currently selected. The "Target" tab contains the following elements:

- A dropdown menu labeled "MQ Native Queues:" with a "New Queue" button to its right.
- A section titled "Dynamic Properties" containing:
 - A text field labeled "Queue name:".
 - A text field labeled "Wait for Reply on specified queue:".
- A section titled "Message Direction" with two radio buttons:
 - ☒ Put to Queue
 - ☐ Get from Queue
- A section titled "Current WSS header handling" with three radio buttons:
 - ☐ Don't modify the request Security header
 - ☐ Remove Layer 7 actor and mustUnderstand attributes from processed Security header
 - ☒ Remove processed Security header from request before routing
- A text field labeled "MQ response timeout:" followed by the text "milliseconds (empty for global default)".
- "OK" and "Cancel" buttons at the bottom right.

Figure 189: MQ Native Routing Properties - [Target] tab

The [Target] tab is used to select the MQ Native Queue to use.

- Choose the connection to use from the **MQ Native Queues** drop-down list. If the connection you need doesn't exist, click **[New Queue]** to create a new connection. See Managing MQ Native Queues in the *Layer 7 Policy Manager User Manual* for more information on defining the queue.
- If a template outbound queue was chosen as the **MQ Native Queue** in the previous step, complete the **Dynamic Properties** section. Dynamic properties are those properties that are set at runtime, rather than at design time.
 - Queue Name:** Enter a dynamic queue name. You may reference a context variable. 

The dynamic queue name setting is enabled for outbound queues that have been designated as a template, and the queue name has been left empty. For details, see "Configuring the MQ Connections Properties" under Managing MQ Native Queues in the *Layer 7 Policy Manager User Manual*.

- **Wait for Reply on specified queue:** Enter a dynamic reply queue name. You may reference a context variable. 

The dynamic reply queue name is enabled for outbound queues that have been designated as a template, has the "Wait for Reply on specified queue" option selected, and the reply queue name is empty. For details, see "Configuring the MQ Connections Properties" and "Configuring the Outbound Options" under Managing MQ Native Queues in the *Layer 7 Policy Manager User Manual*.

3. Specify the direction for the message: **Put to Queue** (default) or **Get from Queue**.

Note: To specify the queue name dynamically, be sure to enter a context variable for the **Queue Name** under "[Dynamic Properties](#)" above.

4. In the **Current WSS header handling** section, specify how to handle the security header:

Table 169: WSS Header Handling during MQ Native routing

Option	Description
Don't modify the request Security header	<p>Instructs the Gateway to leave the security header in the outgoing SOAP request message as-is. The security header in the request may still have been modified if the Gateway needed to decrypt any encrypted material during message processing.</p> <p>Use this setting if the protected service needs to do its own checking of the request's original security header, or if the protected service does not care whether its request messages have a security header.</p> <p>For best performance, use this setting whenever possible to minimize the amount of per-request message modification.</p> <p>Note: Do not modify the Security header if the policy uses WS-Security. For more information, see the "Add or Remove WS-Security Assertion" on page 273.</p>
Remove Layer 7 actor and mustUnderstand attributes from processed Security header	<p>Instructs the Gateway to remove the "mustUnderstand" attribute and "Layer 7" actor from the security header in the outgoing SOAP message.</p> <p>Use this setting if the presence of the Layer 7 actor causes issues with the back end service. In certain cases, this actor may cause the back end service to ignore the Security headers because it believes it is addressed to someone else. You will also use this setting if the back end service does not support Security and would reject a request with "mustUnderstand" asserted on the Security header.</p> <p>An alternative might be to remove the Security header completely, however this will incur a performance penalty for the extra processing required to remove these from the messages. You may want to keep the Security headers intact for logging purposes.</p>

Option	Description
Remove processed Security header from request before routing	<p>Instructs the Gateway to remove any security header that was processed by the gateway before forwarding the request to the protected service.</p> <p>Use this setting when the protected service is not expecting security headers in the forwarded SOAP requests.</p>

5. Enter an **MQ response timeout** value if you wish to override the global default (defined in the *io.mqResponseTimeout* cluster property) for this one queue. The value must be greater than **0** (zero). Enter a value in milliseconds. The assertion will wait for this period of time for a response before timing out.

Configuring the [Request] Tab

Figure 190: MQ Native Routing Properties - [Request] tab

The **[Request]** tab is used to select the message source and to configure any MQ messages properties that may be required. This tab is available only when the message direction in the **[Target]** tab is **Put to Queue**.

1. Choose the message source from the drop-down list: **Request** (default) or **Response**.
2. Configure the properties of the [**Request**] tab as follows.

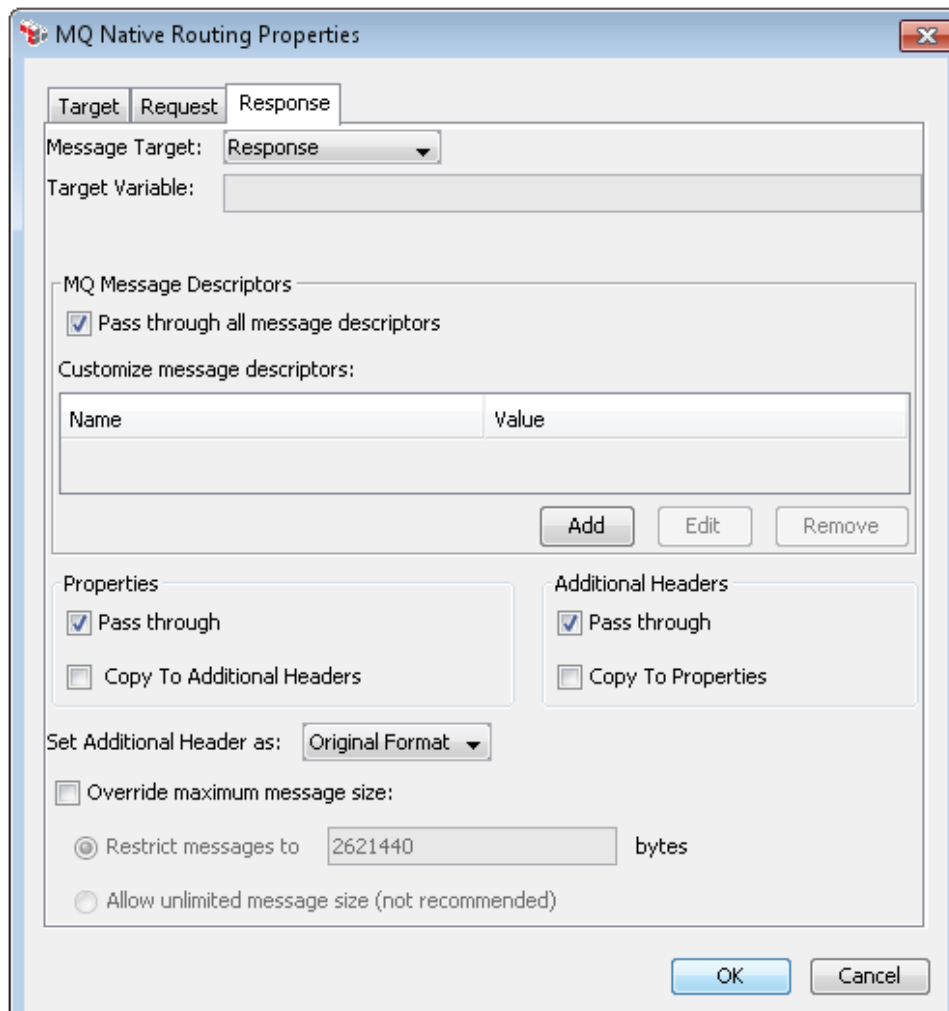
Table 170: MQ Native Route Properties - [Request]

Setting	Description
<i>MQ Message Descriptors</i>	
Pass through MQ message descriptors	<p>Select this check box to allow all MQ descriptors in the source message to pass through. (Note that there will be MQ message properties to pass through only if the original request is an MQ message.)</p> <p>Clear this check box to pass through the default values of the descriptor to the result message.</p>
Customize message descriptors	<p>To customize message descriptors that existed prior to version 8.0, see Customizing MQ Messages in the <i>Layer 7 Policy Manager User Manual</i>. For message descriptors created in version 8.0, it is recommended to use the "Manage Transport Properties/Headers Assertion" on page 515 to add customized values.</p>
<i>Properties</i>	
Pass through	<p>Select this check box to pass all message properties from the message source.</p> <p>Clear this box if you do not want the MQ message properties to pass through.</p>
Copy to Additional Headers	<p>Select this check box to copy the MQ Message properties to the message additional headers.</p> <p>Clear this check box to not perform the copy action.</p>
<i>Additional Headers</i>	
Pass through	<p>Select this check box to pass all MQ additional headers from the message source.</p> <p>Clear this box if you do not want the customized MQ additional headers to pass through.</p>
Copy to Properties	<p>Select this check box to copy the additional header name-value pair to the message properties.</p> <p>Clear this check box to not perform the copy action.</p>
Set Additional Header as	<p>Choose the additional header format from the drop-down list:</p> <ul style="list-style-type: none"> • Original: This option retains the original primary header format (MQRFH or MQRFH2). The header is automatically populated with the values from the original primary header. • MQRFH: All additional headers in the message are replaced with a MQRFH header. The MQRFH header is automatically

Table 170: MQ Native Route Properties - [Request]

Setting	Description
	<p>populated with the values from the original primary header.</p> <ul style="list-style-type: none"> • MQRFH2: The primary additional header is replaced with a MQRFH2 header in the message. The MQRFH2 header is automatically populated with the values from the original primary header.

Configuring the [Response] Tab



The image shows the 'MQ Native Routing Properties' dialog box with the 'Response' tab selected. The 'Message Target' is set to 'Response'. The 'Target Variable' field is empty. Under 'MQ Message Descriptors', the 'Pass through all message descriptors' checkbox is checked. Below this is a table for customizing message descriptors with columns 'Name' and 'Value'. There are 'Add', 'Edit', and 'Remove' buttons. In the 'Properties' section, 'Pass through' is checked and 'Copy To Additional Headers' is unchecked. In the 'Additional Headers' section, 'Pass through' is checked and 'Copy To Properties' is unchecked. The 'Set Additional Header as:' dropdown is set to 'Original Format'. The 'Override maximum message size:' checkbox is unchecked. The 'Restrict messages to' radio button is selected, with a value of '2621440' bytes. The 'Allow unlimited message size (not recommended)' radio button is unselected. 'OK' and 'Cancel' buttons are at the bottom right.

Figure 191: MQ Native Routing Properties - [Response] tab

The **[Response]** tab is used to configure the response message properties and to configure any advanced properties that may be required.

1. Configure the properties of the **[Response]** tab as follows.

Table 171: MQ Native Route Properties - [Response]

Setting	Description
Message Target	Choose the message target from the drop-down list: Request , Response (default), or a Message Variable .
Target Variable	If the target is a context variable, specify the Message variable in the Target Variable field. You can define a new variable or use an existing one. A validation message provides instant feedback on the context variable name entered. For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i> .
<i>MQ Message Descriptors</i>	
Pass through all message descriptors	<p>Select this check box to allow all MQ descriptors in the source message to pass through. (Note that there will be MQ message properties to pass through only if the original request is an MQ message.)</p> <p>Clear this check box to pass through the default values of the descriptor to the result message.</p>
Customize message descriptors	To customize message descriptors that existed prior to version 8.0, see Customizing MQ Messages in the <i>Layer 7 Policy Manager User Manual</i> . For message descriptors created in version 8.0, it is recommended to use the "Manage Transport Properties/Headers Assertion" on page 515 to add customized values to its properties and headers.
<i>Properties</i>	
Pass through	<p>Select this check box to pass all message properties from the message source.</p> <p>Clear this box if you do not want the customized message properties to pass through.</p>
Copy to Additional Headers	<p>Select this check box to copy the primary header name-value pair to the message headers.</p> <p>Clear this check box to not perform the copy action.</p>
<i>Additional Headers</i>	
Pass through	<p>Select this check box to pass all message properties from the message source.</p> <p>Clear this box if you do not want the customized message properties to pass through.</p>
Copy to Properties	<p>Select this check box to copy the primary header name-value pair to the message properties.</p> <p>Clear this check box to not perform the copy action.</p>
Set Additional	Choose the additional header from the drop-down list:

Table 171: MQ Native Route Properties - [Response]

Setting	Description
Header as	<ul style="list-style-type: none"> • Original: This option retains the original primary header format in MQRFH or MQRFH2. The header is automatically populated with the values from the original primary header. • MQRFH: All additional headers in the message are replaced with a MQRFH header. The MQRFH header is automatically populated with the values from the original primary header. • MQRFH2: This primary additional header is replaced with a MQRFH2 header in the message. The MQRFH2 header is automatically populated with the values from the original primary header.
Override maximum message size	<p>Select this check box to override the permitted maximum size of the message. Clear this check box to use the value set in the <i>io.mqMessageMaxBytes</i> cluster property.</p> <ul style="list-style-type: none"> • Restrict messages to: Enter the maximum permitted size of the message, in bytes. • Allow unlimited message size (not recommended): Select this option to allow response messages of unlimited size. Note: This is not recommended and should be used only under the direction of CA Technical Support.

Route via Raw TCP Assertion

The *Route via Raw TCP* assertion is used if the custom transport protocol "**l7.raw.tcp**" has been configured for a listen port. This assertion acts as a client of the server-side transport: it will transmit the request, close the sending side, read the response (if possible), and then initialize the response message with a pre-configured Content-Type.

This assertion will succeed if the raw TCP routing is successful.

Note: The *Route via Raw TCP* assertion can also be used as part of dynamic routing. See *Working with Dynamic Routing* for more information.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.

- When adding the assertion, the **Raw TCP Routing Properties** automatically appear; when modifying the assertion, right-click **Route via Raw TCP** in the policy window and select **Raw TCP Routing Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

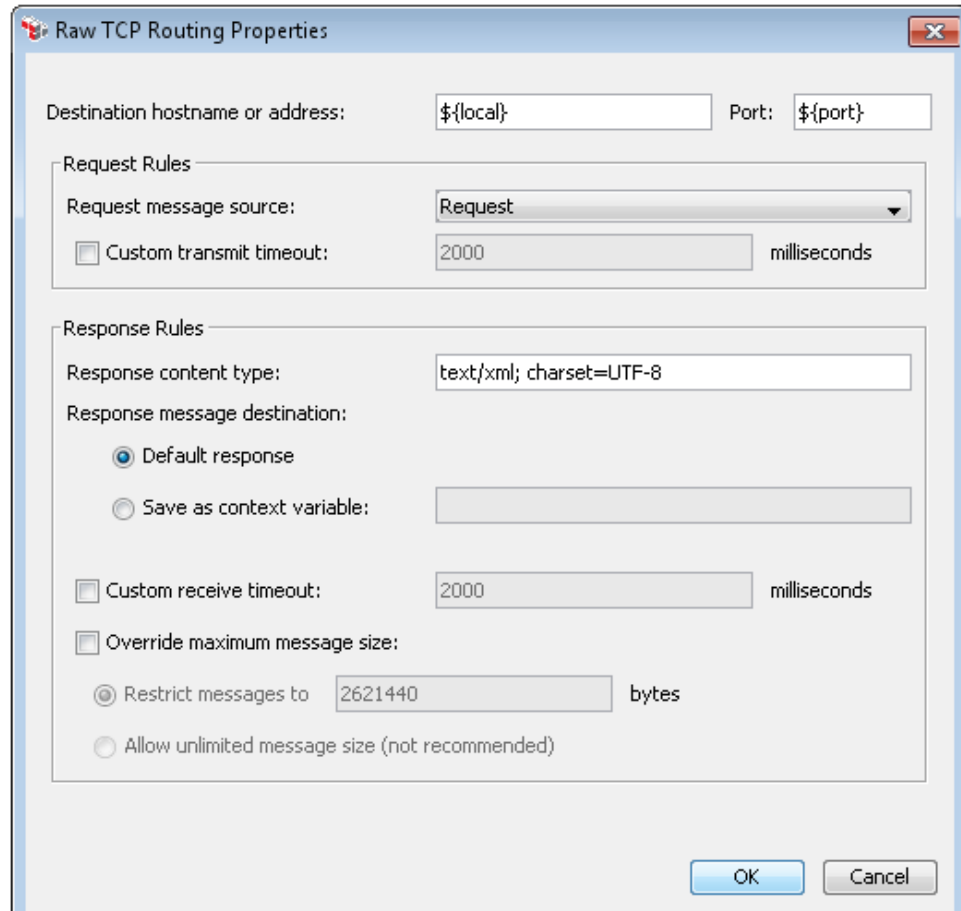








Figure 192: Raw TCP Routing Properties

- Configure the properties as follows.

Table 172: Raw TCP Routing settings

Setting	Description
Destination hostname or address 	Enter the hostname or IP address to which the Gateway should connect to send the request message.
Port 	Enter the port number to use. Alternatively, enter a context variable.

Setting	Description
Request message source 	Use the drop-down list to select the source for the request message: <ul style="list-style-type: none"> from the default Request from a context variable of type Message (this variable must already be defined in the policy before it will appear in the list)
Custom transmit timeout	The amount of time the Gateway should wait for acknowledgment of writes to the server before giving up (also known as <i>socket write timeout</i>). <ul style="list-style-type: none"> Select the check box to enable the timeout and then enter the timeout period, in milliseconds. The default is 2000 milliseconds. Clear the check box to not use a timeout. The Gateway will wait indefinitely. <p>Note: This is just the socket timeout, so it applies per read or write operation. It does not apply to the entire transaction.</p>
Response content type 	Enter the MIME Content-Type to assume for the response (for example, "text/xml").
Response message destination	Indicate where to store the response: <ul style="list-style-type: none"> Default response: Store the response in the default response. Save as context variable: Store the response in the specified Message context variable. If this variable does not already exist, it will be created. 
Custom receive timeout	How long the Gateway should wait for reads from the server to result in additional data before giving up (also known as <i>socket read timeout</i>). <ul style="list-style-type: none"> Select the check box to enable the timeout and then enter the timeout period, in milliseconds. The default is 2000 milliseconds. Clear the check box to not use a timeout. The Gateway will wait indefinitely. <p>Note: This is just the socket timeout, so it applies per read or write operation. It does not apply to the entire transaction.</p>
Override maximum message size	Select this check box to override the permitted maximum size of the routing message. Clear this check box to use the value set in the <i>io.xmlPartMaxBytes</i> cluster property. <ul style="list-style-type: none"> <i>Restrict messages to:</i> Enter the maximum permitted size of the response message, in bytes. You may specify a context variable. 

Setting	Description
	<ul style="list-style-type: none"> <i>Allow unlimited message size (not recommended)</i>: Select this option to allow response messages of unlimited size. This is not recommended and should be used only under the direction of CA Technical Support.

- Click **[OK]** when done.

Route via SSH2 Assertion

The *Route via SSH2* assertion is used to secure requests from the Gateway to backend services to provide SCP and SFTP (SSH File Transfer Protocol) support for the Gateway, so that SCP and SFTP outbound requests can be made to an external SSH server.

To view or modify the list of enabled ciphers for SSH2 routing, see the *ssh.routingEnabledCiphers* cluster property.

This assertion supports SSH2 only; SSH1 is not supported.

Performing SFTP Partial Downloads/Uploads

Note: This section is intended for advanced users who need to deal with a specific use case involving SFTP and partial downloads/uploads. This procedure should not be required as part of a normal workflow.

Specific policy configuration is necessary when attempting to download or upload a file from a listen port that has been configured to use partial upload/uploads for SFTP GET/PUT. Ensure the following policy fragment (or equivalent) is present in the SFTP policy:



- 4  Request: Configure Message Streaming: enable streaming
- 5  Route via SFTP to Server 10.7.49.176 use sftp to put the file on a different server.

Figure 193: Sample policy for SFTP partial downloads

The following table describes each line in the policy fragment in more detail.

Table 173: Assertions in SFTP partial downloads fragment

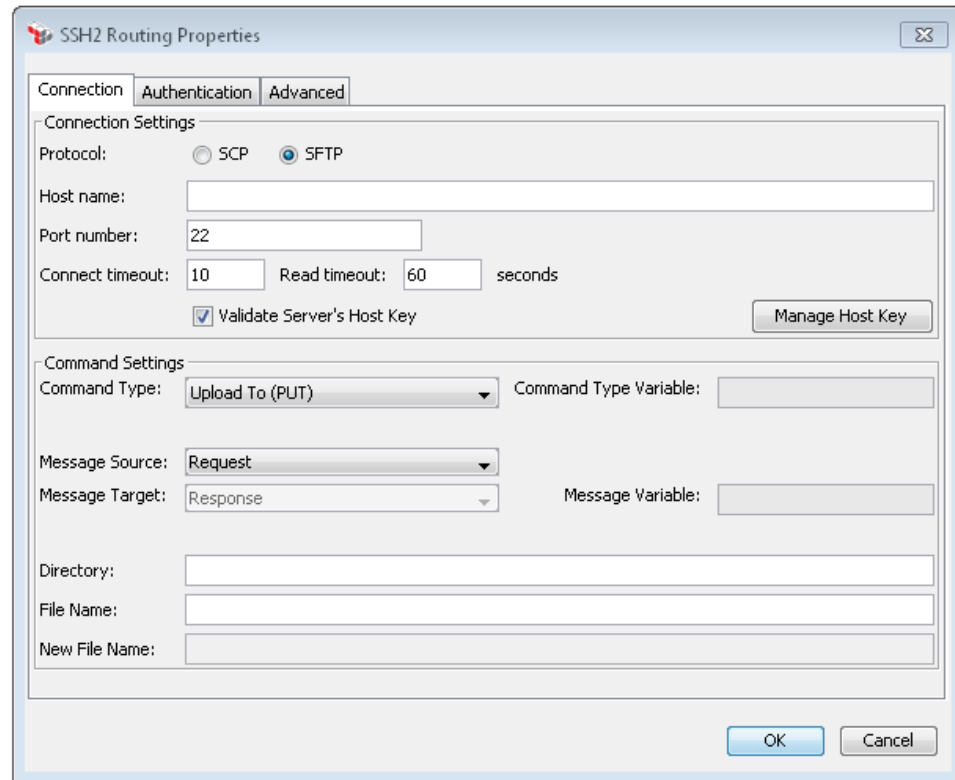
Line	Description
4	Insert the "Configure Message Streaming Assertion" on page 508. Ensure the target is "Request" and that "Enable streaming (no buffering)" is selected.
5	Insert the Route via SSH2 assertion and configure it as follows: [Connection] Tab 1. Choose SFTP for the protocol.

Table 173: Assertions in SFTP partial downloads fragment

Line	Description
	<ol style="list-style-type: none"> Enter the Host name and Port number of the SFTP server. Choose From Variable as the command type and then enter request.command.type as the variable. Enter \${request.ssh.path} as the directory name. Enter \${request.ssh.file} as the file name. Enter \${request.command.parameter.newPath} / \${request.command.parameter.newFile} as the new file name. <p>[Authentication] Tab</p> <p>Specify the credentials for the SFTP server.</p> <p>[Advanced] Tab</p> <ol style="list-style-type: none"> Select the Set File Size to Context Variable check box and enter ssh.file.size as the context variable. Select the Override maximum message size check box and choose [Allow unlimited message size]. Enter the following context variables in these fields: <ul style="list-style-type: none"> File Offset: \${request.command.parameter.offset} File Length: \${request.command.parameter.length}

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the **SSH2 Routing Properties** automatically appear; when modifying the assertion, right-click **Route via SSH2** in the policy window and select **SSH2 Routing Properties** or double-click the assertion in the policy window. The assertion properties are displayed.



The image shows the 'SSH2 Routing Properties' dialog box with the 'Connection' tab selected. The 'Connection Settings' section includes a 'Protocol' dropdown set to 'SFTP', a 'Host name' text field, a 'Port number' dropdown set to '22', and 'Connect timeout' (10) and 'Read timeout' (60) fields followed by 'seconds'. There is a checked 'Validate Server's Host Key' checkbox and a 'Manage Host Key' button. The 'Command Settings' section includes a 'Command Type' dropdown set to 'Upload To (PUT)', a 'Command Type Variable' text field, a 'Message Source' dropdown set to 'Request', a 'Message Target' dropdown set to 'Response', and a 'Message Variable' text field. At the bottom are 'Directory', 'File Name', and 'New File Name' text fields. 'OK' and 'Cancel' buttons are at the bottom right.

Figure 194: SSH2 Routing Properties - [Connection] tab

- Configure the properties of the **[Connection]** tab as follows. If you are unsure of the settings to use, consult with the SSH server administrator.

Table 174: SSH2 Routing Settings [Connection] tab



Setting	Description
<i>Connection settings</i>	
Protocol	Select the protocol to use. Choose from SCP or SFTP .
Host name 	Enter the hostname of the remote server machine or a context variable that will contain the hostname. This name is verified against the X.509 certificate.
Port number 	Specify the port number or a context variable to use for the security method chosen. This default port number is 22 .
Connect timeout	Specify the number of seconds before the SSH connection times out.
Read Timeout	Specify the SSH read timeout in seconds. This timeout applies to downloads only.
Validate Server's	Select this check box to validate the server's SSH public key against a

Table 174: SSH2 Routing Settings [Connection] tab




Setting	Description
Host Key	<p>fingerprint that you will specify using the [Manage Host Key] button. This setting is the default.</p> <p>Clear this check box to not validate the server's host key.</p>
Manage Host Key	<p>This button is available only when you are validating the server's host key. It is used to enter the fingerprint against which the host key is validated. Complete the following:</p> <ul style="list-style-type: none"> • SSH Public Key Fingerprint: Paste the SSH public key fingerprint as retrieved from the remote server's public key location. • Load From File: Click this to load the fingerprint from a text file. <p>About Host Key Fingerprints</p> <p>The fingerprint should be a colon-delimited series of two-digit hexadecimal values; for example: <i>b9:ac:0c:3d:bb:07:a8:a3:cc:eb:d7:f8:c4:89:b1:27</i>. This fingerprint can be determined using the <i>ssh-keygen</i> command on the server:</p> <pre># ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key 2048 b9:ac:0c:3d:bb:07:a8:a3:cc: eb:d7:f8:c4:89:b1:27 /etc/ssh/ssh_host_rsa_key.pub #</pre> <p>Additionally, when most SSH clients (for example, ssh, PuTTY) first connect to a server, the server's fingerprint will be displayed.</p>
<i>Command Selection</i>	
Command Type	<p>Choose the command type from the drop-down list:</p> <p>From Variable</p> <p>Upload To (PUT)</p> <p>Download From (GET)</p> <p>Get Directory Listing (LIST)</p> <p>Get File Attributes (STAT)</p> <p>Delete File (DELETE)</p> <p>Move File (MOVE)</p> <p>Create Directory (MKDIR)</p> <p>Delete Directory (RMDIR)</p> <p>For more information on each type, see "Description of Command Types" in Table 175.</p>
Command Type Variable 	<p>Alternatively, you can specify a context variable instead of choosing a command type from the drop-down list..</p> <p>Note: Ensure that the variable resolves to one of the command types</p>

Table 174: SSH2 Routing Settings [Connection] tab

Setting	Description
	<p>shown above, otherwise the assertion will fail.</p> <p>Tip: When using an SFTP inbound listener, use a value such as <code>\${request.command.type}</code></p>
Message Source	<p>Choose the source message to upload from the drop-down list: Request, or Response. This field is available only when you select Upload To as the Command Type in the [Connection] tab.</p>
Message Target 	<p>Choose the message target from the drop-down list: Request, Response, or Message Variable, (which will be created if it does not exist). This field is available only when you select Download From as the Command Type in the [Connection] tab.</p>
Message Variable 	<p>If you are targeting the message to a variable, enter the context variable in this field.</p>
Directory	<p>Specify the directory to upload to or download from. The user must have read permission on the directory to download and write permission to upload.</p> <p>The following is an example of a directory for an SFTP inbound listener.</p> <p><code>/my/root/directory\${request.ssh.path}</code></p> <p>Note: When uploading, the specified directory must exist already. This assertion will not create a directory.</p>
File name	<p>Enter the name of the file. The user must have read permission to download or write permission to upload.</p> <p>The following is an example of a file name.</p> <p><code>\${request.ssh.file}</code></p>
New File Name	<p>Enter the new file name of the file. This option is only applicable to SFTP MOVE.</p> <p>The following is an example of a new file name (all one line):</p> <p><code>/my/root/dir\${request.command.parameter.newPath}</code> <code>/\${request.command.parameter.newFile}</code></p>

Descriptions of Command Types

Table 175: Command Types for the Command Selection drop down list

Setting	Description
From Variable	<p>Select this option to allow the command type to be determined from the specified context variable. If you select this option, you must specify a valid context variable in the "Command Type Variable" field.</p>
Upload to (PUT)	<p>Select this option to do the following.</p> <ul style="list-style-type: none"> Fail if File Exists

Setting	Description
	<p>The upload will fail if the file already exists.</p> <ul style="list-style-type: none"> Overwrite <p>Uncheck this if partial file uploads are allowed. If there is an existing file with the same name, the file will be overwritten or truncated <i>before</i> the upload starts.</p> <ul style="list-style-type: none"> File Offset <p>Specify the file offset point from which to start writing the file. Default: 0</p>
Download from (GET)	<p>Select this option to do the following.</p> <ul style="list-style-type: none"> File Offset <p>Specify the file offset point from which to start retrieving the file data. Default: 0</p> <ul style="list-style-type: none"> File Length <p>Specify the length of the file data to retrieve. If the EOF ("end of file") is reached, the returned length can be less than the input length.</p> <p>The default will be reading the entire file until EOF, which is -1.</p>
Get Directory Listing (LIST)	<p>Select this option to retrieve the directory listing of the folder specified. It will return the list in an xml format.</p>
Get File Attributes (STAT)	<p>Select this option to retrieve the file attributes for the file specified, by using the file name and directory. It will return a single file element, or, if the file does not exist, it will return 0.</p>
Delete File (DELETE)	<p>Select this option to delete the file by specifying the file name and directory.</p>
Move File (MOVE)	<p>Select this option to move the file or directory to the new location specified. A new file name must be specified, which can either be a relative file name or absolute file path.</p>
Create Directory (MKDIR)	<p>Select this option to create a directory to the location specified.</p>
Delete Directory (RMDIR)	<p>Select this option to delete a directory from the location specified.</p>

4. Select the **[Authentication]** tab.

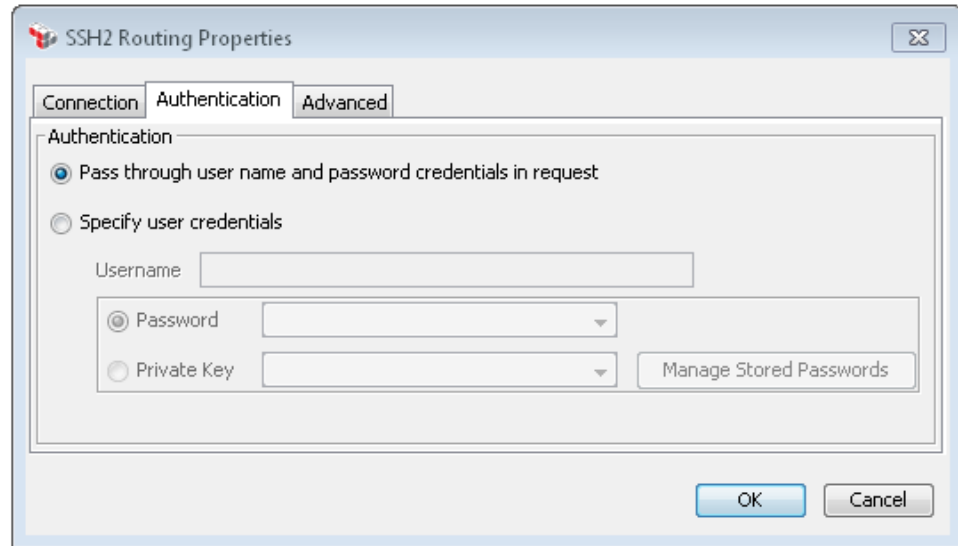


Figure 195: SSH2 Routing Properties - [Authentication] tab

5. Configure the properties of the **[Authentication]** tab as follows. If you are unsure of the settings to use, consult with the SSH server administrator.

Table 176: SSH2 Routing Settings [Authentication] tab

Setting	Description
Authentication	<p>Select the protocol to use:</p> <ul style="list-style-type: none"> • Pass through user name and password credentials in the request: Select this option to use the credentials already present in the request. • Specify user credentials: Select this option to enter specific credentials in the fields below.
Username	Enter the user name to connect to the server.
Password	<p>If authenticating via password, choose the password from the drop-down list.</p> <p>If the password you require is not listed, click [Manage Stored Passwords] to add it to the Gateway's password storage. For more information, see Managing Stored Passwords in the <i>Layer 7 Policy Manager User Manual</i>.</p> <p>Tip: You cannot type the password directly here; it must be defined in the Gateway's secure password storage.</p>
Private Key	<p>If authenticating via private key, choose the key to use.</p> <p>If the private key you require is not listed, click [Manage Stored Passwords] to add it to the Gateway's password storage. For more information, see Managing Stored Passwords in the <i>Layer 7 Policy Manager User Manual</i>.</p>

6. Select the **[Advanced]** tab.

SSH2 Routing Properties

Connection Authentication **Advanced**

Advanced Command Settings

Content type: text/xml; charset=utf-8

File Offset: 0

File Length: -1

☐ Set File Size to Context Variable:

☐ Preserve File Mode (Permission) ☐ Fail if File Exists

☐ Override maximum message size:

☒ Restrict messages to 2621440 bytes

☐ Allow unlimited message size (not recommended)

Current WSS header handling

☐ Don't modify the request Security header

☐ Remove Layer 7 actor and mustUnderstand attributes from processed Security header

☒ Remove processed Security header from request before routing


OK Cancel

Figure 196: SSH2 Routing Properties - [Advanced] tab

7. Configure the properties of the **[Advanced]** tab as follows. If you are unsure of the settings to use, consult with the SSH server administrator.

Table 177: SSH2 Routing Settings [Advanced] tab

Setting	Description
<i>Advanced Command Settings</i>	
Content type	Specify the MIME Content-Type header. This header only applies to downloads, and the content type is used for files being uploaded to the Gateway.
File Offset	<p>Enter a number that represents the point from which to start reading or writing the file. Specifying a number other than "0" allows for partial reads or writes. This option applies for only SFTP GET/PUT.</p> <p>Default: 0</p> <p>Note: If the file offset is "0", the file will automatically be truncated (emptied) before being written to. For values other than "0", the original file contents will be overwritten without truncation</p>

Setting	Description
	occurring.
File Length	<p>Enter the length of the file being uploaded, if for SCP PUT. If it is for SFTP GET/PUT enter the length of the file to download or upload.</p> <p>If the file length is -1, the entire input stream will be read and stashed in order to calculate the length if it is needed. It is only needed for SCP PUT.</p> <p>Default: -1 (for the entire file)</p>
Set File Size to Context Variable	Select the check box to save the file size to the specified context variable. This only applies to SFTP GET/STAT. Clear this check box if you do not want to save the file size to a context variable. This setting is the default.
Preserve File Mode (Permission)	<p>Select this check box to preserve the file mode (permission) if available for SFTP.</p> <p>Clear this check box if you do not want to preserve the file mode (permission). This is the default, and means that every user will have read and write permission.</p> <p>Note: It is recommended to initiate SFTP with "-p" and check the "Preserve File Mode (Permission)" box to ensure that the file permissions are preserved. Only the file mode is preserved. Other metadata such as the access modification times are not preserved. This check box is only available for the "SFTP" protocol and copy method "Upload To". It will also only work with an SFTP inbound listener.</p>
Fail if File Exists	Select this check box to abort the file upload if the file already exists. Clear this check box to upload and overwrite any existing file. This only applies to SFTP Upload commands.
Override maximum message size	<p>Select this check box to override the permitted maximum size of the routing message. Clear this check box to use the value set in the <i>io.xmlPartMaxBytes</i> cluster property. This check box is available only when you select Download From as the Command Type in the [Connection] tab.</p> <ul style="list-style-type: none"> • Restrict messages to: Enter the maximum permitted size of the response message, in bytes. You may specify a context variable.  • Allow unlimited message size (not recommended): Select this option to allow response messages of unlimited size. This is not recommended and should be used only under the direction of CA Technical Support.

Setting	Description
<i>Current WSS header handling:</i> Specify how to handle the security header. This section is available only when you select Upload To as the Command Type in the [Connection] tab.	
Don't modify the request Security header	<p>Instructs the Gateway to leave the security header in the outgoing SOAP request message as-is. The security header in the request may still have been modified if the Gateway needed to decrypt any encrypted material during message processing.</p> <p>Use this setting if the protected service needs to do its own checking of the request's original security header, or if the protected service does not care whether its request messages have a security header.</p> <p>For best performance, use this setting whenever possible to minimize the amount of per-request message modification.</p> <p>Note: Do not modify the Security header if the policy uses WS-Security. For more information, see the "Add or Remove WS-Security Assertion" on page 273.</p>
Remove Layer 7 actor and mustUnderstand attributes from processed Security header	<p>Instructs the Gateway to remove the "mustUnderstand" attribute and 'Layer 7' actor from the security header in the outgoing SOAP message.</p> <p>Use this setting if the presence of the Layer 7 actor causes issues with the back-end service. In certain cases, this actor may cause the back-end service to ignore the Security headers because it believes it is addressed to someone else. You will also use this setting if the back-end service does not support Security and would reject a request with "mustUnderstand" asserted on the Security header.</p> <p>An alternative might be to remove the Security header completely, however this will incur a performance penalty for the extra processing required to remove these from the messages. You may want to keep the Security headers intact for logging purposes.</p>
Remove processed Security header from request before routing	<p>Instructs the Gateway to remove any security header that was processed by the gateway before forwarding the request to the protected service.</p> <p>Use this setting when the protected service is not expecting security headers in the forwarded SOAP requests.</p>

8. Click **[OK]** when done.

Chapter 9: Service Availability Assertions

Notes: (1) Depending on which Gateway product you have installed, not all the assertions shown below may be available. See *Features by Product* in the *Layer 7 Policy Manager User Manual* for a list of which features are available for each product. (2) This category may also include custom-created encapsulated assertions. For more information, see "Working with Encapsulated Assertions" on page 126.

In the Policy Manager, the following assertions are available in the Service Availability category of the [Assertions] tab:

Apply Rate Limit Assertion	573
Apply Throughput Quota Assertion	578
Context Variables Created by This Assertion	579
Limit Availability to Time/Days Assertion	584
Look Up in Cache Assertion	585
Query Rate Limit Assertion	587
Context Variables Created by This Assertion	587
Query Throughput Quota Assertion	589
Context Variables Created by This Assertion	589
Resolve Service Assertion	590
Restrict Access to IP Address Range Assertion	592
Store to Cache Assertion	594

The Service Availability assertions define the access, availability, and throughput inclusions or restrictions for the service.

Apply Rate Limit Assertion

The *Apply Rate Limit* assertion allows you to limit the rate of transactions passing through the Gateway for a given user, client IP address, or other identifier. When this limit is reached, the Gateway can either begin throttling requests or it can attempt to delay the requests until the rate falls below the limit. You can also set a maximum concurrency level to prevent a user from monopolizing Gateway resources.

Use this assertion only if you need to limit the flow of transactions entering the Gateway. If you have a cluster of gateways, the limits entered in this assertion are divided among the number of "up" nodes in the cluster. A node is considered "up" if it has posted its status within the past 8 seconds (configurable via the *ratelimit.clusterStatusInterval* cluster

property). The Apply Rate Limit assertion will check the status of cluster nodes every 43 seconds (configurable via the `ratelimit.clusterPollInterval` cluster property).

Notes: (1) The Gateway will automatically adjust the rates internally should nodes be added or removed from a cluster. There is no need to modify the values in this assertion. (2) If no authenticated user is established in the policy, then the IP address of the requestor is used instead in the Apply Rate Limit assertion.


Using the Assertion


1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Apply Rate Limit...** in the policy window and choose **Rate Limit Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

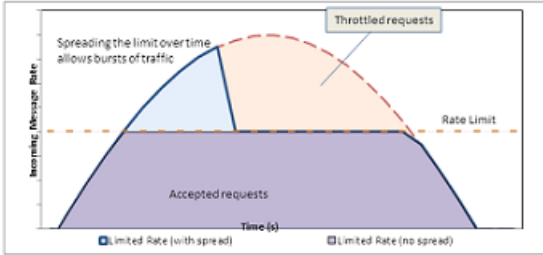

Figure 197: Rate Limit Properties


3. Configure the properties as follows:

Table 178: Rate Limit settings

Setting	Description
Maximum requests per second 	Specify how many requests per second should be processed by the Gateway or cluster. You can enter a context variable that resolves to the maximum requests value. Note: The context variable must either be single-value or multivalued with a specific index reference.
Cluster wide	If the Gateway cluster comprises more than one node, this setting determines whether the value entered in the Maximum requests per

Setting	Description
	<p>second field is split among the nodes or applied to each node.</p> <ul style="list-style-type: none"> Select this check box to split the value cross all the nodes in the cluster. For example, if the maximum is 100, each node in a 4-node cluster will be limited to 25 requests per second. If a node drops out of the cluster, the 100 limit is redistributed across the remaining three nodes. <p>This setting is the default and it replicates the functionality of the assertion prior to version 6.2.</p> <ul style="list-style-type: none"> Clear this check box to allow the maximum requests value on <i>each</i> node. For example, if the maximum is 100, each node in a 4-node cluster will be allowed 100 requests per second, resulting in an effective maximum of 400 requests per second. If one node drops out of the cluster, the effective maximum drops to 300 requests per second (3 x 100) .
Spread limit over X sec window 	<p>Determines whether to allow a burst of requests to be spread across a window of time or whether to enforce a hard cap.</p> <ul style="list-style-type: none"> Select the check box to allow requests to arrive in arbitrary bursts that exceed the Max requests per second rate over an X second window. This can avoid throttling of traffic over prolonged traffic bursts. You may enter a context variable containing the X second window value. This variable can be either single-value or multivalued with a specific index reference. <p>Tip: Setting the burst spread limit to 1 second replicates the "Allow burst traffic" functionality found in versions of the Apply Rate Limit assertion prior to version 6.1.</p> <ul style="list-style-type: none"> Clear the check box to disallow bursts. In this scenario, the Gateway will only accept requests arriving no sooner than <i>1/limit</i> of a second. For example, if the Max requests per second is 100, at least 1/100 second must have elapsed between requests. Requests that arrive sooner are either throttled or shaped (based on the "When limit exceeded" setting). Disallowing burst traffic is recommended only for advanced users. <p>Note: It is not recommended to disable burst traffic on a counter that will be servicing multiple concurrent requests, particularly at high rates. Doing so can lead to unintended throttling or delaying of multiple requests that arrive at exactly the same time.</p> <p>The following graph illustrates how spreading the limit will allow more traffic and throttle fewer requests.</p>

Setting	Description
	 <p>Tip: The effect is akin to a gas tank that slowly refills when not being used. Each request "consumes" some gas and the request fails if there is no more gas. The "Spread limit over" setting lets you control the size of the gas tank.</p>
Limit each	<p>Use the drop-down list to indicate how limiting should occur:</p> <ul style="list-style-type: none"> by the User or client IP address by the Authenticated user name by the Client IP address by the SOAP operation within the request by the SOAP namespace within the request by the Gateway node by a Custom value (enables a limit per value of a context variable); enter the node identifier followed by a context variable that will resolve to the correct entity during run time. <p></p> <p>This limit breakdown impacts both the maximum number of requests per second as well as the maximum concurrency.</p> <p>For example, if you choose "by client IP address" and set the maximum concurrency to 10 and maximum number of requests per second to 100, the assertion will fail if any incoming IP address exceeds either the concurrency of 10 or the 100 requests per second; all IP addresses combined are permitted to exceed these limits however. You can combine multiple instances of this assertion to impose difference limits by different breakdown factors, such as "maximum 10 per IP and maximum 100 for all combined".</p> <p>Tip: To help you construct a custom format, the entry box will display the actual node identifier and context variable associated with each of the other limit options once you've selected the Custom option. For example, when you first open the Rate Limit Properties, User or client IP is selected by default. Now, choose Custom and then reselect User or client IP. You will see that the actual coding behind this is <code><node identifier>-\${request.clientid}</code>.</p>
When limit exceeded	<p>Specify what should happen if the rate limit is exceeded:</p> <ul style="list-style-type: none"> Throttle: Excess requests will cause this assertion to fail and

Setting	Description
	<p>send audit code 6950 (<i>Rate limit exceeded on rate limiter XXXX</i>) to the audit log.</p> <ul style="list-style-type: none"> • Shape: The assertion will attempt to delay requests to avoid exceeding the limit. If the Gateway is unable to spare sufficient resources to hold a request any further, a 503 (<i>Service Unavailable</i>) error may still occur. • Log Only: The assertion will log that the rate limit has been exceeded, but the assertion will not fail. Note: The audit message 6950 will be logged. • Blackout for X sec: Select this check box to fail all requests for the next X minutes after the limit is exceeded, even if the rate of requests falls below the limits defined in this assertion. <p>Note: The number of threads that can be queued within a node is defined by the <i>ratelimit.maxQueuedThreads</i> cluster property. For more information, see "Rate Limit Settings" under Gateway Cluster Properties in the <i>Layer 7 Policy Manager User Manual</i>.</p>
Maximum concurrent requests 	<p>Indicate whether to enforce concurrency limits for a given named rate limiter (as specified by the Limit each setting).</p> <ul style="list-style-type: none"> • Unlimited: Concurrency is not enforced. A named rate limiter can have an unlimited number of active requests simultaneously in the Gateway or cluster. This may result in someone consuming a disproportionately high amount of system resources. • Limited to: Ensure that no named rate limiter can have more than the specified number of concurrent requests passing through this assertion. Requests that exceed the concurrency limit will cause the assertion to fail, with the audit event 6953 (<i>Concurrency exceeded on rate limiter XXXX</i>). <p>You can enter a context variable that contains the maximum concurrent requests value. This variable can be either single-value or multivalued with a specific index reference.</p> <ul style="list-style-type: none"> • Cluster wide: If the Gateway cluster comprises more than one node, this setting determines whether to the value entered in the Limited to field is split among nodes or to be applied to each node. This setting is the default and it replicates the functionality of the assertion prior to version 6.2. <ul style="list-style-type: none"> • Select this check box to split the value across all the nodes in the cluster. For example, if the maximum is 10, each node in a 5-node cluster will result in a concurrency limit of 2 requests per node. • Clear this check box to allow the maximum requests value on <i>each</i> node. For example, if the maximum is 10, every node in the cluster will be allowed 10 concurrent requests. <p>Additional note about how the concurrency limit works:</p>

Setting	Description
	<ul style="list-style-type: none"> The concurrency counter is incremented when a request passes through the Apply Rate Limit assertion (even if the assertion ends up failing). The counter is decremented once the request is completely finished.

- Click **[OK]** when done.

Apply Throughput Quota Assertion

The *Apply Throughput Quota* assertion allows you to limit the number of service requests permitted within a pre-determined time period. At runtime, the CA API Gateway utilizes a counter to enforce a defined quota. The Apply Throughput Quota assertion will fail should the quota limit be exceeded.

Since counters are global to the Gateway, different services can contain Apply Throughput Quota assertions that use the same counter. Similarly, using a single counter across different users or groups facilitates Service Level Agreement (SLA) migration in a policy that must enforce different SLAs for different identities.

Tips: (1) The counters used in this assertion are time-based and not based on the first request. For example, if the throughput quota is configured for "once per hour", the counter starts at the top of the hour, not when the first increment on the counter occurs. (2) There are several system properties that help you adjust the behavior of this assertion. To learn more, refer to the *com.17tech.hacounter.** properties in the "System Properties" appendix in the *Layer 7 Installation and Maintenance Manual* for your Gateway. Note that changing system properties should only be attempted by advanced users or while under the direction of CA Technical Support.

The Apply Throughput Quota assertion is typically used as follows in a policy:

Simple SLA:

Authentication assertion
Group membership assertion
Throughput quota: maximum "x" per "y"
Routing assertion

Two different SLAs for two different groups of users:

Authentication assertion
At least one assertion must evaluate to true
All assertions must evaluate to true

Group membership A

Throughput quota: maximum "x" per "y"

All assertions must evaluate to true

Group membership B

Throughput quota: maximum "z" per "k"

Routing assertion

Note: If no authenticated user is established in the policy, then the IP address of the requestor is used instead in the Apply Throughput Quota assertion.

Context Variables Created by This Assertion

The Apply Throughput Quota assertion sets the following context variables. **Note:** The `<prefix>` is set in the assertion properties (Figure 198).

Table 179: Context variables created by Apply Throughput Quota assertion

Variable	Description
<code>\${<prefix>.id}</code>	Returns the name of the counter used in the assertion.
<code>\${<prefix>.value}</code>	Returns the current value of the counter.
<code>\${<prefix>.period}</code>	Returns the period used for the counter (second, hour, day, month).
<code>\${<prefix>.user}</code>	Returns the user name associated with the last instance.
<code>\${<prefix>.max}</code>	Returns the maximum quota value.

Note: If a prefix is not specified, then the default prefix **counter** is used.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the Throughput Quota Properties automatically appear; when modifying the assertion, right-click **Apply Throughput Quota** in the policy window and choose **Throughput Quota Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Throughput Quota Properties

Quota

Max: 200 per month

Limit each: Authenticated user

☐ Log Only

Counter

Counter ID: 7b6dcb75-\${request.authenticateduser.id}-\${request.authenticateduser.providerid}

☐ Always increment

☒ Increment only when still within quota

☐ Decrement

☐ Reset

☐ By value

Scalability

Consistency Scalability

Improves scalability further but also permits quota overflows.


Variable Prefix:


OK


Figure 198: Throughput Quota Properties


3. Configure the properties as follows.

Table 180: Throughput Quota settings

Setting	Description
Quota	
Max 	Enter a request limit. The value can be any integer between 1 and $(2^{31})-1$. You can also enter a context variable that contains the request limit. Tip: If a value larger than $(2^{31})-1$ is required, then a context variable must be specified.
per	Choose the unit of time from the drop-down list. The units represent closed time periods. For example, if a quota of 1000 requests per hour is configured, then

Setting	Description
	the Gateway will enforce a limit of 1000 requests between the start and end of each hour of the day.
Limit each	<p>Choose the method for limiting the quota from one of the following:</p> <ul style="list-style-type: none"> • Authenticated user: Choose this enforcement option to apply the assertion to each individual identity, or authenticated user, in the policy. An authenticated user is an individual user who has been authenticated to an identity provider. For example, if a quota of 1000 requests per hour is configured for a policy, then each individual requester will be permitted 1000 requests per hour. This setting is the default setting. Tip: This option corresponds to the "Quota per requestor" setting found in previous releases. • Client IP: Choose this option to apply the assertion to the client IP address. • SOAP operation: Choose this option to apply the assertion to the SOAP operation within the request. • SOAP namespace: Choose this option to apply the assertion to the SOAP namespace within the request. • Gateway cluster: Choose this enforcement option to apply the assertion to the Gateway cluster. For example, if a quota of 1000 requests per hour is configured for a policy, then the combined requests per hour for all requesters in the policy cannot exceed 1000. Tip: This option corresponds to the "Global quota" setting found in previous releases. • Custom: Choose this option to monitor throughput using a custom counter entered in the Counter ID field. You will use this option if none of the predefined counters meet your needs. 
Log Only	<p>This check box controls what the assertion does when the quota is exceeded.</p> <ul style="list-style-type: none"> • Select this check box to have the assertion to generate a log entry when the throughput quota is exceeded; the assertion does not fail. • Clear this check box to have the assertion fail if the quota is exceeded. <p>Note: The Log Only check box is not available when the Decrement option is selected below .</p>
<i>Counter</i>	

Setting	Description
Counter ID 	<p>The Counter ID is automatically populated based on the counter chosen in the Limit each field. If a Custom counter was selected, define the counter here.</p> <p>The following identifiers are set for each counter selected:</p> <ul style="list-style-type: none"> • Authenticated user: <code><uuid>- \${request.authenticateduser.id}- \${request.authenticateduser.providerid}</code> • Client IP: <code><uuid>-\${request.tcp.remoteAddress}</code> • SOAP operation: <code><uuid>-\${request.soap.operation}</code> • SOAP namespace: <code><uuid>-\${request.soap.namespace}</code> • Gateway cluster: <code><uuid></code> • Custom: Any string, which must be entered directly into the Counter ID field. This can be anything, including context variables. <p>Tip: If you need to modify one of the present Counter IDs, choose the "Custom" option under Limit each to make the Counter ID field editable. Modifying an existing Counter ID is also a quick way to create a custom counter.</p> <p>Example: You wish to limit the number of requests per hour to 1000 per client IP. To do this, enter "1000" for "Max" and then choose "Client IP" for Limit each. When this assertion is run, a counter is created for each requester coming from a different IP address. If more than 1000 requests per hour are sent from the same IP address, the Throughput Quota assertion blocks that requester.</p> <p>Since a counter can only be incremented once within a single request context, the same counter can be used for different Throughput Quota assertions within the same policy. In other words, regardless of the number of Throughput Quota assertions in a policy, the Gateway will only count each request once towards the quota when the assertions use the same counter.</p>
counter increment/ decrement options	<ul style="list-style-type: none"> • Always increment: Always increment the counter before determining whether the quota limit has been reached. By default, the increment value is 1 unless overridden in the "by value" field. <p>If the resulting counter value exceeds the quota, the assertion may fail, depending on the "Scalability" setting below.</p> <ul style="list-style-type: none"> • Increment only when still within quota: Only increment the counter if the resulting value will not exceed the quota. By default, the increment value is 1 unless overridden in the "by value" field. <p>If the quota limit has not been reached, then the request is counted and the assertion succeeds. If the quota limit has been reached, the assertion may fail, depending on the "Scalability"</p>

Setting	Description
	<p>setting below, and the initiating request is not counted towards the quota.</p> <ul style="list-style-type: none"> • Decrement: Decrement the counter by a single count. By default, the decrement value is 1 unless overridden in the "by value" field <p>Use this option in a policy to ensure that service processing errors or incomplete transactions are not counted towards the quota.</p> <ul style="list-style-type: none"> • Reset: Reset all counter values to zero. The "Quota" and "Scalability" settings will be disabled. • By value: Allows you to specify the increment/decrement behavior:  <ul style="list-style-type: none"> • Select this check box to increment or decrement the counter by a specific integer value. You may reference context variables. • Clear this check box to increment or decrement by 1. <p>Tip: One good use of the "By value" setting is when bandwidth quotas must be observed. Simply set the "By value" increment to <code>\${response.size}</code>. This will increment the counter by the actual size of each response message, giving you an accurate indication of the bandwidth being used.</p>
Scalability	<p>Set the slider to your preferred mix of consistency vs. scalability:</p> <ul style="list-style-type: none"> • Move the slider to Consistency have the Gateway enforce the quota restrictions, without exception. This may not provide optimal performance in situations where brief bursts of messages occur. <p>Note: This setting replicates the behavior of the Apply Throughput Quota assertion prior to version 7.1 and is not recommended unless it is critical that the quota <i>never</i> be exceeded.</p> <ul style="list-style-type: none"> • Move the slider to the midpoint to improve scalability and performance, permitting brief quota overflows under rare conditions. • Move the slider to Scalability for maximum performance and scalability, permitting quota overflows more frequently.
Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

4. Click **[OK]** when done.

Limit Availability to Time/Days Assertion

The *Limit Availability to Time/Days* assertion allows you to restrict service access by a Gateway time and/or day interval. When the Gateway receives a request for the service, it will check the time and/or day restrictions before allowing the message to proceed.

Note: All time and/or day restrictions are based on the time zone of the Gateway.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the **Time/Day Availability Properties** automatically appear; when modifying the assertion, right-click **Limit Availability to...** in the policy window and select **Time/Day Availability Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 199: Time/Day Availability Properties

- Configure the properties as follows:

Table 181: Time/Day Availability settings

Restriction	Description
Restrict access by time	Select the Restrict Time of Day check box and then specify the time period when access is permitted. The time period entered is automatically converted to UTC (Coordinated Universal Time) and displayed on the dialog.
Restrict access by day	<p>Select the Restrict Day of Week check box and then specify the day range when access is permitted. Selecting the same day in both fields will allow access only for that day.</p> <p>Tip: The Gateway uses UTC so take that into account when implementing day restrictions. For example, using the sample screen in Figure 199, access after 17:00 local time on Friday will be disallowed because it will already be Saturday UTC time.</p>

You can restrict access by time and/or day.

4. Click **[OK]** when done.

Look Up in Cache Assertion

The *Look Up in Cache* assertion is used to look up an item in a cache store that was placed by the "Store to Cache Assertion" on page 594. If the lookup is successful, the item is placed into the message target of your choice (request, response, or context variable). If the lookup is unsuccessful, the assertion returns status code 600 ("Assertion Falsified") for cache misses or 601 ("Error in Assertion Processing") on processing errors.

You can optionally override the Content-Type of the cached item during lookup.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Policy Example

The Look Up in Cache assertion is normally paired with the [Store to Cache](#) assertion. The "Look up" assertion should be placed before the routing assertion inside of an "[At least one...](#)" folder. The "Store" assertion should follow immediately afterward:

[Not shown: XPath to extract clientCountry and SKU from request]

[Set variable](#) "cachekey" to "Cache01-`${request.url}-${request.soap.operation}-${clientCountry}-${SKU}`"

[At Least One Assertion Must Evaluate to True](#)

Response: Look Up in Cache with key "\${cachekey}"

[All Assertions Must Evaluate to True](#)

Route via HTTP to URL `http://backend/anydestination`

Response: Store to Cache with key `"${cachekey}"`

In this example, the Gateway will attempt look up `"${cachekey}"` in the cache first. If it is successful, the entry retrieved from the cache is used for subsequent processing, sparing the backend service from needing to respond to the request. If `"${cachekey}"` is not found in the cache, the request is then routed to the back-end service and the response is stored into cache.

Note that the "Store to Cache Assertion" on page 594 should follow the routing assertion.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Look Up in Cache** in the policy window and select **Cache Lookup Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

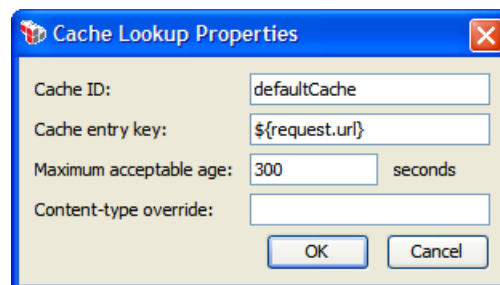





Figure 200: Cache Lookup Properties

3. Configure the dialog as follows:

Table 182: Cache Lookup settings

Setting	Description
Cache ID 	Specify the identifier for the cache to be searched. This identifier was defined in the Store to Cache assertion.
Cache entry key 	Specify the key to be looked up in the cache. This key should normally match the one configured in the Store to Cache assertion.

Setting	Description
Maximum acceptable age 	Specify the maximum acceptable age (in seconds) of a cached item. If an item is below this age, it will be retrieved from the cache and returned in the response. If the cached item exceeds this age, it will not be retrieved. You may reference context variables. The default is 300 seconds.
Content-type override	The original Content-Type is cached and will be used in the "Look Up" message. You may override the original Content-Type if necessary, otherwise leave this field blank.

- Click **[OK]**.

Query Rate Limit Assertion

The *Query Rate Limit* assertion is used to query the status of a custom rate limit counter from the [Apply Rate Limit assertion](#). The results are then placed into context variables.

Tip: The Query Rate Limit assertion is most useful when a custom counter ID has been used in the [Apply Rate Limit assertion](#).

Context Variables Created by This Assertion

The Query Rate Limit assertion sets the following context variables with the query results.

Note: The *<prefix>* is set in the assertion properties (Figure 201) and is optional. There is no default.

Table 183: Context variables created by Query Rate Limit assertion

Variable	Description
<i>\$<prefix></i> .counter.name	The full name of the matching counter.
<i>\$<prefix></i> .counter.requestsremaining	The number of requests the counter would allow at this moment (neglecting concurrency).
<i>\$<prefix></i> .counter.concurrency	The number of requests currently using the counter.
<i>\$<prefix></i> .counter.blackoutmillisremaining	The number of milliseconds of blackout time remaining if the counter is blacked out. Otherwise, this value is 0 (zero).

Using the Assertion

- Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Query Rate Limit for limit per <counter name>** in the policy window and select **Rate Limit Query Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 201: Rate Limit Query Properties

3. Configure the dialog as follows:

Table 184: Rate Limit Query settings

Field	Description
Existing Counter Name	<p>Enter the name of the rate limit counter being queried.</p> <p>The counter name entered here should be the actual <i>resolved</i> counter ID from the Apply Rate Limit assertion. If you simply repeat the context variable from the other assertion, you must ensure that the query uses matching information, otherwise the results will be incorrect.</p> <p>For example, the Apply Rate Limit assertion uses the counter ID: "QA-<code>\${request.authenticateduser}</code>". If you wish to query for user "jsmith". In this case, you would enter the counter name "QA-jsmith" in the Query Rate Limit assertion. If you simply repeated "QA-<code>\${request.authenticateduser}</code>" as the counter name, the results will be correct only if the user performing the query is "jsmith".</p>
Variable Prefix	<p>Optionally, enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy. If set, the format <code>\${<prefix>.counter.name}</code> is used instead of <code>\${counter.name}</code>.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

4. Click **[OK]**.

Query Throughput Quota Assertion

The *Query Throughput Quota* assertion is used to query the status of a custom throughput quota counter from the [Apply Throughput Quota assertion](#). The results are then placed into context variables.

Tip: The Query Throughput Quota assertion is most useful when a custom counter ID has been used in the [Apply Throughput Quota assertion](#).

Context Variables Created by This Assertion

The Query Throughput Quota assertion sets the following context variables. **Note:** The *<prefix>* is set in the assertion properties (Figure 202) and is optional. There is no default.

Table 185: Context variables created by Query Throughput Quota assertion

Variable	Description
<i>\$<prefix></i> .counter.name	The full name of the matching counter.
<i>\$<prefix></i> .counter.sec	The count for the previous second.
<i>\$<prefix></i> .counter.min	The count for the previous minute.
<i>\$<prefix></i> .counter.hr	The count for the previous hour.
<i>\$<prefix></i> .counter.day	The count for the previous day.
<i>\$<prefix></i> .counter.mnt	The count for the previous month.
<i>\$<prefix></i> .counter.lastupdate	The date the counter was last updated, in the format yyyy-mm-dd.

The "*<prefix>*" is optional.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click **Query Throughput Quota for <counter name>** in the policy window and select **Throughput Quota Query Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

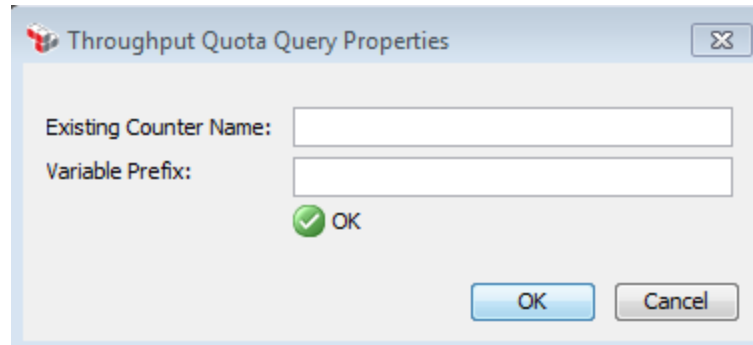


Figure 202: Throughput Quota Query Properties

3. Configure the dialog as follows:

Table 186: Throughput Quota Query settings

Field	Description
Existing Counter Name	<p>Enter the name of the throughput quota counter being queried.</p> <p>The counter name entered here should be the actual <i>resolved</i> counter ID from the Apply Throughput Quota assertion. If you simply repeat the context variable from the other assertion, you must ensure that the query uses matching information, otherwise the results will be incorrect.</p> <p>For example, the Apply Throughput Quota assertion uses the counter ID: "QA-<code>\${request.authenticateduser}</code>" and you wish to query for user "jsmith". In this case, you would enter the counter name "QA-jsmith" in the Query Throughput Quota assertion. If you simply repeated "QA-<code>\${request.authenticateduser}</code>" as the counter name, the results will be correct only if the user performing the query is "jsmith".</p>
Variable Prefix	<p>Optionally, enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy. If set, the format <code>\${<prefix>.counter.name}</code> is used instead of <code>\${counter.name}</code>.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

4. Click **[OK]**.

Resolve Service Assertion

The *Resolve Service* assertion allows you to perform manual service resolution (by URI). When this assertion is encountered in a policy, the normal service resolution logic of the Gateway is bypassed. (This resolution logic is described in *Understanding the Service Resolution Process* in the *Layer 7 Installation and Maintenance Manual*.)

One useful application of this assertion would be to use an [Evaluate Request XPath](#) assertion to seek a particular element (for example, `//ns:SpecialElement`). If this element is present, the Resolve Service assertion is used to route the request to a particular URI (for example, `/specialElementHandler`). If this element is not present, then the normal service resolution process is used.

Note: The Resolve Service assertion must be placed in a *message-received* [global policy fragment](#) (or in a policy fragment that is imported into this global policy fragment) in order for it to execute prior to service resolution. In all other scenarios, this assertion will fail because the service will have already been resolved by the time the service policy executes.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the Resolve Service Properties automatically appear; when modifying the assertion, right-click **Resolve Service with URI...** in the policy window and select **Resolve Service Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

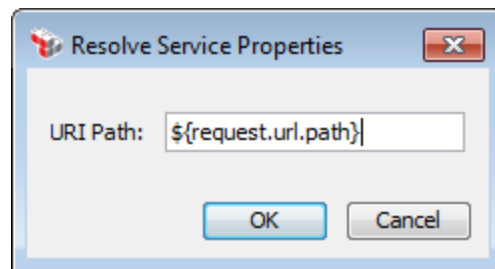


Figure 203: Resolve Service Properties

3. Enter or edit the URI path to be used for service resolution.

Tip: You can use the `$(request.url.path)` context variable to specify the URI path. This variable returns the path of the Gateway URL where the message was received.

4. Click **[OK]** when done.

Restrict Access to IP Address Range Assertion

The *Restrict Access to IP Address Range* assertion allows you to restrict or allow service access based on the IP address of the web service or XML application requestor.

The IP address of the requestor considered when this assertion is run can either be the actual remote IP address available at the TCP level or a string extracted from the message. The latter case can be used, for example, when requests are first forwarded through multiple network components before arriving at the Gateway. If such network components are configured to pass down the original IP address through an HTTP or SOAP header, that information source can then be configured in the Restrict Access to IP Address Range assertion using context variables.

Note: When using a context variable as the source for the IP address, that source is first filtered using the following Regular Expression: `\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}`. This will filter out any extraneous information, such as a client port number. The accepted formats are "ipv4_literal", "ipv4_literal:port", "ipv6_literal", "[ipv6_literal]:port".

Using the assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **IP Address Range Properties** automatically appear; when modifying the assertion, right-click **[Allow|Forbid] IP Address Range** in the policy window and select **IP Address Range Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

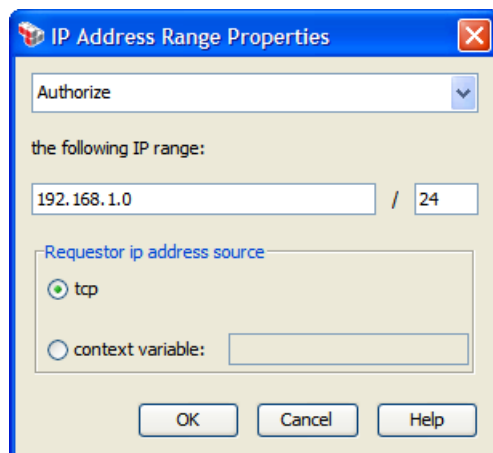



Figure 204: IP Address Range Properties

3. Configure the properties as follows:

Table 187: IP Address Range settings

Setting	Description
Authorize/Forbid	From the drop-down list, select Authorize if you are permitting access to the IP range listed. Select Forbid if you are restricting access to the IP range listed.
IP range	Enter the allowable or forbidden IP address and/or "bits" in accordance with the CIDR (Classless Inter-Domain Routing) standard. Both IPv4 and IPv6 addresses are supported.
Requestor IP address source 	<p>Specify how the Gateway should determine the source IP address:</p> <ul style="list-style-type: none"> Select TCP to use the IP address associated with the TCP request. Select Context variable and then enter any context variable that resolves to a valid IP address. The default is request.tcp.remoteAddress, which will return the remote address of the TCP connection through which the message arrived. <p>To learn more about context variables, see Context Variables in the <i>Layer 7 Policy Manager User Manual</i>.</p>

4. Click **[OK]** when done.

Store to Cache Assertion

The *Store to Cache* assertion is used to store messages or a string from a target message to a cache store of your choice. You can then use the "Look Up in Cache Assertion" on page 585 to retrieve the cached contents. This will reduce the load on back-end services and potentially improve response times.

This assertion will always succeed. If the assertion encounters problems creating or updating a cache, it is reported in the system audit log, but the assertion will not fail.

All cache entries are deleted when a Gateway node shuts down.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Policy Example

The Store to Cache assertion is normally paired with the [Look Up in Cache](#) assertion. The "Look Up" assertion should be placed before the routing assertion inside of an "[At least one...](#)" folder. The "Store" assertion should follow immediately afterward. The following policy fragment is an example:

```
[Not shown: XPath to extract clientCountry and SKU from request]
Set variable "cachekey" to "Cache01-{$request.url}-{$request.soap.operation}-
{$clientCountry}-{$SKU}"
At Least One Assertion Must Evaluate to True

Response: Look Up in Cache with key "{$cachekey}"
All Assertions Must Evaluate to True

Route via HTTP to URL http://backend/anydestination
Response: Store to Cache with key "{$cachekey}"
```

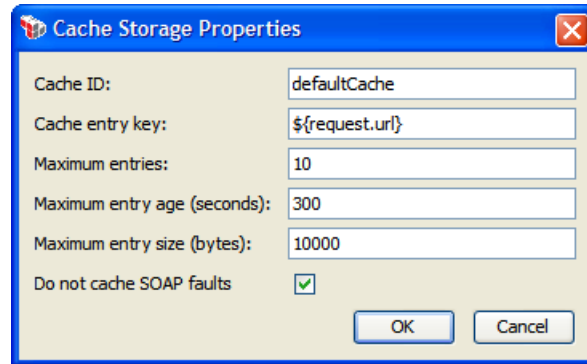
In this example, the Gateway will attempt look up "{\$cachekey}" in the cache first. If it is successful, the entry retrieved from the cache is used for subsequent processing, sparing the backend service from needing to respond to the request. If "{\$cachekey}" is not found in the cache, the request is then routed to the back-end service and the response is stored into cache.

Note that the Store to Cache assertion should follow the routing assertion.

Using the Assertion

1. Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Store to Cache** in the policy window and select **Cache Storage Properties** or double-click the assertion in the policy window. The assertion properties are displayed.







The dialog box titled "Cache Storage Properties" contains the following fields and controls:


- Cache ID:** Text field with value "defaultCache".
- Cache entry key:** Text field with value "\${request.url}".
- Maximum entries:** Text field with value "10".
- Maximum entry age (seconds):** Text field with value "300".
- Maximum entry size (bytes):** Text field with value "10000".
- Do not cache SOAP faults:** Check box, which is checked.
- Buttons:** "OK" and "Cancel" buttons at the bottom right.

Figure 205: Cache Storage Properties

3. Configure the dialog as follows:

Table 188: Cache Storage settings

Field	Description
Cache ID 	<p>Enter a label to identify the cache store to use. If no cache store exists with that label, a new cache store is created.</p> <p>The cache identifier is used in the Look Up in Cache assertion to look up entries in the corresponding cache store.</p>
Cache entry key 	<p>Cached entries are identified within a cache store by a unique cache entry key. This key is used by the Look Up in Cache assertion to look up entries and retrieve them from a cache store.</p> <p>Specify the key or reference a context variable that will contain the key. You may reference context variables.</p> <p>Note: The cache entry key is configured for a group of related cache "store" and "lookup" assertions and should normally be the same expression.</p>
Maximum entries 	<p>Enter the maximum number of cached entries that the store can hold. When this maximum is reached, each new item will replace the oldest one in the store. You may reference context variables.</p> <p>The default is 10.</p>
Maximum entry age 	<p>Enter the maximum age (in seconds) of items in the cache before they are discarded. You may reference context variables.</p> <p>The default is 300 seconds.</p>

Field	Description
Maximum entry size 	Enter the maximum size (in bytes) of the items to cache. The default is 10000 bytes.
Do not cache SOAP faults	This check box determines whether SOAP fault responses are cached: <ul style="list-style-type: none"> • Select this check box to exclude SOAP faults from being cached. This may help prevent the cache from being filled too quickly, especially if many SOAP faults are generated. • Clear this check box to include SOAP faults in the cache. Note that this will cause a SOAP fault to always be returned until the cached entry expires, even when the SOAP fault condition no longer occurs.

4. Click **[OK]**.

Chapter 10: Logging, Auditing, and Alerts Assertions

In the Policy Manager, the following assertions are available in the Logging, Auditing, and Alerts category of the [Assertions] tab:

Message Auditing	597
System Audits	598
Administrative Audits	598
Policy Message Audits	598
Add Audit Detail Assertion	600
Audit Messages in Policy Assertion	602
Capture Identity of Requestor Assertion	604
Customize SOAP Fault Response Assertion	607
SOAP Faults	611
Send Email Alert Assertion	612
Send SNMP Trap Assertion	615

The Logging, Auditing, and Alerts assertions set the threshold for the Gateway audit messages in the Gateway Audit Events window and configure the properties for SNMP trap alerts and email alerts.

Note: This category may also include custom-created encapsulated assertions. For more information, see "Working with Encapsulated Assertions" on page 126.

Message Auditing

The CA API Gateway generates three types of audit messages:

- **System audits**, which you cannot control
- **Administrative audits**, which you can control using the cluster property *audit.adminThreshold*
- **Policy message audits**, which allows you a high degree of control

The following sections provide more details about each audit type.

System Audits

These are internal messages that are constantly generated in the background by the Gateway. These messages typically describe "housekeeping" tasks such as: server starting , license updated, connecting to a JMS endpoint, etc.

System audit events are normally rated *Fine*, *Finer*, or *Finest* in the severity scale, although some may be rated *Info*. System audit events are always available in the audit event log (you will need to set the filter slider to "All" to see all these events).

You have no control over system audits: they happen automatically, without requiring any assertions.

Administrative Audits

These are messages that occur when an administrative action is performed via the Policy Manager , via an administrative API, or through the Enterprise Service Manager. Examples of such actions include: publishing or updating a policy, creating a user, etc.

You can control which of these messages you want to save by using the *audit.adminThreshold* cluster property. By default, all messages at level *Info* or higher are logged.

Policy Message Audits

These are messages generated during the processing of a policy. The bulk of these are simply informational messages that have the severity level *Info*. The more important messages are rated *Warning* or *Severe*. By default, the Gateway is set to save only *Warning* or *Severe* messages; *Info* messages are eventually discarded. This behavior is by design, to prevent your audit log from being cluttered with a mass of informational messages. In most instances, you are only interested in knowing when something goes wrong.

There are several cluster properties that can be used by advanced users to more precisely control policy message audits during the auditing process. Of particular importance are the following two properties:

audit.messageThreshold
audit.detailThreshold

The values of these properties, plus the levels selected in the [Audit Messages in Policy](#) and [Add Audit Detail](#) assertions, will determine whether a message is logged and at what level.

Additional resources:

- To learn more about the auditing cluster properties, see Audit Cluster Properties in the *Layer 7 Policy Manager User Manual*. To learn about the interaction between the threshold cluster properties and the auditing-related assertions, see Table 189 below.
- To learn how to use context variables to extract a wide variety of details from an audit event, see Working with the Audit Sink Policy in the *Layer 7 Policy Manager User Manual*.

Expanding the Scope of Policy Message Audits for Troubleshooting

If you are troubleshooting an elusive problem or if you want to see all the informational messages, add an [Audit Messages in Policy](#) assertion in the policy and set its "trigger" severity level to *Warning*. What this does is elevate all informational messages to a *Warning* severity level as they pass through the assertion. In other words:

"Info" messages become --> "Warning" messages

"Warning" messages remain --> "Warning" messages

"Severe" messages remain --> "Severe" messages

This results in everything being recorded to the audit event log, as all messages now meet the preset threshold of *Warning*. This is illustrated in the following diagram:

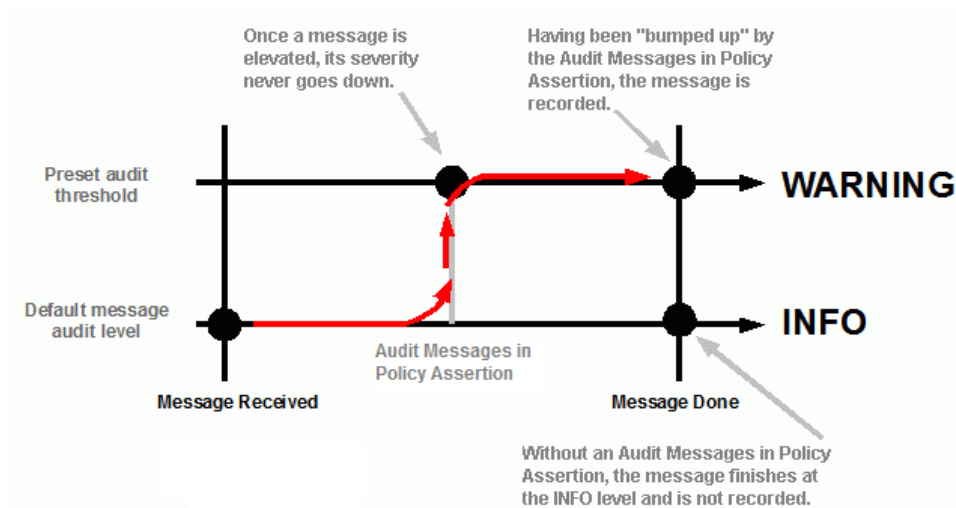


Figure 206: Audit message path

Note: A message is elevated only if an [Audit Messages in Policy](#) assertion is present *and* the level in that assertion is set to *Warning*. Without elevation, only the more important messages are saved to the audit events log. The Audit Messages in Policy assertion can only boost a level or leave it unchanged. Subsequent invocations of this assertion in a policy cannot *lower* a severity level. See Table 189 below for a summary of the interactions.

Table 189: Interaction between cluster properties and auditing assertions

Incoming Message Type	"Audit Messages in Policy Assertion" in policy	Cluster-wide properties		Result
		<i>audit. message Thres hold</i>	<i>audit. detailThreshold</i>	
Audit 'Message' @ 'INFO'	'Info' or not present	'WARNING'	'INFO'	Not Logged
		'INFO'	'INFO'	Logged at 'INFO' level
	'WARNING'	'WARNING'	'INFO'	Logged at 'WARNING'
Audit 'Detail' @ 'INFO'	'Info' or not present	'WARNING'	'INFO'	Logged at 'INFO' level
		'WARNING'	'WARNING'	Not Logged
	'WARNING'	'WARNING'	'INFO'	Logged at 'WARNING'
Audit 'Message' @ 'WARNING'	'Info' or not present	'WARNING'	'INFO'	Logged at 'WARNING'
		'INFO'	'INFO'	
	'WARNING'	'WARNING'	'INFO'	
Audit 'Detail' @ 'WARNING'	'Info' or not present	'WARNING'	'INFO'	Logged at 'WARNING'
		'WARNING'	'WARNING'	
	'WARNING'	'WARNING'	'INFO'	

Add Audit Detail Assertion

The *Add Audit Detail* assertion lets you define a custom message that can enhance the context of an audit message. These messages are then recorded either in the audit records or a Gateway log, depending on how the assertion is configured. The custom message will also appear the "Associated Logs" tab in the Event Details Pane of the Gateway Audit Events window if audit details are directed to the audit log.

For more information about defining logs, see Managing Log Sinks in the *Layer 7 Policy Manager User Manual*.

To learn more about how this assertion interacts with the [Audit Messages in Policy](#) assertion and with pertinent cluster properties, see "Message Auditing" on page 597.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the Audit Detail Properties automatically appear; when modifying the assertion, right-click **Add Audit Details...** in the policy window and select **Audit Detail Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

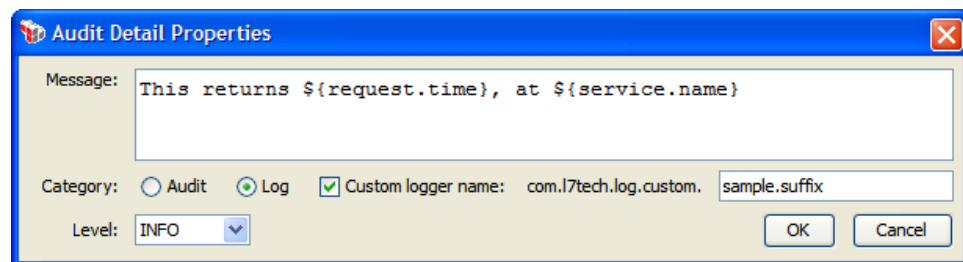




Figure 207: Audit Detail Properties

- Configure the properties as follows:

Table 190: Audit Detail settings

Setting	Description
Message 	Type a message in the box. This message will be displayed when the audit appears in the Gateway Audit Events window. Tip: Include context variables within the message to reveal additional information about the audit condition, if necessary.
Audit	Select this option to direct the audit detail message to the Audit log sink. This option replicates the default behavior of this assertion prior to version 5.4. Tip: Audit logs are the log sinks with the category "Audits". For more information, see Managing Log Sinks in the <i>Layer 7 Policy Manager User Manual</i> .
Log	Select this option to direct the audit detail message to the Gateway log sink. This is useful for situations where (for example) the logged information is too large to be comfortably stored in the audit database for extended periods of time. For example, storing trace information

Setting	Description
	<p>from a policy debug tracing.</p> <p>Tip: Gateway logs are the log sinks with the category "Gateway Log". For more information, see Managing Log Sinks in the <i>Layer 7 Policy Manager User Manual</i>.</p>
Custom logger name 	<p>Select this check box if you want the logged information to be identified by a custom logger name, rather than the default logger name <code>com.l7tech.server.policy.assertion.ServerAuditDetailAssertion</code>.</p> <p>If you choose to use a custom logger name, enter a suffix to be added to the custom logger name, to ensure uniqueness. You may reference context variables.</p> <p>Note: If a specified context variable cannot be resolved during run time, the default logger name shown above is used.</p>
Level	<p>Select a severity level for your message from the drop-down list. This level, along with the level set in the Audit Messages in Policy assertion, determines whether your message appears in the Gateway Audit Events window.</p>

- Click **[OK]** when done.

Audit Messages in Policy Assertion

The *Audit Messages in Policy* assertion is used to enable auditing of [messages](#) within a policy. It records events pertaining to the processing of a [policy](#)—for example, assertion violations, authentication failures, routing errors, etc. You can view these events later in the Gateway Audit Events window.

For example, when used in an [At least one assertion must evaluate to true](#) assertion folder after an [Evaluate Request XPath](#) assertion, the Audit Messages in Policy assertion will execute and audit the request message only if the XPath assertion fails. When this happens, XPath query results are reported in the Gateway Audit Events window. If the XPath assertion in this scenario succeeds, then the Audit Messages in Policy assertion does not execute.

Tip: To learn more about the auditing process, including how the Audit Messages in Policy assertion interacts with the [Add Audit Detail](#) assertion and the various cluster properties, see "Message Auditing" on page 597.

Using the Assertion

- Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Audit Messages in Policy** in the policy window and choose **Audit Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

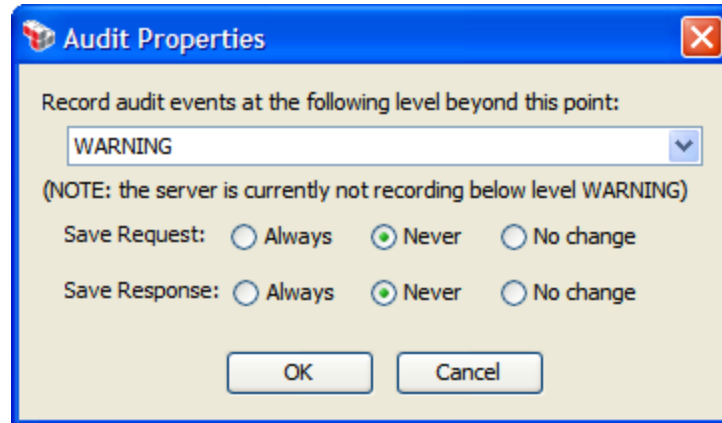


Figure 208: Audit Properties

3. Configure the properties as follows:

Table 191: Audit settings

Setting	Description
Record audit events at the following level beyond this point	<p>This setting changes the severity of the logged messages to either "Info" or "Warning". Whether the message is ultimately saved to the database depends on the <i>audit.messageThreshold</i> cluster property .</p> <p>Choose WARNING to set the severity of all messages to "Warning". This will cause all messages to be logged, regardless of whether the <i>audit.messageThreshold</i> cluster property is set to INFO or WARNING.</p> <p>Choose INFO to set the severity of all messages to "Info". Whether the messages are then logged depends on the <i>audit.messageThreshold</i> cluster property:</p> <ul style="list-style-type: none"> • If the cluster property is set to INFO, all messages will be logged. • If the cluster property is set to WARNING, no messages will be logged. <p>For a detailed description of the effects of the "trigger" threshold on auditing, see "Message Auditing" on page 597.</p>
Save request Save response	<p>Indicate whether to save the code of the request or response:</p> <ul style="list-style-type: none"> • Always: Save the code.

Setting	Description
	<ul style="list-style-type: none"> • Never: Do not save the code. • No change: If used in a service policy, this is the same as 'Never'. If used in a debug trace policy, this setting will preserve the setting of any Audit Messages in Policy assertion in the service policy (if no such assertion appears in the service policy, then the code is not saved). <p>Saving the code for the request/response will allow you to view them later in the Event Details Pane of the Gateway Audit Events window.</p> <p>Special note for Trace Policy</p> <p>When the Audit Messages in Policy assertion appears in a debug trace policy, the "Always" and "Never" settings here will <i>override</i> the equivalent settings in the target service policy. If there is no Audit Messages in Policy assertion in the service policy but one in the trace policy and it is set to "Always", then the code will be saved.</p>

Note: Recording all message events or saving request/response code will increase the size of your log substantially

4. Click **[OK]** when done.

Capture Identity of Requestor Assertion

The *Capture Identity of Requestor* assertion is used to determine the identity of a requestor (that is, the customer) for auditing or reporting using any of the following methods:

- By capturing the requestor's IP address
- By capturing the requestor's authenticated User ID (where available)
- By capturing the value from a context variable that contains identifying information about the requestor (for example, from a context variable created by the [Evaluate Request XPath](#), [Evaluate Response XPath](#), or [Evaluate Regular Expression](#) assertions, or a context variable that extracts information from the HTTP header)

You can define up to five mappings in a Capture Identity of Requestor assertion, however there can only be a maximum of five distinct mappings per policy, regardless of how many assertions are present. A "distinct mapping" is defined as follows:

- Each context message mapping consists of three parts: *Type*, *Key*, *Value*
- Mappings are distinct if their Types differ OR if the Type is the same, the Keys differ

Example 1: These mappings are distinct:

Mapping 1: IP Address, IP_ADDRESS (SYSTEM DEFINED)

Mapping 2: Custom Mapping, My_Value, Gold_Medal

Example 2: These mappings are also distinct:

Mapping 1: Custom Mapping, Value_A, Gold_Medal

Mapping 2: Custom Mapping, Value_B, Gold_Medal

The information that is captured can be viewed in the [Details] tab of the Gateway Audit Events window:

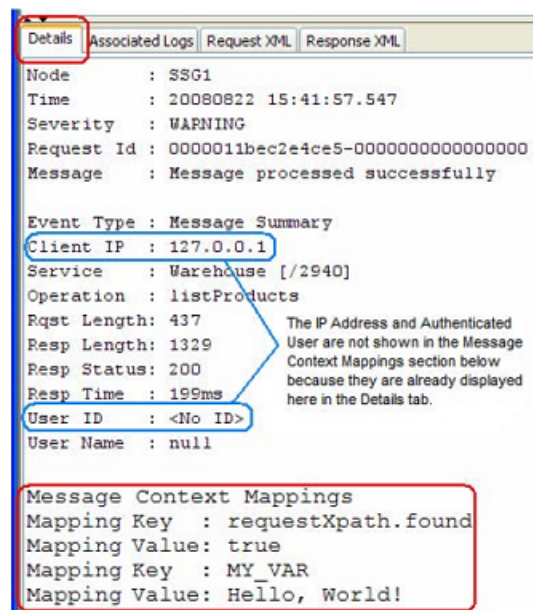


Figure 209: Viewing message context mappings in the Gateway Audit Events window

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Capture Identity of Requestor** in the policy window and select **Requestor Identity Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

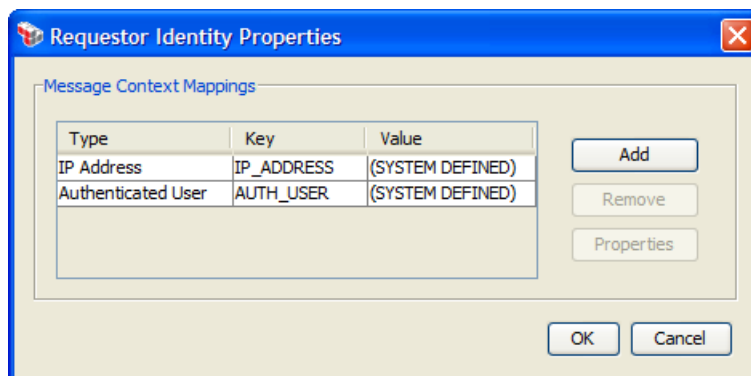



Figure 210: Requestor Identity Properties

3. Configure the properties as follows:

Table 192: Requestor Identity settings

To...	Do this...
Add a message context mapping	<ol style="list-style-type: none"> Click [Add]. The Configure Message Context Mapping dialog appears. You can add a maximum of five mappings per Capture Identity of Requestor assertion. Select the mapping Type: <ul style="list-style-type: none"> IP Address: Obtain the identity from the customer's IP address. Authenticated User: Obtain the identity from the customer's authenticated User ID. Custom Mapping: Obtain the identity using a custom mapping defined in the Key and Value fields. In the Key field, enter a string of text (no spaces) to identify your custom mapping. The key cannot be changed when using the <i>IP Address</i> or <i>Authenticated User</i> types. In the Value field, specify a context variable that holds identifying information about the requestor.  <p>The following are examples of context variables that you can use:</p> <ul style="list-style-type: none"> <code>\${request.username}</code> <code>\${request.authenticateduser}</code> a context variable created by an XPath assertion (Evaluate Request XPath or Evaluate Response XPath) a context variable created by the Evaluate Regular Expression assertion a context variable that returns information from the HTTP header <p>Tip: You can add descriptive text to the variable name to make</p>

To...	Do this...
	<p>it easier to read. For example, "High Value Customer: \${request.username}".</p> <p>The value cannot be changed when using the <i>IP Address</i> or <i>Authenticated User</i> types.</p>
Remove a message context mapping	<ol style="list-style-type: none"> 1. Select the mapping to be removed. The last mapping cannot be removed. 2. Click [Remove].
Edit a message context mapping	<ol style="list-style-type: none"> 1. Select the mapping to be edited. 2. Click [Properties]. The Configure Message Context Mapping dialog appears. 3. Edit the fields as necessary. The Key and Value fields cannot be edited for system defined types. 4. Click [OK] when done.

4. Click [**OK**] when done.

Customize SOAP Fault Response Assertion

The default behavior of the Gateway is to return a generic fault message within a SOAP envelope ("SOAP fault") when a problem occurs in a policy—for example, an assertion failure, authentication failure, routing failure, etc. The *Customize SOAP Fault Response* assertion lets you configure the SOAP fault response on a policy-by-policy basis. You can configure the level of detail returned and whether the SOAP faults are digitally signed. The following options are available for the SOAP fault detail level:

- **Drop connection:** When the policy fails, simply drop the connection without providing any response.
- **Generic SOAP fault:** Return a brief SOAP fault message.
- **Medium detail:** Return a SOAP message with more details.
- **Full detail:** Return a comprehensive SOAP fault message.
- **Template:** Lets you define your own message to be returned.

Tip: The *Customize SOAP Fault Response* assertion is intended to override the general Gateway SOAP fault response for a particular policy—it does not control whether a SOAP fault is returned but rather how the SOAP fault will appear *if* a SOAP fault should occur. If you do not need to override the general response, then this assertion is not required. For more information about the general SOAP fault response, see "SOAP Faults" on page 611.

To learn more about selecting a private key for this assertion, see *Selecting a Custom Private Key* in the *Layer 7 Policy Manager User Manual*.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Customize SOAP Fault Response as...** in the policy window and select **Fault Response Properties** or double-click the assertion in the policy window. The assertion properties are displayed.




Figure 211: Fault Response Properties

3. Select a SOAP fault level to use:

Table 193: Fault Response settings

Setting	Description
Drop Connection	Simply drops the connection when a SOAP fault or any other policy error is encountered; no error is returned.
Generic SOAP Fault	Returns a simple SOAP fault which states that a policy violation has occurred.

Setting	Description
Medium Detail	<p>Takes the "Generic SOAP Fault" setting and adds policy violation details for each assertion violated. Includes any audit detail messages generated by the failed assertions.</p> <p>At this setting, all messages of severity level "Info" or higher are included. For more information, see "Message Auditing" on page 597.</p> <p>Note: Avoid using this setting if you do not want to reveal the reasons for rejection to the requestor.</p>
Full Detail	<p>Takes the "Medium Detail" setting and adds information for each assertion that was evaluated during the request (regardless of whether it succeeded or failed).</p> <p>At this setting, messages of all severity levels are included. For more information, see "Message Auditing" on page 597.</p> <p>Note: Avoid using this setting if you do not want to reveal the reasons for rejection to the requestor.</p>
Template Fault 	<p>Allows you to define your own template response. This is the same as using the Return Template Response to Requestor assertion, except the "Response HTTP Status" is hard coded to '500' and the "Response Content Type" is always 'text/xml'. You may reference context variables within the template.</p>
[Include the policy download URL...]	<p>For all settings except for <i>Drop Connection</i>, you can specify whether the policy download URL should be included with the SOAP fault in an HTTP header, if it is required. For example, a failure of an XPath assertion would not cause the policy URL to be included, while a credential assertion such as Require HTTP Basic Credentials would include the URL.</p> <p>The default is to include this URL.</p>
[Sign SOAP Fault]	<p>For all settings except for <i>Drop Connection</i>, you can specify that the SOAP fault be digitally signed. This setting overrides the <i>soapfault.sign</i> cluster property.</p> <p>When the SOAP fault is signed, the Gateway chooses the signing key in the following order of preference (this overrides the <i>soapfault.privateKeyAlias</i> cluster property):</p> <ol style="list-style-type: none"> 1. Custom Private key: If a custom private key has been selected for the assertion, it is used for signing. For more information, see <i>Selecting a Custom Private Key in the Layer 7 Policy Manager User Manual</i>. 2. Session key: If a custom private key has <u>not</u> been selected or if the [Use default private key] option was selected on the Private Key Alias dialog, then the session key will be used. A session key exists if the policy uses a security method that relies on a session key (for example, Kerberos token profile, secure conversation, encrypted key).

Setting	Description
	<p>3. Default SSL key: If no session key exists and no custom private key was selected, then the default SSL key is used. To learn more about the default SSL key, see Private Key Properties in the <i>Layer 7 Policy Manager User Manual</i>.</p> <p>IMPORTANT: If a custom private key was selected and that key is subsequently destroyed or becomes unavailable, then the SOAP faults will <u>not</u> be signed, regardless of the Sign SOAP Fault check box. The default SSL key will not be used.</p>
Use SOAP Fault for all errors	<p>Select this check box to return a SOAP fault regardless of the error. This will display the complete set of audit detail messages, including messages that are not associated with an assertion. For example, this option can help you diagnose errors such as:</p> <ul style="list-style-type: none"> • Non-SOAP and malformed XML errors for SOAP services • Errors during WS-Security processing, such as digital signature validation errors <p>IMPORTANT: The Customize SOAP Fault assertion must be placed within a "message received" or "pre security" global policy fragment in order for the [Use SOAP Fault for all errors] option to have any effect. For more information on these policies, see "Working with Global Policy Fragments" on page 106.</p>
Use Client Fault code for all errors	<p>Select this check box to override the fault code with the client's fault code. This will result in "<faultcode>soapenv:Client</faultcode>" being returned.</p> <p>Clear this check box to use the server's fault code. This will result in "<faultcode>soapenv:Server</faultcode>" being returned. This setting is the default.</p>

- Click **[OK]** when done.

The fault response selected is added to the assertion name in the policy window, along with any custom private key selected. For example: "Customize SOAP Fault Response as Full Detail (Key: XYZ)".

SOAP Faults

The Gateway provides two different ways to handle SOAP faults: *general SOAP fault response* and *customized SOAP fault response*.

General SOAP Fault Response

The default behaviour for the Gateway is to return a standard unsigned SOAP fault when an exception occurs in a policy. You can configure the default behaviour using the "Fault Level" cluster properties. This default behaviour is used unless the SOAP fault response has been customized for a particular policy.

Customized SOAP Fault Response

If you need to override the default SOAP fault behaviour in a particular policy, use the [Customize SOAP Fault Response](#) assertion. This assertion allows you to customize the following for that policy:

- The level of detail to include in the response (overrides the *soapfault.level* cluster property)
- Whether SOAP faults should be digitally signed (overrides the *soapfault.sign* cluster property)
- If the faults are to be signed, the signing key is determined in the following order of preference (overrides the *soapfault.privateKeyAlias* cluster property):
 - a. **Custom Private key:** If a custom private key has been selected for the assertion, then that is used. For more information, see *Selecting a Custom Private Key* in the *Layer 7 Policy Manager User Manual*.
 - b. **Session key:** If a custom private key has not been selected or if the **[Use default private key]** option was selected on the Private Key Alias dialog, then the Gateway will attempt to use the session key.
 - c. **Default key:** If no session key exists and no custom private key was selected, then the default SSL key is used. To learn more about the default SSL key, see *Private Key Properties* in the *Layer 7 Policy Manager User Manual*.

Send Email Alert Assertion

The *Send Email Alert* assertion allows you to instruct the Gateway to deliver a pre-configured email message whenever the assertion is encountered in a policy.

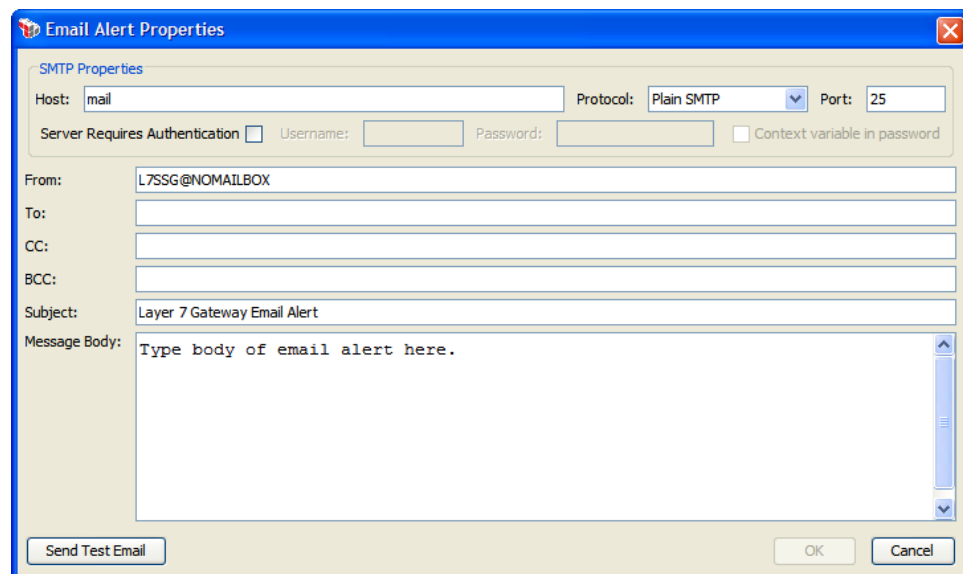
The placement of the assertion in the policy path determines when and why an email is sent. For example, the assertion could be placed in an **"At least one assertion must evaluate to true"** assertion folder after an [Evaluate Response XPath](#) assertion. If the required response message element is not found and the Evaluate Response XPath assertion fails, then the Send Email Alert assertion will execute.

Tip: The Send Email Alert assertion will fail if the outgoing email account is improperly configured. To configure the policy so that a failure of the Send Email Alert assertion does not cause a total policy failure, place the assertion in an "At least one assertion must evaluate to true" assertion folder with a [Continue Processing](#) assertion.

If you are encountering email timeouts while using this assertion, try adjusting the *mail.outConnectTimeout* and *mail.outTimeout* cluster properties.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the Email Alert Properties automatically appear; when modifying the assertion, right-click **Send Email Alert** in the policy window and select **Email Alert Properties** or double-click the assertion in the policy window. The assertion properties are displayed.



The dialog box titled "Email Alert Properties" contains the following fields and controls:










- SMTP Properties** section:
 - Host:
 - Protocol:
 - Port:
 - Server Requires Authentication: ☐
 - Username:
 - Password:
 - Context variable in password: ☐
- From:
- To:
- CC:
- BCC:
- Subject:
- Message Body:
- Buttons at the bottom: "Send Test Email", "OK", and "Cancel".

Figure 212: Email Alert Properties

- Configure the properties as follows:

Tip: Context variables may be used in the following fields for greater flexibility: **Host**, **Port**, **Username**, **Password**, **From**, **To**, **CC**, **BCC**, **Subject**.

Table 194: Email Alert settings

Setting	Description
Host 	The name of the outgoing mail server displayed as the default. Modify if necessary.
Protocol	Select the email protocol to use: Plain SMTP (default), SMTP over SSL , or SMTP with STARTTLS . The default setting should be appropriate in most instances. Consult your system administrator if you are unsure of the protocol. Note: You may need to configure trust for the SMTP server if using the "SMTP over SSL" or "SMTP with STARTTLS" protocols. For more information, see Managing Certificates in the <i>Layer 7 Policy Manager User Manual</i> .
Port 	The port used by the default mail server is displayed. Modify if necessary.
Server Requires Authentication	Select this check box if a name and password is required to log onto the email server.
Username  Password	If authentication is required, enter the user name and password.
Context variable in password	Select this check box to allow the assertion to correctly recognize context variables used in the Password field; for example, you will be using the <code>\${secpass.*}</code> context variables. For more information, see Stored Password Properties in the <i>Layer 7 Policy Manager User Manual</i> .
From 	Optionally enter a response email address.
To 	Enter the email addresses of the recipients who will receive the alert. Separate multiple addresses with a comma.
CC 	Optionally enter email addresses for CC (carbon copy) recipients. Separate multiple addresses with a comma.
BCC 	Optionally enter email addresses for BCC (blind carbon copy) recipients. Separate multiple addresses with a comma. Recipients in the 'To' and 'CC' lists will not see the recipients in the 'BCC' list.
Subject 	Enter a subject line describing the alert email.
Message Body 	Enter the body of the alert email. You may include context variables within the message, if necessary.
Send Test Email	Sends a test email to the recipients. Use this to verify that the settings are correct. Note: The [Send Test Email] button will not work if context variables have been used in the Email Alert Properties.

4. Click **[OK]** when done.

Send SNMP Trap Assertion

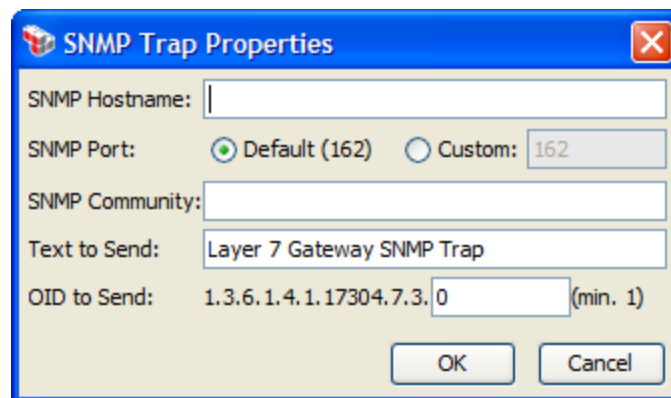
The *Send SNMP Trap* assertion allows you to instruct the Gateway to broadcast a Simple Network Management Protocol (SNMP) trap. When the Send SNMP Trap assertion is encountered in a policy execution path, an SNMP trap event will be broadcast to a predefined network address. The assertion is typically used to trigger an alert based on the result of a previous assertion.

For example, two assertions can be combined into a logical "At least one assertion must evaluate to true" assertion folder, the first assertion requiring validation, the later being the Send SNMP Trap assertion. If the assertion requiring validation fails, then the Send SNMP Trap assertion will execute, hence broadcasting the alert.

Note: There are two types of SNMP traps: v1 and v2. The Send SNMP Trap assertion sends a "v2" trap.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the SNMP Properties automatically appear; when modifying the assertion, right-click **Send SNMP Trap to...** in the policy window and select **SNMP Trap Properties** or double-click the assertion in the policy window. The assertion properties are displayed.







The image shows a dialog box titled "SNMP Trap Properties" with a blue header bar and a red close button. It contains several input fields and radio buttons. The "SNMP Hostname" field is empty. The "SNMP Port" section has two radio buttons: "Default (162)" which is selected, and "Custom:" followed by a text box containing "162". The "SNMP Community" field is empty. The "Text to Send" field contains the text "Layer 7 Gateway SNMP Trap". The "OID to Send" field contains the text "1.3.6.1.4.1.17304.7.3." followed by a text box containing "0" and a "(min. 1)" label. At the bottom right are "OK" and "Cancel" buttons.

Figure 213: SNMP Trap Properties

3. Configure the properties as follows:

Table 195: SNMP Trap settings

Setting	Description
SNMP Hostname 	Enter the network address that should receive the SNMP alert. You may reference context variables.
SNMP Port	The default SNMP trap destination port is set to "162". This is the IANA (Internet Assigned Numbers Authority) standard SNMP trap port. To configure a different port, select the Custom option and enter an alternate port number.
SNMP Community 	Optionally enter the SNMP community that should be used by the SNMP trap. You may reference context variables.
Text to Send 	Optionally enter some text to send in the SNMP trap. You may reference context variables within the trap message, if necessary.
OID to Send 	Complete the OID of the SNMP trap. This is used for identification purposes on a network. You may reference context variables. Note: If the OID entered is invalid, a value of '1' will be used instead.

4. Click **[OK]** when done.

Chapter 11:

Policy Logic Assertions

Notes: (1) Depending on which Gateway product you have installed, not all the assertions shown below may be available. See Features by Product in the *Layer 7 Policy Manager User Manual* for a list of which features are available for each product. (2) This category may also include custom-created encapsulated assertions. For more information, see "Working with Encapsulated Assertions" on page 126.

In the Policy Manager, the following assertions are available in the Policy Logic category of the [Assertions] tab:

Add Comment to Policy Assertion	618
All Assertions Must Evaluate to True Assertion	619
At Least One Assertion Must Evaluate to True Assertion	619
Compare Expression Assertion	621
Continue Processing Assertion	625
Create Routing Strategy Assertion	626
Context Variables Created by This Assertion	626
Execute Routing Strategy Assertion	630
Context Variables Created by This Assertion	630
Export Variables from Fragment Assertion	632
When Used in a Global Policy Fragment	632
Generate UUID Assertion	634
Include Policy Fragment Assertion	635
Join Variable Assertion	636
Look Up Context Variable	637
Context Variables Created by This Assertion	638
Look Up Item by Index Position Assertion	640
Look Up Item by Value Assertion	641
Manipulate Multivalued Variable Assertion	642
Map Value Assertion	644
Process Routing Strategy Result Assertion	648
Run All Assertions Concurrently Assertion	651
Technical Issues to Consider	651
Configuring the Assertion	652
Run Assertions for Each Item Assertion	653
Context Variables Created by this Assertion	654
Set Context Variable Assertion	656
Split Variable Assertion	661
Stop Processing Assertion	664

The Policy Logic assertions organize and define the structure, logic, and processing conditions for the policy.

Add Comment to Policy Assertion

The *Add Comment to Policy* assertion allows you to insert a comment at any point in a policy path or within a policy folder in the policy development window. The Add Comment to Policy assertion is useful for self-documenting complex policies, especially those with nested assertions and policy folders.

The Add Comment to Policy assertion has no effect on the runtime processing of a policy (in other words, it returns neither a "true" nor "false" when processed).

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the Comment Properties automatically appear; when modifying the assertion, right-click **Comment: ...** in the policy window and select **Comment Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

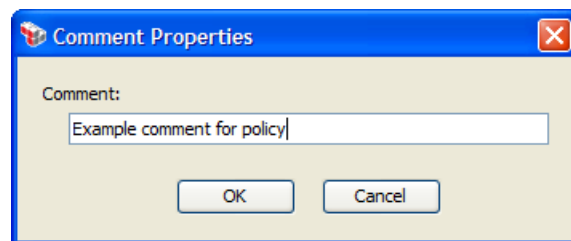


Figure 214: Comment Properties

3. Enter or edit the comment . You do not need to include the normal XML comment delimiter characters (e.g., "<!--comment text-->"), as the Policy Manager will add these for you.
4. Click [**OK**] when done.

All Assertions Must Evaluate to True Assertion

The "All assertions must evaluate to true" assertion is a folder that organizes and defines the processing conditions for the assertions that it contains and for the overall policy. When assertions are grouped into one of these folders, each successive child assertion is processed until all assertions succeed, yielding a success outcome for the folder. Processing in this assertion folder will stop when the first child assertion fails, yielding a fail outcome for the folder—and possibly the entire policy. For more information about parent and child assertions in a policy, see "Policy Organization" on page 2.

Tip: The "All assertions must evaluate to true" assertion will always succeed if there are no child assertions contained within it, or if all the child assertions have been disabled.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the policy development window, see "Adding an Assertion" on page 112. You can also right-click anywhere in the policy development window and then select **Add 'All' Folder**. This creates an assertion folder in the policy window.
 - To change the configuration of an existing assertion, proceed to step 2 below
2. Populate the folder with child assertions using any of the following methods:
 - Add an assertion by dragging and dropping it from policy window or the [Assertions] tab.
 - Remove an assertion by dragging and dropping it back into the policy window or by [deleting](#) the assertion.
3. Repeat to add additional assertion folders, if necessary.

At Least One Assertion Must Evaluate to True Assertion

The "At least one assertion must evaluate to true" assertion is a folder that organizes and defines the processing conditions for the assertions that it contains and for the overall policy. When assertions are grouped into one of these folders in the policy window, each

successive child assertion is processed until a single assertion succeeds, yielding a success outcome for the folder. If all child assertions in the folder fail, then the overall folder—and possibly the entire policy—fails. For more information about parent and child assertions in a policy, see "Policy Organization" on page 2.

Tip: If you do not want the failure of this assertion to fail the entire policy, then add a [Continue Processing](#) assertion into the assertion folder. The Continuing Processing assertion will always evaluate to true, preventing the failure of a policy due to the failure of a non-essential or conditional assertion in an "At least one assertion must evaluate to true" folder.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the policy development window, see "Adding an Assertion" on page 112. You can also right-click anywhere in the policy development window and then choose **Add "At least one" Folder**. This creates an assertion folder in the policy window.
 - To change the configuration of an existing assertion, proceed to step 2 below
2. Move the assertion folder to the appropriate position within the policy.
3. Populate the folder with child assertions using any of the following methods:
 - Add an assertion by dragging and dropping it from policy window or the [Assertions] tab.
 - Remove an assertion by dragging and dropping it back into the policy window or by [deleting](#) the assertion.

At least one child assertion is required, otherwise the "At least one..." assertion will fail.

4. Repeat to add additional assertion folders, if necessary. Be sure to [validate](#) the policy when done.

Tip: To add a new identity into the assertion folder, right-click the assertion folder in the policy development window and then choose **Add User or Group**. This adds a [Authenticate User or Group](#) assertion to the policy. Refer to "Policy Organization" on page 2 for more information.

Compare Expression Assertion

The *Compare Expression* assertion is used to compare the result of evaluating of a single context variable or an expression against a series of rules during the runtime processing of a policy. This assertion succeeds only if all the rules are satisfied.

Note: For brevity, the term "result" is used throughout the rest of this topic to denote the result of evaluating an expression.

The Compare Expression assertion can do the following:

- Ensure that the result is, or can be converted to, a specific data type.
- Perform simple comparisons against the result of another expression using standard operators such as "equals", "less than", and "contains".
- Verify that the number of values in the result falls between a designated minimum and maximum.
- Verify that the result matches a Regular Expression.
- Verify that the length of the result, expressed as a string, falls between a designated minimum and maximum number of characters.

Example 1:

The Compare Expression assertion is used with two XPath assertions (for example, the [Evaluate Request XPath](#) and/or the [Evaluate Response XPath](#) assertions) to evaluate the result of a particular attribute in a [Require SAML Token Profile](#) assertion attribute statement matches a value found in the message body. In this scenario, at least one of the XPath assertions may have a non-default variable prefix so that the second XPath assertion does not overwrite the variables set by the first.

Example 2:

The Compare Expression assertion is used to verify that exactly one node has been found by a previous Evaluate Request XPath assertion by testing that `${requestXPath.count} = 1` (use a Simple Comparison rule with the "equals" operator).

Tips: (1) If you have difficulties getting the comparisons to work, ensure that the variable names are entered correctly—for example, verify that the curly braces "{ }" are used, not parenthesis "()". To reference a variable that variable syntax `${}` must be used. (2) If your policy logic requires determining whether a context variable exists, see "Checking Existence of Context Variables" under "Context Variables" in the *Layer 7 Policy Authoring User Manual*.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the Compare Expression Properties automatically appear; when modifying the assertion, right-click **Compare Expression** in the policy window and select **Compare Expression Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

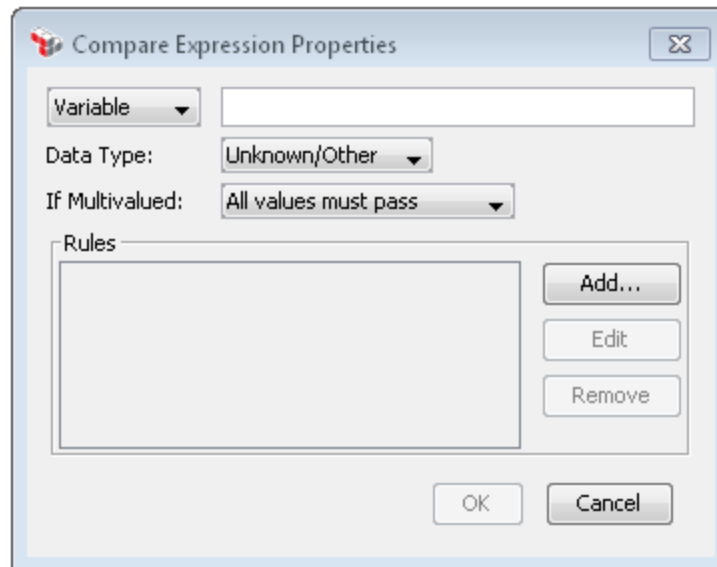



Figure 215: Compare Expression Properties

3. Select either **Variable** or **Expression** from the drop-down list.
 - If you select **Variable**, enter a single variable to be evaluated. If you enter a multivalued variable, then all values of the variable will be tested against the "Rules" list and depending on what was selected for the "If Multivalued" drop-down list.
 - If you select **Expression**, enter the expression to be evaluated. The expression can contain either a string, a single variable, multiple variables, or a string containing one or more variables. It can reference context variables, in the format `${contextVariable.name}`. This expression will be tested against the "Rules" list. 

You can include constant values in the expression. For example, the variable `${requestXpath.result}` contains the value "B". If you enter the expression `"A${requestXpath.result}C"`, the resulting value that will be compared is "ABC". However, if the Expression field contains anything but a single context variable reference (such as `${var}`), selecting a data type other than **String** is unlikely to yield useful results.

Note: Each Comparison Expression assertion in a policy can evaluate only one expression at a time, however multiple rules may be created for that expression. If you wish to evaluate several expressions, add more than one Comparison Expression assertion to the policy.

4. Select the **Data Type** for the expression. This verifies that the result is, or can be converted to a value of the selected type.

Table 196: Compare Expression assertion: Data type

Data Type	Description
Unknown/Other	The result can be in any data type. The assertion will not attempt to convert. For comparison purposes, "Unknown/Other" data types are treated as strings.
String	The result can be any string of characters. Note: If the result is an integer, decimal, or Boolean value, it will be converted to a character string for the purposes of comparison.
Integer	The result must contain only numeric characters. The assertion will fail if non-numeric characters appear in the result.
Decimal	The result must contain only numeric characters, optionally with a decimal point. The assertion will fail if non-numeric characters other than a decimal point appear in the result.
Boolean	<p>The result must be equal to "true" (not case sensitive by default) when the rule is equal to True for the assertion to succeed. If the result contains anything else—or is empty—then the assertion is falsified.</p> <p>Examples</p> <p>If the assertion is configured: <i>"Proceed if \${var.result} is equal to false"</i>:</p> <ul style="list-style-type: none"> The assertion will succeed if the result for <code>\${var.result}</code> is <i>false</i>, <i>123</i>, <i>abc</i>, etc. The assertion will fail if the result for <code>\${var.result}</code> is <i>true</i>. <p>If the assertion is configured: <i>"Proceed if \${var.result} is equal to true"</i>:</p> <ul style="list-style-type: none"> The assertion will succeed if the result for <code>\${var.result}</code> is <i>true</i>. The assertion will fail if the result for <code>\${var.result}</code> is anything but <i>true</i>.

Data Type	Description
Date/Time	<p>The expression must either be a date/time variable or the result is a timestamp or a string that can be recognized by the Gateway. If the expression cannot be converted, this assertion will fail.</p> <p>Note: If a date/time variable is specified in the expression and Date/Time is not selected as the Data Type, the Policy Manager will convert the variable into a string for comparison purposes.</p> <p>The variable will contain date/time information. These variables will behave similar to the built-in variables described in "Date/Time Variables".</p>


- If using a multivalued context variable, select how you would like the variable handled.

Table 197: Compare Expression assertion: Handling multivalued context variables

Rule	Description
All values must pass	The Comparison assertion will succeed only if every value in the multivalued variable satisfies the conditions defined in the assertion. This setting is the default.
Any value must pass	The assertion will be successful if any of the multiple values meets the condition(s).
Compare first value only	The assertion will be successful if the first value meets the condition(s).
Compare last value only	The assertion will be successful if the last value meets the condition(s).
Fail assertion	The assertion will fail if a multivalued context variable is encountered. This setting should be used when multiple values are not expected.

- Click **[Add]** to add a new rule. You are prompted to select the type of rule:

Table 198: Compare Expression assertion: Comparison rules

Rule	Description
Simple Comparison	<p>Compares the result against another value using basic operators (such as "less than", "greater than"). The value being compared to can be a number, string, or another context variable. If it is a string, you can enforce an exact case match by selecting the [Case Sensitive] check box. If the expression contains multiple values, then every value must satisfy the comparison. </p> <p>Tip: When using the Integer data type, it is best to perform the simple comparison against other integers. If comparing against a decimal value, be aware that the decimal value will be truncated into an integer</p>

Rule	Description
	<p>for comparison purposes. For example, consider the following configurations for the assertion:</p> <ul style="list-style-type: none"> • "Proceed if <code>\${var.result}</code> is a Decimal Number and is less than 45.7", and <code>\${var.result}</code> returns 45. The assertion succeeds because 45 is less than 45.7. • "Proceed if <code>\${var.result}</code> is an Integer and is less than 45.7", and <code>\${var.result}</code> returns 45. The assertion fails because 45 is not less than 45 (the decimal '45.7' is truncated to '45' as the data type is Integer and the comparison is between integers).
Number of Values	Verifies that the number of values in the result falls between the stated minimum and maximum. For expressions containing a single value, you can use Min=1, Max=1 to indicate that a value is "required", or Min=0, Max=1 to indicate that a value is "optional". Check the "Unlimited" box to set no limit to the maximum value.
Regular Expression	<p>Matches the result against a constant regular expression. If the expression contains multiple values, then every value must match the regular expression.</p> <p>Tip: For a more powerful application of regular expressions in a policy, use the "Evaluate Regular Expression Assertion" on page 449.</p>
String Length	<p>Verifies that the length of the result, expressed as a string, falls between the stated minimum and maximum. If the expression contains multiple values, then every value must satisfy the length constraint.</p> <p>For example, if the expression result is "abcd" and a String Length rule stipulates that the minimum length is 5, then the assertion is falsified.</p>

7. Click **[OK]** when done.

Continue Processing Assertion

The *Continue Processing* assertion is a placeholder assertion that will always yield a successful or true processing result. When used in an "At least one assertion must evaluate to true" assertion folder, the Continue Processing assertion will ensure that the incidental failure of a non-essential assertion within the same folder will not cause an overall policy failure.

Note: The Continue Processing assertion is intended for use within an "At least one assertion must evaluate to true" assertion folder. It serves no purpose if used elsewhere.

Using the Assertion

- Add the assertion to an "At least one assertion must evaluate to true" assertion folder the policy development window. For more information, see "Adding an Assertion" on page 112.

The assertion is added to the policy window; no further configuration is required.

Create Routing Strategy Assertion

The *Create Routing Strategy* assertion is used to create routing strategies that form the foundation of the Gateway's dynamic routing capabilities.

In this assertion, you configure a route list that contains a list of route destinations. These destinations are usually multivalued variables that store a list of the possible servers at the back end. The assertion then parses the destination servers and creates a route list that is stored in another variable that is used by the *Execute Routing Strategy* and *Process Routing Strategy Result* assertions.

Note: The *Create Routing Strategy* assertion must precede *Execute Routing Strategy* and *Process Routing Strategy Result* assertions. Before creating a route, see *Working with Dynamic Routing Strategy*.

Context Variables Created by This Assertion

The *Create Routing Strategy* assertion sets the following context variables with the query results.

Table 199: Context variables created by the *Create Routing Strategy* assertion

Variable	Description
<code>\${strategy}.routeList</code>	<p>This returns the list of routes from the Route List table in a multivalued context variable, where <code><strategy></code> is the value entered in the Routing Strategy Prefix field. You can access items within this multivalued variable using the <code>\${strategy}.routeList.<index></code> syntax. This variable is set only for outbound messages.</p> <p><i>Example:</i> Using the sample route data in Figure 1, assume that the route specified by the variable <code>\${jdbcQuery.servers}</code> resolves to: <i>server 1</i>, <i>server 2</i>. Based on the default Routing Strategy Prefix strategy, the multivalued variable <code>\${strategy}.routeList</code> will be created that contains the following values.</p> <pre>invalid server1</pre>

Variable	Description
	<i>server2</i>
<code>\${<strategy>}</code>	This context variable is created using the value entered in the Routing Strategy Prefix field. For example, based on the default value strategy , the context variable that is created is named <code>\${strategy}</code> . This context variable returns the chosen routing strategy and is used by the Execute Routing Strategy and Process Routing Strategy Result assertions. This variable is set only for outbound messages.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click "**Create Routing Strategy...**" in the policy window and then select **Create Routing Strategy Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

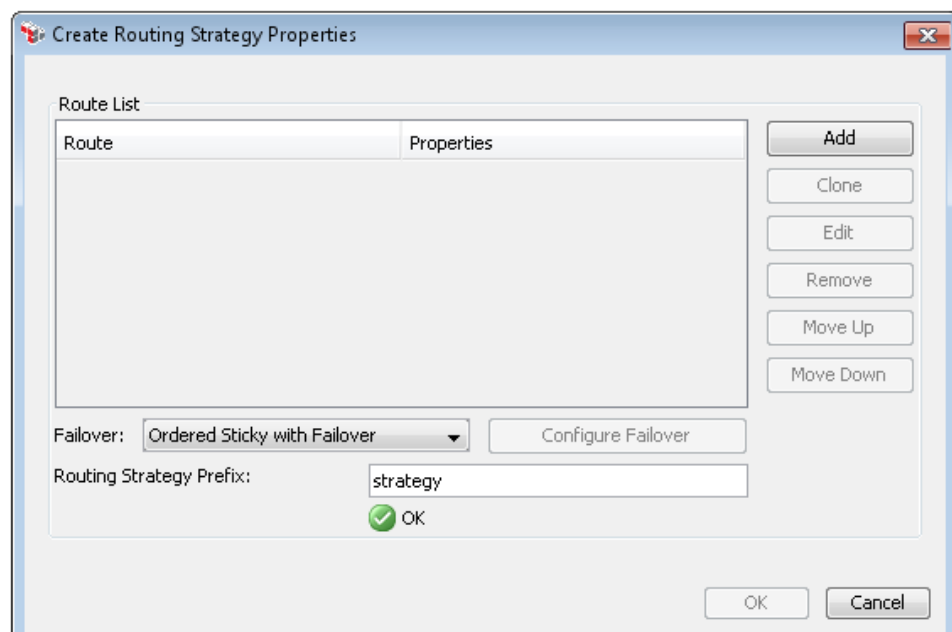


Figure 216: Create Routing Strategy Properties

- Choose a task to perform.

Table 200: Configuring the Route List

To...	Do this...
Add a new Route	<ol style="list-style-type: none"> 1. Click [Add]. The New Route dialog is displayed. 2. Enter a route name. 3. Configure the route properties. For more information, see Table 3. 4. Click [OK].
Clone an existing Route	<ol style="list-style-type: none"> 1. Select route to clone. 2. Click [Clone]. The Clone Route dialog is displayed. 3. Modify the route name and properties as required. For more information, see Table 3. 4. Click [OK].
Edit an existing Route	<ol style="list-style-type: none"> 1. Select route to edit. 2. Click [Edit]. The Edit Route dialog is displayed. 3. Modify the route name and properties as required. For more information, see Table 3. 4. Click [OK].
Remove a Route	<ol style="list-style-type: none"> 1. Select route to remove. 2. Click [Remove]. You are prompted to confirm. 3. Click [Remove] to confirm.
Move Up	Move the selected route up one line.
Move Down	Move the selected route down one line.
Failover	<p>Choose a failover from the drop-down list.</p> <ul style="list-style-type: none"> • Ordered Sticky with Failover: The Gateway sends service messages to the first route in the list until the route does not respond (fails). When this occurs, the next route in the list is used. <p>Tip: The cluster property <code>io.failoverServerRetryDelay</code> controls the delay before the Gateway retries a failed server. The default is to wait 15 minutes when using the "Ordered Sticky with Failover" strategy.</p> • Random Sticky with Failover: The Gateway chooses a route at random in the beginning of each session and uses it for the duration of the session. If the chosen route fails, another route is randomly selected. • Round-Robin: The Gateway rotates through the route list sequentially on a request-by-request basis (round-robin), until a valid route is found. If a valid route is not found, and the end of the route list is reached, the cycle stops. See "Execute Routing Strategy Assertion" on page 630 for more information.

To...	Do this...
	<p>Tip: The cluster property <i>io.failoverServerRetryDelay</i> controls the delay before the Gateway retries a failed server. The default is to wait 5 minutes when using the "Round Robin" strategy.</p>
Configure Failover	This option is only available with a custom failover.
Routing Strategy Prefix	<p>Enter a prefix that will serve two purposes:</p> <ul style="list-style-type: none"> The value entered here will be added as a prefix to the <code>\${<strategy>.routelist}</code> (shorthand for <code>"\${<routingStrategyPrefix>.routelist}"</code>) variable that is created by this assertion. The value entered here will also be used to create its own context variable that will be used to store the chosen strategy. <p>The default is strategy.</p> <p>For more information these two variables, see "Context Variables Created by This Assertion".</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

Table 201: Configuring route properties

To...	Do this...
Add a route property	<ol style="list-style-type: none"> Enter a name in the Route field. Click [Add]. The New Property dialog appears. Complete the New Property dialog box as follows. <ul style="list-style-type: none"> Property Name: Enter the Property Name. Value: Enter the value associated with the Property Name.
Edit a route property value	<ol style="list-style-type: none"> Highlight the route in the Route Property and click [Edit]. The Edit Property dialog appears. Edit the value in the Value field of the Edit Property dialog. This value will appear in the Properties column of the Route List.
Delete a route property	<ol style="list-style-type: none"> Highlight the route to delete in the Route Property and then click [Delete]. The route disappears from the Route Properties list in

To...	Do this...
	the New Route dialog as well as from the Properties column in the Route List.

- Click **[OK]** when done.

Execute Routing Strategy Assertion

The *Execute Routing Strategy* assertion is the second part of the Gateway's dynamic routing capability. It takes the chosen routing strategy from the `${<strategyPrefix>}` variable populated by the [Create Routing Strategy](#) assertion, acquires a route destination and places it in a new route variable.

Note: Before you can execute a routing strategy, create a route first, through the [Create Routing Strategy Assertion](#). See *Working with Dynamic Routing Strategy* before creating a route. The *Create Routing Strategy* assertion must precede *Execute Routing Strategy*.

Context Variables Created by This Assertion

The *Execute Routing Strategy* assertion sets the following context variables:

Table 202: Context variables created by the *Execute Routing Strategy* assertion

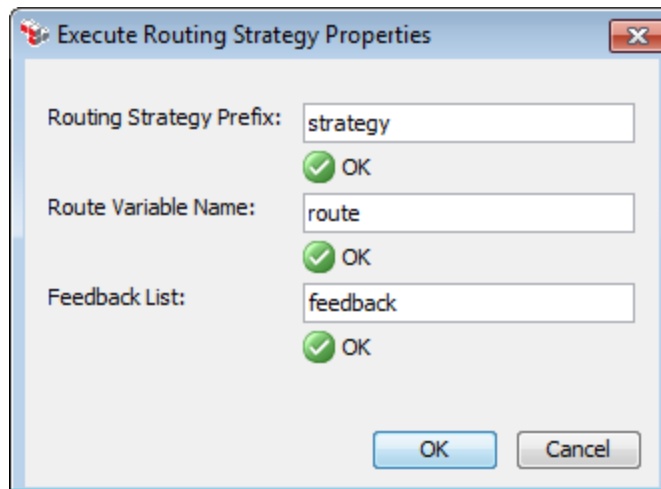
Variable	Description
<code>\${<route>}</code>	Returns the chosen route from the strategy.
<code>\${<feedbackList>}</code>	Returns feedback information for the routing.

Tip: Even though both the `${<route>}` and `${<feedbackList>.<index>.<route>}` variables return the selected route from the strategy, they serve different purposes. The `<feedbackList>` variable may be referenced outside the loop and stored elsewhere externally, while the `<route>` variable is set each time the [Execute Routing Strategy](#) assertion is run. The `<feedback>` variable serves as an audit that can be analyzed later for decision-making purposes on routes.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click "**Execute Routing Strategy...**" in the policy window and then select **Execute Routing Strategy Properties** or double-click the assertion in the policy

window. The assertion properties are displayed.



The dialog box titled "Execute Routing Strategy Properties" contains three input fields, each with a green checkmark and the text "OK" below it:




- Routing Strategy Prefix:
- Route Variable Name:
- Feedback List:

At the bottom right are "OK" and "Cancel" buttons.

Figure 217: Execute Routing Strategy Properties

- Configure the properties as follows.

Table 203: Execute Routing Strategy assertion settings

Setting	Description
Routing Strategy Prefix 	Specify the context variable that is storing the chosen routing strategy. This should match the "Routing Strategy Prefix" entered in the Create Routing Strategy assertion. Default: strategy
Route Variable Name 	Specify the context variable that will store the chosen route from the strategy. Default: route The default results in the variable <code>\${route}</code> that can be used to retrieve the route destination. This variable is set only for outbound messages.
Feedback List 	Specify the context variable that will store the feedback from the routing. This feedback includes the Current Route information for both successful or failed routing attempts. Feedback is collected after each routing attempt and added to the Feedback List context variable only when the Process Routing Strategy Result assertion is run. Default: feedback The default results in the multivalued variable <code>\${feedback}</code> . This variable is set by both inbound and outbound messages.

- Click **[OK]** when done.

Export Variables from Fragment Assertion

The *Export Variable from Fragment* assertion is used to flag context variables created within a policy fragment as being "in use". This will make the variables available to whichever policy includes the fragment. For example, if XPath-based assertions ([Evaluate Request XPath](#) or [Evaluate Response XPath](#)) are used in a fragment and the XPath context variables are referenced in the including policy, the Export Variables from Fragment assertion ensures that the variables are created and made available.

Why do the context variables need to be "made available"? For maximum performance, the Gateway creates custom context variables only when it detects that the variables will be used by another assertion. In a policy fragment, it is not possible to determine ahead of time whether the variables within the fragment will be used in a policy that includes the fragment, so they are not created. The Export Variables from Fragment assertion allows you to flag specific variables as being "in use".

Add this assertion to a [policy fragment](#), after the assertion that creates the context variables.

Tip: The Export Variable from Fragment assertion is only used in a policy fragment and only when an XPath-based assertion is also present. It provides no additional functionality if used outside of a fragment because XPath context variables are available automatically in that scenario.

When Used in a Global Policy Fragment

When the Export Variables from Fragment assertion is used in a [global policy fragment](#), it will automatically copy the values of the designated variable(s) up to the parent policy, prefixed with "request.shared". This will make it easier to use the values in (for example) the audit sink policy.

Using Figure 218 below as an example, the values from the two selected variables would be copied over to these new context variables if this assertion was used in a global policy fragment:

```
${request.shared.requestXPath.element}
${request.shared.requestXPath.result}
```

Example: Accessing variables from global policy fragment in audit sink policy

The following example illustrates how to access the "message-received" variable from the audit sink policy:

1. Create a "message-received" global policy fragment containing these assertions:

Audit Messages in Policy (WARNING)

Set Context Variable varMessageReceived as String to: Hello from message-received global policy

Export Variables from Fragment: varMessageReceived

2. Create a custom audit sink policy with this assertion:

Add Audit Details: log, custom logger "audit":

"varMessageReceived=\${request.shared.varMessageReceived}"

3. Consume any service.
4. The Gateway log will contain this line:

varMessageReceived=Hello from message-received global policy

Using the Assertion

1. Do one of the following:
 - To add the assertion, [create a policy fragment](#) or [edit an existing policy fragment](#) and then [add the assertion](#) to the policy development window.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Export Variables from Fragment** in the policy window and select **Export Variables from Fragment Properties**. The assertion properties are displayed.

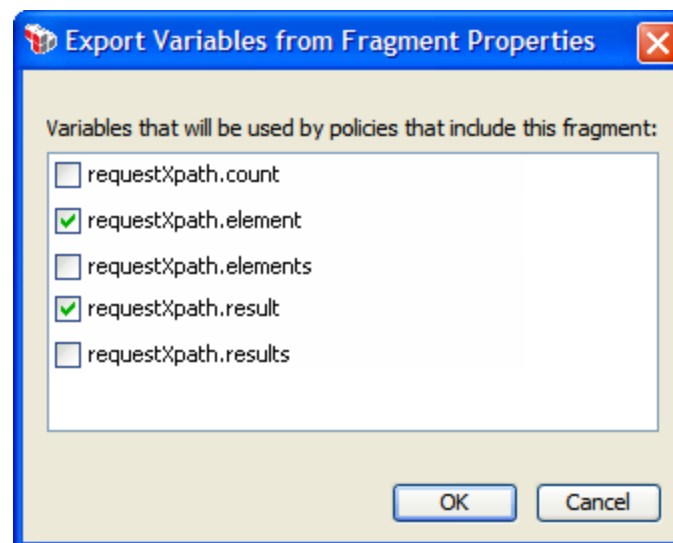


Figure 218: Export Variables from Fragment Properties

3. Select each variable to be made available to policies outside of the fragment.

For more information about the variables, see "Evaluate Request XPath Assertion" on page 458 or "Evaluate Response XPath Assertion" on page 461.

Tip: If the variable prefix is modified in the XPath assertions, the new names are reflected in Figure 218. However, variables that were previously selected will still appear under the old variable names. This will cause validation warnings about variables not being defined. Should this happen, return to the Export Variables from Fragment Properties and re-select the correct updated variables.

4. Click **[OK]** when done.

Generate UUID Assertion

The *Generate UUID* assertion creates any number of universally unique identifiers (UUIDs) and stores them in a context variable. If a single UUID is generated, it is stored in a single-value context variable, if multiple UUIDs are generated, then they are stored in a multivalued context variable. For more information, see Working with Multivalued Context Variables in the *Layer 7 Policy Manager User Manual*.

The UUIDs generated are RFC4122 compliant.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. This assertion contains default settings that are appropriate for most instances. To change any of the settings, right-click **Generate UUID** in the policy window and select **Generate UUID** or double-click the assertion in the policy window. The assertion properties are displayed.

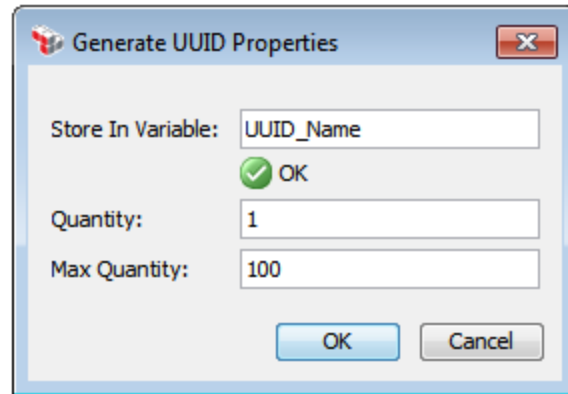



Figure 219: Generate UUID Properties

3. Configure the properties as follows:

Table 204: Generate UUID settings

Setting	Description
Store In Variable	Enter the name of the variable to store the UUID in. Tip: You do not need to enclose the variable name with the wrapper characters "\$()".
Quantity 	Enter the number of UUIDs to generate. The minimum value is 1, the maximum value is the amount listed in the Max Quantity field, below. You may specify a context variable.
Max Quantity	Enter the maximum number of UUIDs to generate for the variable. The default value is 100.

4. Click [OK] when done.

Include Policy Fragment Assertion

The *Include Policy Fragment* assertion is used to [add a policy fragment](#) to a service policy.

To learn more about how policy fragments can benefit you, see [Policy Fragments](#). To learn how to create policy fragments, see "Creating a Policy" on page 21.

Using the Assertion

1. Add the assertion to the policy development window. For more information, see [Adding an Assertion](#).
2. Select the fragment to be inserted from the **Select Policy Fragment to Include** dialog that appears.

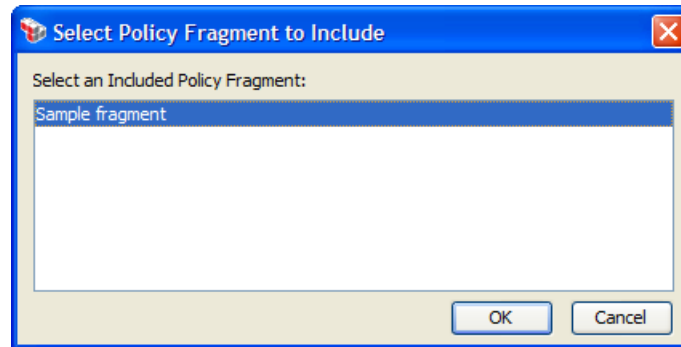


Figure 220: Selecting a policy fragment to include

3. Click **[OK]** to add the fragment to the policy. Use the Assertions Tool Bar to reposition the fragment if necessary

Join Variable Assertion

The *Join Variable* assertion combines the values in a multivalued context variable into a single-value context variable by concatenating each value with a user-defined string.

Example:

The input variable "`${varIn}`" is a multivalued context variable that contains the values "one", "two", "three". After joining this variable into "`${varOut}`" with a delimiter of '+', you now have a single-value context variable with the value "one+two+three".

Tip: The join effect can also be achieved by using the syntax "`${variable};`", where the value after the ';' character is the delimiter string. For example, "`${varIn};`" will also result in "one+two+three" using the example above. For more information, see "Concatenation Options during Interpolation" under Working with Multivalued Context Variables in the *Layer 7 Policy Manager User Manual*.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the Split Variable Properties automatically appear; when modifying the assertion, right-click **Join variable <source> into <target>...** in the policy window and select **Join Variable Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

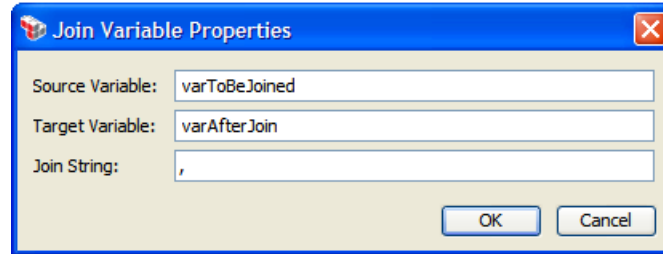




Figure 221: Join Variable Properties

3. Configure the properties as follows:

Table 205: Join Variable settings

Setting	Description
Source Variable 	Enter the context variable containing the source values to be joined. This must be a multivalued context variable.
Target Variable 	Enter the context variable that will hold the results of the join. If this variable does not already exist, it will be created.
Join Value	Enter any string to be used as a delimiter for the joined values. The default is a comma (',').

4. Click **[OK]** when done.

Look Up Context Variable

The *Look Up Context Variable* assertion is used to dynamically look up the value of a context variable and then store the value in another context variable. You specify an expression, which can contain one or more context variables plus static text. During runtime, this expression is resolved and its value is placed in another context variable, which can then be used later in the policy.

Example:

Consider the following variables and their values:

- `foo = bar`
- `ingredient = chocolate`
- `chocolate.bar = goodness`

In the Look Up Context Variable assertion, enter the following expression:

```
${ingredient}.${foo}
```

And keep the default prefix: *lookup*

The expression will resolve to **chocolate.bar**, which becomes the variable to be looked up by the assertion. The variable `chocolate.bar` contains the value "goodness", so this is placed in the assertion variable `lookup.output`. You can then reference `${lookup.output}` elsewhere in the policy to retrieve the value "goodness".

Context Variables Created by This Assertion

The Look Up Context Variable assertion sets the following context variables. **Note:** The default `<prefix>` is "lookup" and can be changed in the assertion properties (Figure 1).

Table 206: Context variables created by the Look Up Context Variable assertion

Variable	Description
<code><prefix>.found</code>	Contains true if the expression was found, otherwise contains false .
<code><prefix>.multivalued</code>	Contains true if the value found in the expression is multivalued. Contains false if the value is not multivalued or if the expression is not found.
<code><prefix>.output</code>	Contains the value of the expression. For more information on what is populated into this variable, see the example in the topic introduction.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the **Look Up Context Variable Properties** automatically appears. When modifying the assertion, right-click **<target>: Look Up Context Variable** in the policy window and select **Look Up Context Variable Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

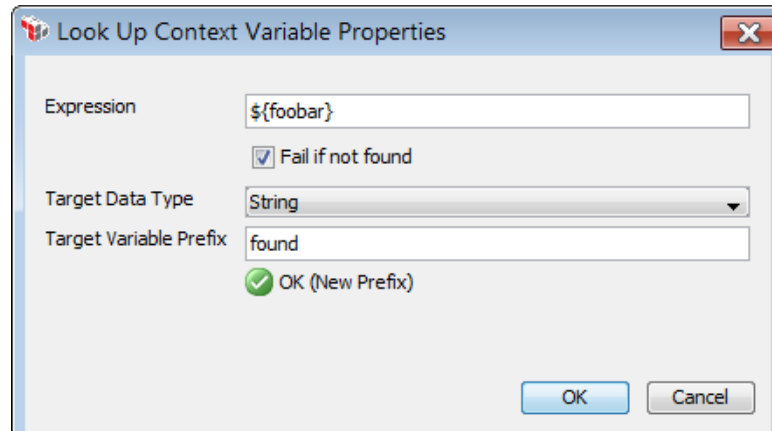




Figure 222: Look Up Context Variable Properties

3. Configure the properties as follows:

Table 207: Look Up Context Variable settings

Setting	Description
Expression 	<p>Specify the source context variables to evaluate. You can enter more than one variable but note the following limitations:</p> <ul style="list-style-type: none"> Nested context variables are not supported (for example, <code>\${foo.\${bar}}</code>). All variables must be on the same level (for example, <code>\${foo}.\${bar}</code>). <p>When referencing multivalued context variables, you can use indexing to extract a single value. For more information, see "Indexing Options during Interpolation" in Working with Multivalued Context Variables in the <i>Layer 7 Policy Manager User Manual</i>.</p>
Fail if not found	<p>Select this check box to fail the assertion if the variable in the Expression does not exist. This setting is the default.</p> <p>Clear this check box to not fail the assertion if the variable is not found.</p>
Target Data Type	<p>Choose the data type for the target value returned: String, Date/Time, X.509 Certificate, XML Element, or Message.</p>
Target Variable Prefix 	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>The default variable prefix is lookup.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

4. Click **[OK]** when done.

Look Up Item by Index Position Assertion

The *Look Up Item by Index Position* assertion looks up an item based on index position in a multivalued context variable and then stores the value in another context variable.

This assertion is designed to work with the "Look Up Item by Value Assertion" on page 641, which creates a context variable containing the index position.

Example:

The (Non-SOAP) [Verify XML Element](#) assertion creates several context variables. Using the [Look Up Item by Value](#) assertion, you find that a particular element is in the third position in the *elementsVerified* context variable. With this knowledge, you can use the Look Up Item by Index Position assertion to look up the signature method that was used (in the *signatureMethodUris* variable), as well as the digest method employed (in the *digestMethodUris* variable).

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Look Up Item by Index Position** in the policy window and select **Look Up Item by Index Position Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

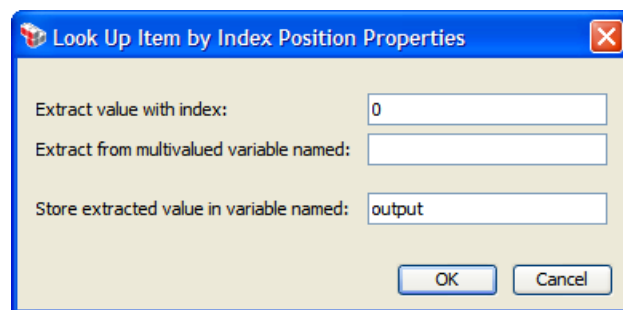





Figure 223: Look Up Item by Index Position Properties

3. Enter the index position to interpolate a previously-set variable that contains the index, such as `${index}`. To extract the first item in the variable, enter **0**, to extract the second item, enter **1**, etc. 

4. Enter the name of the context variable from which to extract. 
5. Enter the name of the context variable that will hold the extracted value. The default name "**output**" is offered. **Tip:** You do not need to enclose the variable name within the wrapper characters "\${ }". 
6. Click [**OK**].

Look Up Item by Value Assertion

The *Look Up Item by Value* assertion scans a multivalued context variable to find a specified value. If the value is found, its index position is returned.

Example:

`${Variable1}` contains the value you are searching for: {"book"}

`${Variable2}` contains the strings: {"magazine", "book", "newspaper"}

`${Variable3}` will contain the lookup result, which is {1}

Tip: Remember, multivalued context variables use zero-based positioning. Thus, the first item in the list is always item "0".

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Look Up Item by Value** in the policy window and select **Look Up Item by Value Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

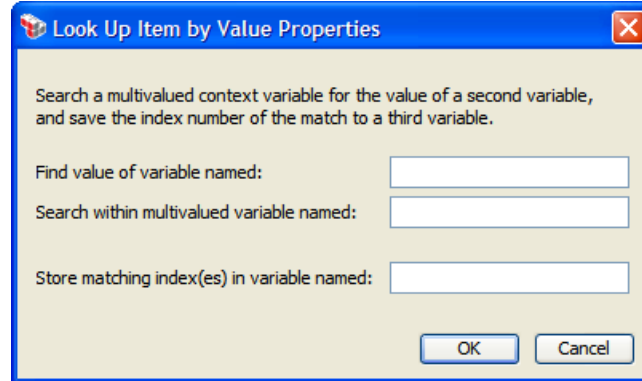





Figure 224: Look Up Item by Value Properties

3. Enter a context variable containing the value you are seeking. 
4. Enter the multivalued context variable that will be scanned. 
5. Enter the name of the context variable that will hold the results of the lookup. If this variable does not already exist, it will be created. 
6. Click **[OK]**.

Manipulate Multivalued Variable Assertion

The *Manipulate Multivalued Variable* assertion is used to both create new Multivalued Variables and to append new values to existing Multivalued Variables.

Multivalued variables can hold values of different types. This assertion does not support all available types which may exist at runtime. From the [Set Context Variable](#) assertion, only the String, Integer and Date/Time are supported.

The following Java types are supported if you are writing custom Java code via custom assertions:

```
java.lang.String
java.lang.Integer
java.lang.Double
java.lang.Float
java.lang.Boolean
java.util.Date (and any subclass; for example, Timestamp.)
```

Assertions may also set variables that do not belong to any of the types listed above. If the value being appended is not one of the supported Java types, then the assertion will fail.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click "**Manipulate Multivalued Variable...**" in the policy window and then select **Manipulate Multivalued Variable Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

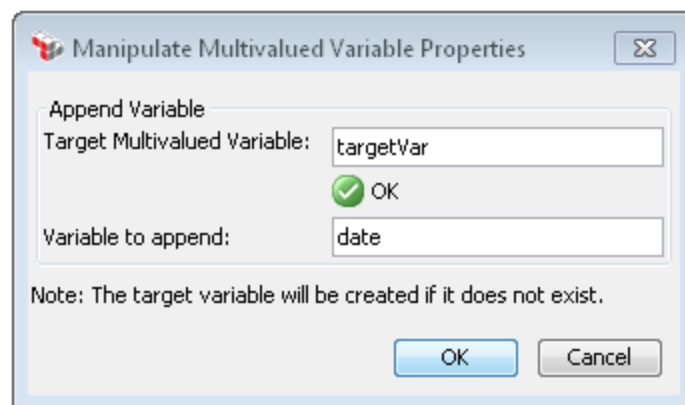




Figure 225: Manipulate Multivalued Variable Properties

- Configure the properties as follows:

Table 208: Manipulate Multivalued Variable settings

Setting	Description
Target Multivalued Variable 	Specify the name of the multivalued context variable. If the variable does not already exist, it will be created. Notes: (1) The target variable may be an existing multivalued variable. However, not all existing multivalued variables can be modified after they are created. If an attempt is made to update such a variable the assertion will fail. Any variables created by this assertion can be modified by this assertion. (2) Be sure to observe the naming rules described under "Context Variable Naming Rules" in the <i>Layer 7 Policy Manager User Manual</i> .
Variable to append 	Specify the name of the variable to append to the multivalued variable. Only a single variable can be referenced. The reference can include a variable syntax such as "\${myvar}" or "myvar", or an array syntax such as \${myVar[0]} or "myVar[0]". The variable itself can be a multivalued variable.

- Click **[OK]** when done.

Map Value Assertion

The *Map Value* assertion is used to map values in a policy. It matches a value against a number of regular expressions to produce an output value that is stored in a context variable. The result is taken from the first line that matches the value.

This assertion is particularly useful in a "message-received" [global policy fragment](#) for mapping URLs to published services, when the mapping is more complex than can be conveniently expressed using the built-in mapping support.

Tip: You can view the Map Value assertion as providing the functionality of a switch statement, or as a means of defining a table with input and output values.

Policy Example

The Map Value assertion can be used in a "message-received" [global policy fragment](#) to override the CA API Gateway's service resolution logic with your own. Assume the following settings in the assertion properties:

- *Value to Map:* **`${request.url.query}`**
- *Mappings:*
 - Pattern:* **`east`**; *Result:* **`/east_uri`**
 - Pattern:* **`west`**; *Result:* **`/west_uri`**
- *Output variable:* **`uri`**

The assertion will be interpreted as follows:

1. Examine the value: `${request.url.query}`.
2. If this value matches the regular expression "east", then set the output variable "url" to "east_url".
3. Otherwise, if this value matches the regular expression "west", then set the output variable "url" to "west_url".
4. If neither regular expression is matched, then the assertion fails and the output variable "url" is not set.

When used in a "message-received" global policy fragment, the fragment might resemble the following:

[At least one assertion must evaluate to true](#)

[All assertions must evaluate to true](#)

Map value from `${request.url.query}`, using map { "east" => "/east_uri", "west" => "/west_uri" }, output to `${uri}`

Resolve service with URI `${uri}`

Continue processing

At runtime, this is how various values will be interpreted:

?east -> /east_uri

?direction=west -> /west_uri (substring match)

?beast -> /east_uri (substring match; example regex does not distinguish word boundaries)

?direction=west&otherdirection=east -> /east_uri (first match wins--the matches are attempted from top row in table to bottom until one succeeds)

?direction=north -> assertion FAILED (no matching row)

Capture Group Pseudo-Variables

The "pseudo variables" `${0}`, `${1}`, `${2}`, etc., are available while the output value is being set after a successful match. These variables contain the capture groups from the matching regular expression. One use for these variables would be to translate a REST-style call (where the target object name is encoded into the URL path) into a SOAP-style call (where the target object name is identified in an XML payload).

Example #1:

The following is a simple example of pseudo-variables extracted from a phone number. The value to map: **`${request.url.path}`**.

- Value to map: **`${phone}`**
- Pattern: `(\d{3})-(\d{3})-(\d{4})`

Note that for this example, the result and output variable does not matter because the intent is to describe the capture group pseudo-variables.

During run time, `${phone}` contains the value **800-555-1234** and is successfully matched against the pattern above. This will set the following pseudo-variables:

`${0}` is set to the entire string matched by the regular expression – "800-555-1234"

`${1}` is set to the first capture group – "800"

`${2}` is set to the second capture group – "555"

`${3}` is set to the third capture group – "1234"

Note: The \${0} to \${3} pseudo-variables only exist while the mapping result is being set. They are not available for use in subsequent assertions in the policy.

Example #2:

The following is a more complex example showing pseudo-variables in use in a mapping:

- **Pattern:** `^/users/(\w+)/paystub/(\d+)`
- **Result (on one line):**

```
<info><action>getPaystub</action><user>${1}</user>
<stubid>${2}<stubid></info>
```
- **Pattern:** `^/users/(\w+)/vacations/(\d+)/(\d+)`
- **Result (on one line):**

```
<info><action>getVacation<action><user>${1}</user><year>${2}
</year><month>${3}</month></info>
```

During runtime, if a request arrives at a URL similar to:

```
/users/bob/paystub/123
```

The output variable will be set to :

```
<info><action>getPaystub</action><user>bob</user><stubid>123</stubid></info>
```

The pseudo-variables used here are:

\${1} is set to the first capture group - "bob"
\${2} is set to the second capture group - "123"

Similarly, if a request arrives at a URL similar to:

```
/users/sue/vacations/2012/3
```

The output variable will be set to:

```
<info><action>getVacation</action><user>sue</user><year>2012</year><month>3</month></info>
```

The pseudo-variables used here are:

\${1} is set to the first capture group - "sue"
\${2} is set to the second capture group - "2012"
\${3} is set to the third capture group - "3"

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click **Map Value** in the policy window and select **Map Value Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

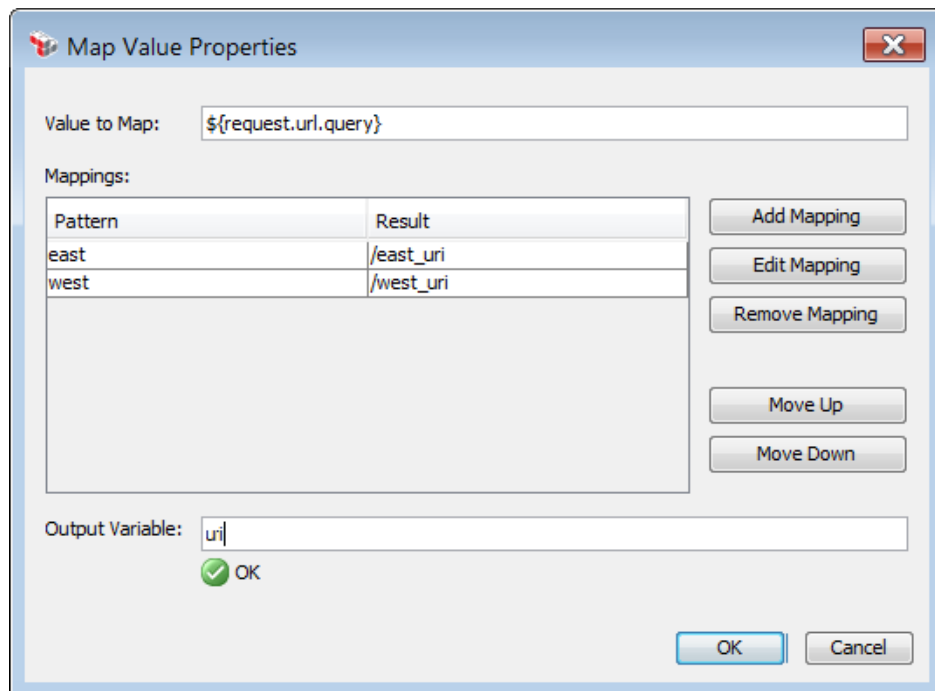





Figure 226: Map Value Properties

- Configure the dialog as follows:

Table 209: Map Value settings

Setting	Description
Value to Map 	Enter an expression to be compared against the mappings defined below. You may enter a literal expression or a context variable.
Mappings 	Define the list of regular expressions to be mapped: To add a mapping: <ol style="list-style-type: none"> Click [Add Mapping]. Enter a regular expression Pattern to be compared against

Setting	Description
	<p>the value specified above. You may reference context variables.</p> <p>Note: If a referenced context variable does not exist (for example, due to a misspelling), it will by default be treated as an empty string, which could result in a pattern that matches any input. To catch such issues during policy development, set the cluster property <i>template.strictMode</i> to true.</p> <ol style="list-style-type: none"> Enter a Result that will be placed in the output variable if the mapping is successful. You may reference context variables, as well as special regex capture group pseudo-variables. For details, see "Capture Group Pseudo-Variables" above. Click [OK]. <p>To edit a mapping:</p> <ol style="list-style-type: none"> Select the mapping to edit. Click [Edit Mapping]. Modify the pattern or result as required. Click [OK]. <p>To remove a mapping:</p> <ol style="list-style-type: none"> Select the mapping to remove. Click [Remove Mapping]. The line is removed immediately. <p>To change the order of the mappings:</p> <ol style="list-style-type: none"> Select a mapping. Click [Move Up] or [Move Down] as required. The result from the first matching line is placed in the output variable.
Output Variable 	<p>Specify a single context variable that will contain the result from a successful match.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>

- Click **[OK]**.

Process Routing Strategy Result Assertion

The *Process Routing Strategy Result* assertion is used to process the results of a route and generate feedback status of the strategy. This assertion is part of the Gateway's dynamic routing capabilities.

Use of this assertion is optional, as some strategies do not require feedback and may ignore the Feedback List context variable set by this assertion. However, feedback may still be useful elsewhere in the policy.

Note: Before you can process the result of a routing strategy, create a route first, through the [Create Routing Strategy Assertion](#). See [Working with Dynamic Routing Strategy](#) before creating a route. The Create Routing Strategy assertion must precede Execute Routing Strategy and Process Routing Strategy Result assertions.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click "**Process Routing Strategy...**" in the policy window and then select **Process Routing Strategy Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

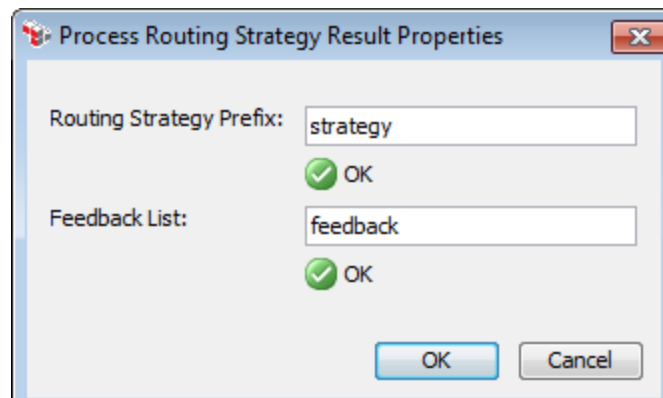


Figure 227: Process Routing Strategy Result Properties

3. Configure the properties as follows.

Table 210: Process Routing Strategy Result Properties

Variable	Description
Routing Strategy Prefix	Specify the context variable that is storing the chosen routing strategy. This should match the "Routing Strategy Prefix" entered in the Create Routing Strategy and Execute Routing Strategy assertions. Default: strategy
Feedback List	Specify the context variable that will store the feedback list. This should match the "Feedback List" entered in the Execute Routing Strategy assertion. The Feedback List variable will contain feedback data, including the selected route, reason code returned, latency of the assertion, and status.

Variable	Description
	<p>Default: feedback</p> <p><i>Example:</i> Based on the default value, you can retrieve the first feedback data with these variables:</p> <ul style="list-style-type: none"> • <code>\${feedback.1.route}</code> – returns the first route on the Feedback List. • <code>\${feedback.1.reasonCode}</code> – returns the reason code sent back by the routing assertion. • <code>\${feedback.1.latency}</code> – returns the latency of the assertion. • <code>\${feedback.1.status}</code> – returns the status of the route ("-1" = fail; "0" = success).

Based on the Feedback List variable, the Process Routing Strategy Result assertion will also populate the following context variables.

Table 211: Feedback information for the current route

Variable	Direction	Description
<code>\${<feedbackList>.current.reasonCode}</code>	Inbound	<p>Returns the reason code of the current route. This value is sent back by the routing assertion.</p> <p>Tip: This currently returns the same value as <code>\${httpRouting.reasonCode}</code>, which is created by the Route via HTTP(S) assertion.</p>
<code>\${<feedbackList>.current.status}</code>	Inbound	<p>Returns the status of the current route as an integer.</p> <p>Note: This variable is initially created by the Set Context Variable assertion. For more information, see Working with Dynamic Routing.</p>
<code>\${<feedbackList>.current.latency}</code>	Inbound	<p>Returns the latency for routing.</p> <p>Note: This variable is initially created by the Set Context Variable assertion. For more information, see Working with Dynamic Routing.</p>
<code>\${<feedbackList>.current.route}</code>	Inbound	<p>Returns the current route in the loop that is being monitored for feedback by the Process Routing Strategy Result assertion.</p> <p>Note: This variable is initially created by the Execute Routing Strategy assertion. This variable returns the same route as <code>\${route}</code>, but can be reset and overwritten if necessary.</p>

Note: The variables in Table 211 contain feedback for the current route. This differs from using indexing on the `${<feedback>}` multivalued context variable, which can access feedback information on routes other than the current route (for example, for the first feedback data based on the example shown earlier).

4. Click **[OK]** when done.

Run All Assertions Concurrently Assertion

The *Run All Assertions Concurrently* assertion is a folder that organizes one or more child assertions that must all evaluate to true. It is similar to the "All assertions must evaluate to true" composite assertion in that the assertion succeeds only when all the child assertions succeed. The major difference is that the Run All Assertions Concurrently assertion runs all its immediate child assertions concurrently, without waiting for earlier children to finish before beginning to evaluate subsequent children, helping reduce overall latency. This is useful when a response must be assembled from the results of a number of HTTP requests to various back-end services and you don't want all of the calls to be issued serially.

For more information about parent and child assertions in a policy, see "Policy Organization" on page 2.

Tip: The Run All Assertions Concurrently assertion will always succeed if there are no child assertions contained within it, or if all the child assertions have been disabled.

Technical Issues to Consider

The Run All Assertions Concurrently assertion must be used with care, as running many concurrent threads may cause issues that would not otherwise appear if the standard "All assertions..." folder is used. In particular, note the following:

- Running assertions concurrently may cause the Gateway to run out of memory more quickly. The effect is similar to increasing the values for the *io.httpMaxConcurrency* and *io.httpCoreConcurrency* cluster properties.
- Messages larger than 200KB in size are not recommended.
- When too many assertions are being processed concurrently, new concurrent assertions will be delayed.
- If the number of assertions queued up to run concurrently exceeds the limit set in the *concall.globalMaxConcurrency* cluster property, there could be a large delay if there are slow jobs in the queue (for example, HTTP routing to a slow back-end web service). Should this occur, try the following:
 - reduce the number of concurrent assertions
 - increase the *concall.globalMaxConcurrency* cluster property
 - decrease the *io.httpMaxConcurrency* cluster property

- Unlike the "All assertions..." folder where processing stops at the first child assertion that fails, the Run All Assertions Concurrently assertion evaluates *all* child assertions, even those below one that fails. When a failure occurs, the assertion status code returned is for the first child assertion that failed.
- Only context variables of type String or Message that are used by a child assertion will be available when a child assertion is executing.
- The original request and response message, including any transport-specific message, will be unavailable unless they were copied into String variables before concurrent execution or copied as Message variables using the [Set Context Variable](#) assertion, with a source expression similar to `${request.mainpart}`.
- If an assertion inside a Run All Assertions Concurrently assertion writes to the default request or response, that information is ignored and lost.
- Gathered credentials and authenticated user information is not available during concurrent execution.
- A Run All Assertions Concurrently assertion may be nested within another Run All Assertions Concurrently assertion, but this will cause the Gateway to run out of concurrency more quickly.

Configuring the Assertion

You can configure the concurrency behaviour using these cluster properties:

```
concall.globalMaxConcurrency  
concall.globalCoreConcurrency  
concall.globalMaxWorkQueue
```

For more information, see "Input/Output Settings" in Gateway Cluster Properties in the *Layer 7 Policy Manager User Manual*.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the policy development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below
2. Populate the folder with child assertions using any of the following methods:
 - Add an assertion by dragging and dropping it from policy window or the [Assertions] tab.

- Remove an assertion by dragging and dropping it back into the policy window or by [deleting](#) the assertion.

Run Assertions for Each Item Assertion

The *Run Assertions for Each Item* assertion is a composite assertion that behaves similarly to the [All Assertions Must Evaluate to True](#) assertion, with the exception that it may evaluate its child assertions more than once (as a loop) or possibly not at all, depending on the context variable it is configured to use.

This assertion evaluates its child assertions once for each member of the specified multivalued context variable. If the multivalued variable is empty or does not exist, the assertion will always succeed and will not evaluate any of its child assertions. For more information about parent and child assertions in a policy, see "Policy Organization" on page 2.

W A R N I N G

Avoid using a debug trace policy when the Run Assertions for Each Item assertion is present in a policy, especially if the assertion is nested beneath another Run Assertions for Each Item assertion. Doing so can cause the trace policy to be invoked a very large number of times, potentially impacting system performance and generating excessively large trace logs.

The following example illustrates how the Run Assertions for Each Item assertion might be used. In the sample policy fragment below, you will send a list of greetings to a pair of URLs:

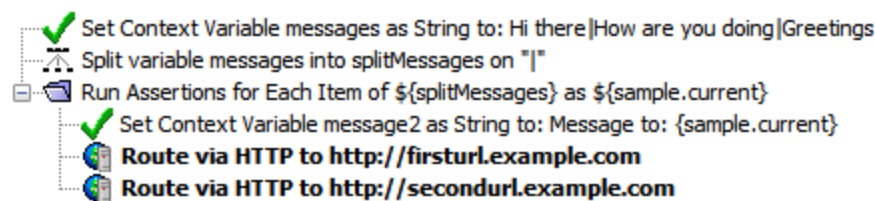


Figure 228: Sample Policy Fragment for Run Assertions for Each Item assertion

The following table explains each line in the policy example:

Table 212: Explanation of Policy Fragment for Run Assertions for Each Item assertion

1	The Set Context Variable assertion creates a new variable named <i>messages</i> , with the content "Hi there How are you doing Greetings".
2	The Split Variable assertion breaks up the value in <i>messages</i> into a new multivalued context variable name <i>splitMessages</i> . The break occurs at the " " character.

3	The Run Assertions for Each Item assertion, using <code>splitMessages</code> as the input and "sample" as the variable prefix. When you point at this with the mouse, a tooltip displays the context variables that will be created by this assertion, using the variable prefix entered.
4	The first child assertion. This uses the Set Context Variable assertion to create a new variable named <code>message2</code> . Set this variable to whatever is in <code>splitMessages</code> for that iteration. This is specified by the context variable <code>\${sample.current}</code> . Thus, for the first iteration, <code>message2=Hi there</code> ; for the 2nd iteration, <code>message2= How are you doing</code> , etc.
5	<p>The Route via HTTP(S) assertions are configured to use the context variable <code>\${message2}</code> as a request message source. The message contained in the variable <code>message2</code> is updated on each iteration and sent to the two URLs. This will result in the following POST commands:</p> <pre> POST "Hi there" to firsturl POST "Hi there" to secondurl POST "How are you doing" to firsturl POST "How are you doing" to secondurl POST "Greetings" to firsturl POST "Greetings" to secondurl </pre> <p>If any of the POST requests fail to reach the target server, the Run Assertions for Each Item assertion will fail and the remaining POSTS will not be attempted.</p>

When the *Run Assertions for Each Item* assertion is finished, the `${sample.iterations}` variable will contain "3", while the `${sample.exceededlimit}` variable will contain "false".

For more information on multivalued context variables and how they work, see [Working with Multivalued Context Variables](#) in the *Layer 7 Policy Manager User Manual*.

Terminating the Execution of Child Assertions

It is possible to terminate the execution of the child assertions prior to the completion of the loop. To do this, set the following context variable, either inside or outside of the loop:

Variable name: `${<prefix>.break}` (where "<prefix>" is from Table 214)

Variable value: **true**

When set outside of the loop, the loop will be terminated immediately. When set inside the loop, the loop will complete its iteration before terminating.

Tip: Use the "Set Context Variable Assertion" on page 656 to create the variable and set the value to **true**.

Context Variables Created by this Assertion

The Run Assertions for Each Item assertion sets the following context variables. **Note:** The `<prefix>` is set in the assertion properties (Figure 229). There is no default.

Table 213: Context variables created by Run Assertions for Each Item assertion

Variable	Description
<code>\${<prefix>}.current</code>	The current member of the multivalued variable. During iteration, the <code>\${<prefix>}.current</code> variable will take on the value of each member of the multivalued variable. After iteration, it remains set to the last value it had. If the multivalued variable existed but was zero-length, then the <code>.current</code> variable will not exist after iteration has finished.
<code>\${<prefix>}.iterations</code>	The number of iterations that have been completed successfully.
<code>\${<prefix>}.exceededlimit</code>	This variable contains "true" if the assertion ended because the iteration limit was reached, otherwise it contains "false". While processing is in progress, this variable always contains "false".

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the **Run Assertions for Each Item Properties** automatically appear; when modifying the assertion, right-click **Run Assertions for Each Item** in the policy window and select **Run Assertions for Each Item Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

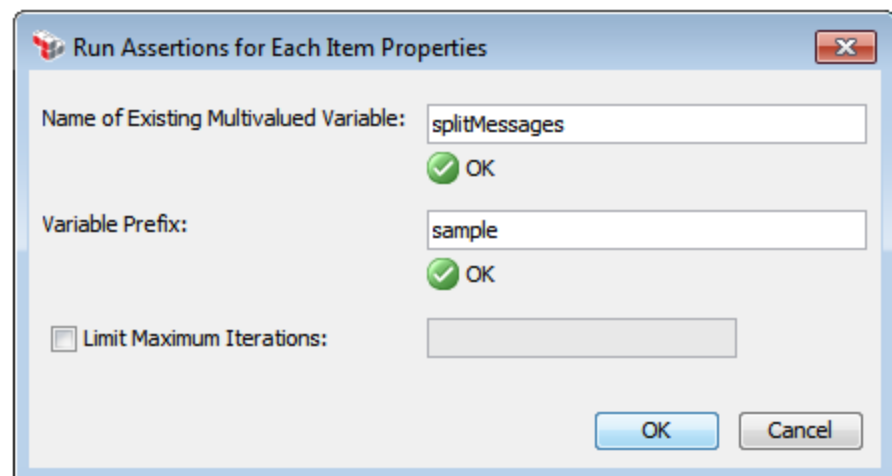



Figure 229: Run Assertions for Each Item Properties

- Configure the properties as follows:

Table 214: Run Assertions for Each Item settings

Setting	Description
Name of Existing Multivalued Variable 	<p>Specify the name of the variable to iterate over.</p> <p>Note: If this variable is empty, the assertion always succeeds and none of the child assertions are evaluated.</p>
Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>
Limit Maximum Iterations	<p>Optionally, you may limit the number of iterations by selecting this check box and entering the maximum number of iterations in the corresponding field. If a maximum is not specified, then the assertion will attempt to process all members in the multivalued context variable.</p> <p>If specified, iteration will halt after the child policy has invoked this many times, regardless of whether it has been invoked for all members in the multivalued variable.</p> <p>Tip: After the iterations are complete, the context variable <code>\${<prefix>.exceededlimit}</code> will be set to "true" if this limit was hit. Otherwise, it will be set to "false".</p>

Set Context Variable Assertion

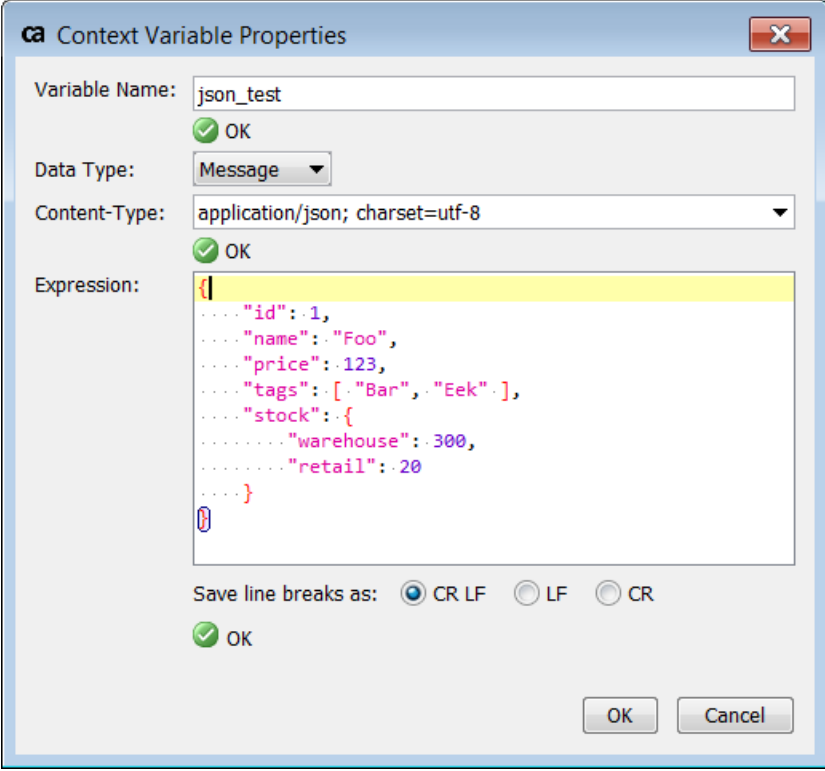
The *Set Context Variable* assertion allows you create custom context variables. These user-defined variables behave the same as the system predefined ones, except that you can control the contents of the variables. You can even define complete messages with a custom context variable. These messages can be used later in the policy by the [Route via HTTP\(S\)](#) assertion as the request message source, and by the [Evaluate Response XPath](#) assertion as the XML message source.

Tip: This assertion can be used to map one of the predefined context variables to a different name. This is necessary in instances where the predefined context variable name is incompatible with a specific subsystem.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.

2. When adding the assertion, the **Context Variable Properties** automatically appear; when modifying the assertion, right-click **Set Context Variable...** in the policy window and select **Context Variable Properties** or double-click the assertion in the policy window. The assertion properties are displayed.



The dialog box is titled "Context Variable Properties" and contains the following fields and controls:

- Variable Name:** A text field containing "json_test".
- Data Type:** A dropdown menu set to "Message".
- Content-Type:** A dropdown menu set to "application/json; charset=utf-8".
- Expression:** A text area containing a JSON object:

```
{
  "id": 1,
  "name": "Foo",
  "price": 123,
  "tags": ["Bar", "Eek"],
  "stock": {
    "warehouse": 300,
    "retail": 20
  }
}
```
- Save line breaks as:** Three radio buttons: "CR LF" (selected), "LF", and "CR".

There are green checkmark icons next to the "Data Type", "Content-Type", and "Save line breaks as" sections. At the bottom right are "OK" and "Cancel" buttons.


Figure 230: Context Variable Properties - data type "Message"


Figure 231: Context Variable Properties - data type "Date/Time"

3. Configure the properties as follows:

Table 215: Context Variable settings

Setting	Description
Variable Name	<p>Enter a name for the context variable. This name can include letters or number, but it should not be enclosed by the <code>{ }</code> delimiter characters. For example, use "gatewayTime" and not <code>"\${gatewayTime}"</code>.</p> <p>You can also enter request or response to set the request message or response message, respectively. A validation message provides instant feedback on the context variable name entered. For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>
Data Type	<p>If the variable name is not built-in, specify the data type from the drop-down list:</p> <ul style="list-style-type: none"> • String: The variable will contain a string of characters. • Integer: The variable will contain integers. • Message: The variable will contain a message. Enter the message body into the Expression box below. Message type variables can be used for later routing or XPath assertions. • Date/Time: The variable will contain date/time information. For more information, see "Context Variable Data Types" in Context Variables in the <i>Layer 7 Policy Manager User Manual</i>. <p>Tip: Creating a new variable of type "Date/Time" and accepting the defaults in all the other fields will produce a</p>

Setting	Description
	<p>variable that is identical to the built-in variable <code>\${gateway.time}</code>. The default value is an UTC formatted ISO-8601 string of <code>\${gateway.time}</code>. For more information, see "Date/Time Variables" in Context Variables in the <i>Layer 7 Policy Manager User Manual</i>.</p> <p>You must select "Message" if setting a request or response variable.</p>
Format <i>(Date/Time only)</i>	<p>Choose a format from the drop-down list. The assertion will use this format to interpret the value in the Expression field. The default is "<auto>", which means the assertion will try to determine the format automatically. You may reference a context variable, which will be evaluated as part of an expression and must resolve to a single format string.</p> <p>Tip: The formats <Timestamp>, <Millisecond Timestamp> (13 digits), and <Second Timestamp> (10 digits) can be used for finer control in parsing the Expression field:</p> <ul style="list-style-type: none"> A timestamp entered in the Expression field will successfully match the formats: <auto> and <Timestamp> and <Millisecond Timestamp> if it is a millisecond timestamp and <Second Timestamp> if it is a seconds timestamp. A timestamp in milliseconds entered will successfully match the formats: <auto>, <Timestamp>, and <Milliseconds Timestamp>. A timestamp in seconds entered will successfully match the formats: <auto>, <Timestamp>, and <Seconds Timestamp>. <p>Note: You can customize the options shown in the drop-down list by editing the cluster property <code>datetime.customFormats</code>. You can customize the formats attempted by the "<auto>" selection by editing the cluster property <code>datetime.autoFormats</code>.</p>
Time Offset <i>(Date/Time only)</i> 	<p>Enter a time offset. The value can be negative or positive and can reference variables. The unit of time applied is determined by the drop down beside the field.</p> <p>Default: seconds</p> <p>Choose the offset unit from the drop-down list if necessary.</p> <p>You may reference context variables.</p>
Preview <i>(Date/Time only)</i>	<p>Displays a preview of the date format based on date expression entered. The preview includes any time offset entered, except when a context variable has been specified.</p> <p>The preview of the parsed date expression can be used to troubleshoot formats based on sample inputs.</p> <p>Notes: (1) The format used for the preview is ISO 8601, in the W3C format <code>yyyy-MM-dd'T'HH:mm:ss.SSS'Z'</code>. The date format is shown in GMT. (2) A preview is displayed only when it is possible to parse the</p>

Setting	Description
	format entered. If a preview is not possible, "No preview is available" is displayed.
Content-Type (Message only)	<p>If the Data Type is "Message", use the drop-down list to select a Content-Type for the message (for example, "text/xml; charset=utf-8"). The Content-Types listed have a characterencoding that is compatible with the default XML encoding.</p> <p>A custom value for content-type can be entered. Variables are not supported for this field.</p> <p>The instant validator will check for correct syntax.</p>
Expression 	<p>Enter the string value or message body here, or leave empty. You can use existing context variables within the expression by enclosing them within the standard "\${ }" variable reference syntax. For example: "\${gateway.time}".</p> <p>The expression for variables of type Message can be a simple text document, an XML document, a MIME multi-part document, or other valid textual data for the content type entered. For example, the expression can be a SOAP request message. You can modify the Route via HTTP(S) assertion to use this expression referencing its variable as the "request message source", rather than using the default request.</p> <p>Syntax Highlighting</p> <p>The Expression box displays syntax highlighting for Message variables of Content - Type "text/xml" or "application/json". This helps you identify errors more easily.</p> <p><i>XML Highlighting:</i></p> <ul style="list-style-type: none"> • non-XML values are not highlighted • for invalid XML values, only the correctly formatted XML elements are highlighted • for valid XML elements, the opening and closing tags are highlighted <p>Tip: Closing tag for an XML element are auto-completed for your convenience. For example, typing "<xml> test </" will result in the closing tag "</xml>" being auto completed.</p> <p><i>JSON Highlighting:</i></p> <ul style="list-style-type: none"> • non-JSON values are not highlighted • characters that surround JSON elements (whether valid or invalid) are highlighted • placing the cursor after one of the JSON characters will highlight its paired character <p>See Figure 230 for examples of the JSON highlighting.</p>

4. Click **[OK]** when done. The new variable name is available to subsequent assertions in the policy. The **[OK]** button is unavailable if there are errors identified by any of the [instant validators](#).

Split Variable Assertion

The *Split Variable* assertion splits a single-value context variable into multiple values, creating a multivalued context variable as a result.

Example:

The input variable "\${varIn}" contains the value "one,two,three". After splitting this variable into "\${varOut}", you now have a multivalued context variable where:

```

${varOut[0]} = one
${varOut[1]} = two
${varOut[2]} = three

```

Example 2:

If the specified split pattern cannot be found in the source value, then entire source value is simply copied over to the target: input variable "\${varIn}" contains the value "one,two,three" but the split pattern is ';;':

```

${varOut} = one,two,three
${varOut[0]} = one,two,three

```

Note that if you attempt to use \${varOut[1]} and \${varOut[2]} in this case, no values will be returned.

Example 3:

The following is an advanced example that shows the power of using a regular expression in the Split Pattern of the Split Variable assertion. In this example, the full XML syntax is omitted for simplicity; a list of node names is the output:

The input variable "\${varIn}" contains this fragment:

```

<xml>
  <node>entry 1</node>
  <node>entry 2</node>
</xml>

```

The Split Pattern is this regular expression: `((\s)*\<. +?\>[n]*)`

After splitting this variable into "\${varOut}", you now have a multivalued context variable where:

```

${varOut[0]} = entry 1
${varOut[1]} = entry 2

```

For more information about using the indexing feature ([0], [1], etc.) in context variables, see "Indexing Options during Interpolation" under Working with Multivalued Context Variables in the *Layer 7 Policy Manager User Manual*.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the Split Variable Properties automatically appear; when modifying the assertion, right-click **Split variable <source> into <target>...** in the policy window and select **Split Variable Properties** or double-click the assertion in the policy window. The assertion properties are displayed.


Figure 232: Split Variable Properties

Tip: The '\${ }' wrapper shown in Figure 232 is for illustrative purposes only. It is not necessary to use this wrapper in either variable name. If entered, it will be removed after the assertion is saved.

3. Configure the properties as follows:

Table 216: Split Variable settings

Setting	Description
Source Variable	Enter the context variable containing the source value to be split.

Setting	Description
Target Variable 	Enter the multivalued context variable that will hold the results of the split. If this variable does not already exist, it will be created.
Split Pattern	<p>Enter the pattern that will be used to split the source variable. The Split Pattern will be interpreted as either a regular expression or as a string literal depending on the Regular Expression check box. The default is a comma (','), which indicates splits will occur after each comma found in the source variable.</p> <p>Note the following:</p> <ul style="list-style-type: none"> If the Split Pattern results in no matches on the source variable string, then the target variable's value will be the same as the input string. As a result, "\${varOut}" and "\${varOut[0]}" will be the same. If the Regular Expression check box is selected and the Split Pattern cannot be parsed, a validator warning is displayed and a WARNING message will be logged when the policy is saved. This may result in the policy failing each time it is run, as this assertion will fail each time it executes.
Regular Expression	<p>Select this check box to interpret the Split Pattern as a regular expression. This setting is the default.</p> <p>Clear this check box to interpret the Split Pattern as a literal value.</p>
Ignore Empty Values	<p>Select this check box to discard all empty values in the resulting variables, regardless of where they occur.</p> <p>Clear this check box to discard only trailing empty values. This setting is the default and it preserves the behaviour of this assertion prior to version 6.1.5.</p> <p><i>Example:</i></p> <p>Source string = "„3,,,"</p> <p>Split on " , "</p> <p>Target = <code>\${target}</code></p> <p>If "Ignore Empty Values" = SELECTED, the following value is populated into the variable <code>\${target}</code>:</p> <p style="text-align: center;"><code>\${target} = 3</code></p> <p>If "Ignore Empty Values" = UNSELECTED, the following values are populated into the multivalued variable <code>\${target}</code>:</p> <p style="text-align: center;"><code>\${target[0]} = <empty></code></p> <p style="text-align: center;"><code>\${target[1]} = <empty></code></p> <p style="text-align: center;"><code>\${target[2]} = 3</code></p>

4. Click **[OK]** when done. If the **[OK]** button is not activated, check that the Target Variable is able to accept a value.

Stop Processing Assertion

The *Stop Processing* assertion halts the processing of a policy when it is encountered. It is useful in the following scenarios when you need to stop and report on a message condition:

- A particular Evaluate [Request](#) or [Response](#) XPath assertion is successful. However, this success indicates a backend failure.
- Processing of a message needs to stop as soon as it is reported to the audit subsystem (via the [Audit Messages in Policy](#) assertion)

You can resume processing by using the "Continue Processing Assertion" on page 625.

Note: When a policy is halted and not resumed, it will cause the client application to return an error. This sudden termination of the policy may also result in misleading error messages from the client (for example, that an assertion has been falsified).

Using the Assertion

- Add the assertion as described in [Adding an Assertion](#).

The assertion is added to the policy window; no further configuration is required.

Chapter 12: Threat Protection Assertions

Notes: (1) Depending on which Gateway product you have installed, not all the assertions shown below may be available. See Features by Product in the *Layer 7 Policy Manager User Manual* for a list of which features are available for each product. (2) This category may also include custom-created encapsulated assertions. For more information, see "Working with Encapsulated Assertions" on page 126.

In the Policy Manager, the following assertions are available in the Threat Protection category of the [Assertions] tab:

Automatic Threat Protection	666
TCP/IP-Based Attacks	666
Coercive Parsing and XML Bomb	666
External Entity Attack	667
Schema Poisoning	667
WSDL Scanning	667
XML Routing Detours	668
Limit Message Size Assertion	668
Protect Against Code Injection Assertion	670
Protect Against Cross-Site Request Forgery Assertion	672
Context Variable Created by This Assertion	673
Protect Against Document Structure Threats Assertion	675
Protect Against JSON Document Structure Threats Assertion	678
Protect Against Message Replay Assertion	680
Protect Against SQL Attack Assertion	684
SQL Injections Detected	684
Scan Using ICAP-Enabled Antivirus Assertion	687
Context Variables Created by This Assertion	688
Scan Using Sophos Antivirus Assertion	692
Context Variables Created by This Assertion	692
Validate or Change Content Type Assertion	694
Validate JSON Schema Assertion	696
Validate OData Request Assertion	699
Retrieving the Service Metadata Document	700
Notes and Limitations	700
Context Variables Created by This Assertion	700
Validate XML Schema Assertion	703

The Threat Protection assertions help protect against common web service and XML threats.

Automatic Threat Protection

The CA API Gateway comes with a variety of built-in threat protection, including those for:

- TCP/IP-based attacks
- Coercive parsing and XML bomb attacks
- External entity attacks
- Schema poisoning
- WSDL scanning
- XML routing detours

Note: To ensure that the built-in protections are applied to the request, the service policy must include at least one assertion that accesses the request body—the "Protect Against Document Structure Threats Assertion" on page 675 is most commonly used.

TCP/IP-Based Attacks

The Gateway protects against all the common TCP/IP-based attacks, such as ICMP flood, SYN flooding, "ping of death", and various routing redirect style attacks.

Packet-level attacks are handled by the Gateway's default OS-level configuration (in particular, by the default firewall configuration).

Packet-level attacks are not logged.

Coercive Parsing and XML Bomb

A coercive parsing attack attempts to exploit the "bolt-on" interfaces used to link legacy systems with XML components in an existing infrastructure. The attack tries to overwhelm a system's processing capabilities or install malicious mobile code.

This attack is also known as a "DTD Entity Expansion Attack". According to the SOAP specifications, a SOAP message must not contain a DTD declaration. The Gateway can prevent messages containing DTD declarations from passing through, by terminating them before any policy processing begins. To block DTD declarations, ensure that the "Protect Against Document Structure Threats Assertion" on page 675 is present in the service policy or in a [global policy fragment](#).

From a technical perspective, the XML parser will not allow DOCTYPE declarations. When the parser encounters a message containing a DOCTYPE, it terminates parsing without expanding the entity or entities. The Gateway then logs and [audits](#) a warning that a message was badly formed. This allows administrators to monitor potential intrusion attempts, while keeping the protected services safe.

External Entity Attack

XML can be used to build documents dynamically by pointing to a URI where the actual data exists. These external entities may not be trustworthy, as an attacker could replace the data being retrieved with malicious code.

By default, the Gateway does not resolve external entities and the Gateway can be configured using the [Evaluate Request XPath](#) and [Evaluate Response XPath](#) assertions to block all messages containing references to external entities.

Schema Poisoning

A schema describes the constraints and structure of a message, as well as optional processing instructions. Parsers use schemas to interpret web service messages. Since schemas describe the necessary preprocessing instructions, they are vulnerable to tampering if not stored securely.

Schema poisoning involves an attacker attempting to compromise a system by replacing or tampering with the schema. To protect against this, the Gateway does not load schemas from unauthorized locations. All schemas must be loaded by the administrator; dynamic loading is not permitted.

WSDL Scanning

A WSDL document describes a web service, including what operations are supported. In addition to this information, a WSDL may expose details about the implementation that could be used by an attacker. An attacker might cycle through the various command and string combinations to discover unintentionally related or unpublished application program interfaces.

The Gateway selectively proxies all internal WSDLs, shielding access to the original WSDLs on the application servers. The Gateway will deny direct access to all WSDLs even when an attacker guesses a related unpublished WSDL. By preventing unauthorized access to a web service (and its WSDL), this type of information scanning is blocked.

XML Routing Detours

XML routing detours can occur if an attacker sends a message to a Web service containing bogus routing information to override the normal routing. The detoured message can then pass through unknown or untrusted hosts, making it possible for the attacker to view or modify the contents of the message. This rerouting is prevented when any of the Policy Manager routing assertions are used. These assertions explicitly define the route of the message and overriding the route is not possible. For more information, see "Chapter 8: Message Routing Assertions" on page 507.

XML routing may also occur if WS-Routing (Web Services Routing) is used by the sender/receiver of the message, because this specification permits the source to define the route of a message.

Although the Gateway enforces strict, explicit routing of messages, intermediates can also be prevented from viewing and or changing sensitive content by using the extensive encryption and signature facilities within the Gateway.

Limit Message Size Assertion

The *Limit Request Size* assertion allows you to specify a size limit for an entire message (including attachments) or just the XML portion of a message (not including attachments). When the request size exceeds the designated limit, the Gateway will reject the message and terminate policy execution.

This assertion should be placed before the routing assertion in the policy.

Note: The Limit Request Size assertion was designed to prevent Denial of Service attacks, hence the immediate cessation of the policy upon failure. If you wish to continue processing the policy even after the request size has exceeded a certain value, you could use the [Compare Expression](#) assertion with the `${request.http.header.content-length}` context variable. However be aware that this method is not foolproof as the declared content-length is easily forged or it may not be present in the request. For added protection, also include a Limit Message Size assertion later in the policy to enforce a hard cap should the policy logic be misled by the content-length header. Also, keep in mind that the `${request.http.header.content-length}` will be smaller than the message if compression is used.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153. Note that you can also select the target message in the assertion properties.

Using the Assertion

1. Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **Message Size Limit Properties** automatically appear; when modifying the assertion, right-click **<target>: Limit Message Size** in the policy window and select **Message Size Limit Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

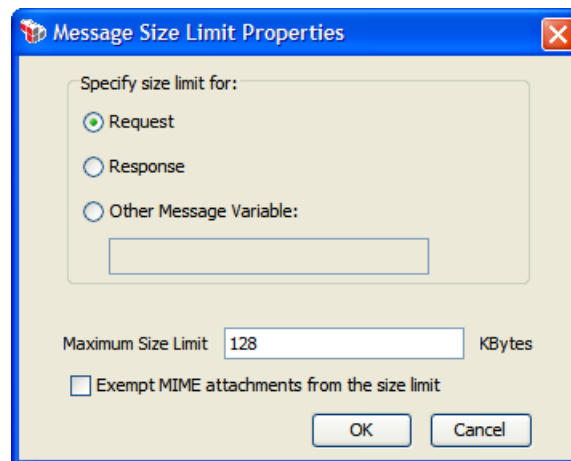




Figure 233: Message Size Limit Properties

3. Configure the properties as follows:

Table 217: Request Size Limit settings

Setting	Description
Specify size limit for 	Select the target message to be controlled by the size limit: Request , Response , or a context variable of type Message. Tip: The target message can also be changed by using the Message Target dialog. For more information, see "Selecting a Target Message" on page 153.
Maximum Size Limit 	Enter the maximum message size that should be accepted by the Gateway in kilobytes (KB). The value must be a whole number or a context variable that resolves to the limit.
Exempt MIME attachments from the size limit	By default, the size limit applies to the entire message. To apply the limit to only the XML portion of the message and exempt any MIME attachments, select the check box. Note: Messages with more than 32K of headers in the MIME attachment portion of a message will always be rejected, regardless of the setting of this check box.

4. Click **[OK]** when done.

Protect Against Code Injection Assertion

The *Protect Against Code Injection* assertion provides threat protection against code injection attacks targeting web services and Web applications, including AJAX applications. Use this assertion to protect against the following threats:

HTML/JavaScript Injection (Cross-site Scripting)

PHP Code Injection—Eval injection

Shell Injection

LDAP DN Injection

LDAP Search Injection

XPath Injection

This assertion can help protect vulnerable parameters in the path (or URI) of the URL, in addition to the URL query string and message body.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **Code Injection Protection Properties** automatically appear; when modifying the assertion, right-click **<target>: Protect against Code Injection** in the policy window and select **Code Injection Protection Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

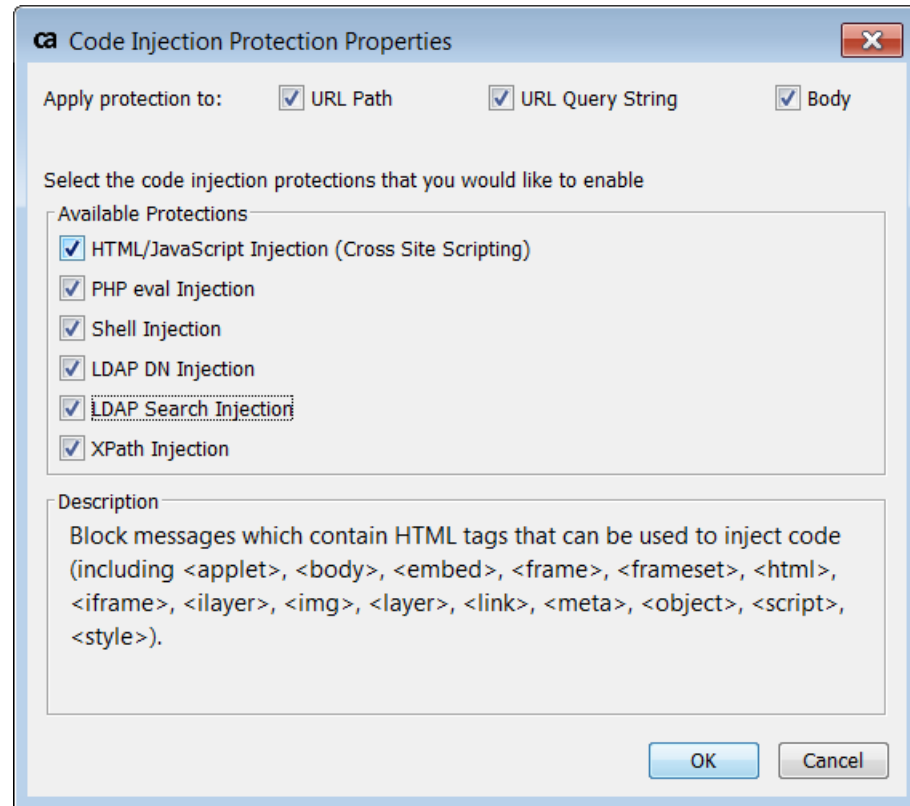



Figure 234: Code Injection Protection Properties

3. Configure the properties as follows.

Table 218: Code Injection Protection settings

Setting	Description
Apply protection to: 	Specify where to apply the protection: <ul style="list-style-type: none"> • URL Path: Select this to protect the URL Path. • URL Query String: Select this to protect the query parameters in the URL. • Body: Select this to protect the body of the message. These will be scanned depending on the Content-Type header: <ul style="list-style-type: none"> • <i>application/x-www-form-urlencoded</i>: Scans Form Post parameters • <i>multipart/form-data</i>: Scans each MIME part; depends on Content-Type of MIME part • <i>text/xml</i>: Scans attribute values and character-data • anything else: Scans the entire message body
Available Protections	Select one or more injection threats to protect against. Point at each option to see a description of the protection offered. The assertion will

Setting	Description
	<p>fail upon the first protection violation detected.</p> <p>IMPORTANT: This assertion checks for injection of <i>any</i> executable code, not just malicious code. This is because it is not always possible to determine which code is malicious or benevolent. Be especially careful when using this protection on responses, because returned HTML often contains legitimate uses of the restricted tags.</p>

- Click **[OK]** when done.

Protect Against Cross-Site Request Forgery Assertion

The *Protect Against Cross-Site Request Forgery* assertion helps detect and prevent against CSRF (Cross-Site Request Forgery) attacks.

This assertion provides two mechanisms to help detect a CSRF attack:

- *Double Submit Cookie Validation:* This can be used to validate the contents of a cookie that contains some session identifier, to see if it matches the same session identifier contained in a request parameter.
- *HTTP Referer validation:* This can be used to ensure that the referer value belongs to a whitelist of domains. Although the referer domain is easily spoofed, this validation reduces the attack vectors for a CSRF attack.

This assertion can only work with HTTP requests and will fail if the request is not HTTP.

Double Submit Cookie Validation

If enabled, any of the following conditions will cause the assertion to fail:

- Required cookie is not present.
- Required cookie has a name that contains non-ASCII characters.
- Required parameter is not found or does not contain a single value.
- The parameter does not match the cookie's value.

For more information about double submit cookies, visit the OWASP site at: <http://www.owasp.org> and search for "CSRF double submit cookies".

HTTP-Referer Header Validation

If enabled, the following conditions will cause the assertion to fail:

- If the HTTP-Referer header is empty and the **Allow empty values** check box is not selected.
- The value of HTTP-Referer is not a valid URL.
- The domain of the HTTP-Referer does not belong to the whitelist.

Context Variable Created by This Assertion

The Protect Against Cross-Site Request Forgery assertion will set the context variable **csrf.valid.token** to the value of the cookie. You can use this variable later in the policy to validate the value.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Protect Against CSRF Forgery** in the policy window and select **CSRF Protection Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

CSRF Protection Properties

Double Submit Cookie Validation

☒ Enable Double Submit Cookie Validation

Cookie Name:

HTTP Parameter Name:

Allowed HTTP Methods:

HTTP-Referer Header Validation

☒ Enable HTTP-Referer Header Validation

Options: ☐ Skip validation if header is missing or empty

List of Trusted Domains:

Valid Domains:

Add Edit Remove

OK Cancel

Figure 235: CSRF Protection Properties

3. Configure the dialog as follows:

Table 219: CSRF Protection settings

Setting	Description
Enable Double Submit Cookie Validation	Select this check box to enable double submit cookie validation. Clear this check box to not check for double cookie submissions. This will deactivate the other settings in this section.
Cookie Name	Enter the name of the cookie that the HTTP parameter name will be checked against.
HTTP Parameter Name	Enter the name of the HTTP parameter to be checked against the cookie name.
Allowable HTTP Methods	Choose the HTTP methods allowed: GET , POST , or GET and POST .

Setting	Description
Enable HTTP-Referer Header Validation	Select this check box to enable validation of the HTTP-Referer header value. Clear this check box to not validate the HTTP-Referer header value. This will deactivate the other settings in this section.
Skip validation if header is missing or empty	Select this check box to omit the validation step if the HTTP-Referer is missing or empty. Clear this check box to always validate the HTTP-Referer header.
Valid Domains	From the drop-down list, choose the valid domains: <ul style="list-style-type: none"> • Current Domain: Use the current domain of the Gateway. This will match the header against the domain value associated with the request, which may be the domain of the cluster <u>or</u> the individual Gateway host. • List of Trusted Domains: Enter your own list of trusted domains in the box below. <ul style="list-style-type: none"> • Click [Add] to add a new domain. • Click [Edit] to modify the selected domain. • Click [Remove] to remove the selected domain from the list.

4. Click **[OK]**.

Protect Against Document Structure Threats Assertion

The *Protect Against Document Structure Threats* assertion allows you to specify size limits for incoming XML requests to protect against XDoS (XML Denial of Service) attacks using oversized files. When the text or attributes of an incoming request exceed the specified limits, the Gateway rejects the message and blocks further processing of the policy.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **Document Structure Threat Protection Properties** automatically appear; when modifying the assertion, right-click

<target>: Protect against Document Structure Threats in the policy window and select **Document Structure Threat Protection Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 236: Document Structure Threat Protection Properties

3. Configure the properties as follows:

Table 220: Document Structure Threat Protection settings

Setting	Description
Reject if any XML contiguous text has length exceeding	<p>Select this check box to reject any incoming request with a text node (or CDATA section) containing more than the specified number of contiguous characters. Clear this check box to accept any contiguous length.</p> <p>"Contiguous" in this context refers to the characters between XML tags. For example: <code><tag>this is a string of contiguous characters</tag></code>. This check does not differentiate between start and end tags, so the following text is also considered contiguous characters: <code></endTag>blank spaces and return characters between tags are also contiguous text<startTag></code>.</p> <p>Note: The length of <i>attribute names</i> are excluded from this setting. To manage requests based on the attribute name, use the "Reject if any XML attribute name has length exceeding" setting.</p>
Reject if any XML attribute value has length exceeding	<p>Select this check box to reject any incoming request with an attribute value longer than the specified number of characters. Clear this check box to accept attribute values of any length.</p> <p>The length of an attribute value is the number of characters between the quotes of any attribute, not including the attribute name itself. For</p>

Setting	Description
	example, the length of this attribute value is 12: <code></code> .
Reject if any XML attribute name has length exceeding	<p>Select this check box to reject any incoming request with an attribute name longer than the specified number of characters. Clear this check box to accept attribute names of any length.</p> <p>Note: XML attribute name lengths are independent of the <i>"Reject if any XML contiguous text has length exceeding"</i> setting.</p>
Reject if XML element nesting depth exceeds	<p>Select this check box to reject any incoming request that contains more than the specified number of nested levels. Clear this check box to accept requests with any number of nested levels.</p> <p>The nesting count begins at the top of the XML document. If it is a SOAP message, the envelope is level 1, the body is level 2, etc.</p>
Reject if distinct namespace declarations exceeds	<p>Select this check box reject any incoming requests that contains more than the specified number of distinct namespace URI declarations. Clear this check box to accept requests with any number of namespace declarations.</p> <p>Note: A value of '0' (zero) means unlimited, which is the same as clearing the check box.</p>
Reject if distinct namespace prefix declarations exceeds	<p>Select this check box reject any incoming requests that contains more than the specified number of distinct namespace prefix declarations. Clear this check box to accept requests with any number of namespace prefix declarations.</p> <p>Note: A value of '0' (zero) means unlimited, which is the same as clearing the check box.</p>
Reject SOAP request that contain more than	<p>Select this check box to reject any SOAP requests with more than the specified number of payload elements. Clear this check box to ignore the number of payload elements in a request.</p> <p>A SOAP envelope requires one body section but may contain multiple payload elements (header is optional):</p> <pre> Envelope Header (optional) Body Payload Payload Payload </pre> <p>Multiple payloads are uncommon and an attack may be launched using multiple payload elements to evade simplistic validity checks. For example, the XPath might match against payload #2, but the application ignores payload #2 and uses payload #1 instead.</p>
Require a valid SOAP envelope (one Body, no	<p>Select this check box to reject any requests that do not contain a valid SOAP envelope. Clear this check box to not check the validity of a SOAP envelope.</p>

Setting	Description
trailers)	<p>A valid envelope contains exactly one Body section, optionally preceded by exactly one Header section, with no SOAP trailers.</p> <p>This setting guards against invalid SOAP envelopes that contain multiple body sections or trailers, which may be caused either by an attack or an error in the client application.</p>

- Click **[OK]** when done.

Protect Against JSON Document Structure Threats Assertion

The *Protect Against JSON Document Structure Threats* assertion validates and enforces constraints on the structure of JSON documents. When the structure of an incoming JSON document exceeds a specified constraint, the Gateway rejects the JSON document and blocks further processing of the policy.

The following example JSON documents are referenced in the property descriptions below:

- Example 1:* Single typed value document:

"one simple value"

- Example 2:* Typical document:

```
{
  "msg": "Hello",
  "color": [0,0,255],
  "options": {
    "underline": false,
    "bold": true
  }
}
```

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Tips: (1) This assertion should be placed before any other JSON-related assertions ([Apply JSON Transformation](#), [Evaluate JSON Path](#), [Validate JSON Schema](#)) in order to protect them against DOS attacks. (2) This assertion will always evaluate the document for valid JSON structure, even if no limits are enabled.

Using the Assertion

- Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the **JSON Document Structure Threat Protection Properties** automatically appear; when modifying the assertion, right-click **<target>: Protect against JSON Document Structure Threats** in the policy window and select **JSON Document Structure Threat Protection Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

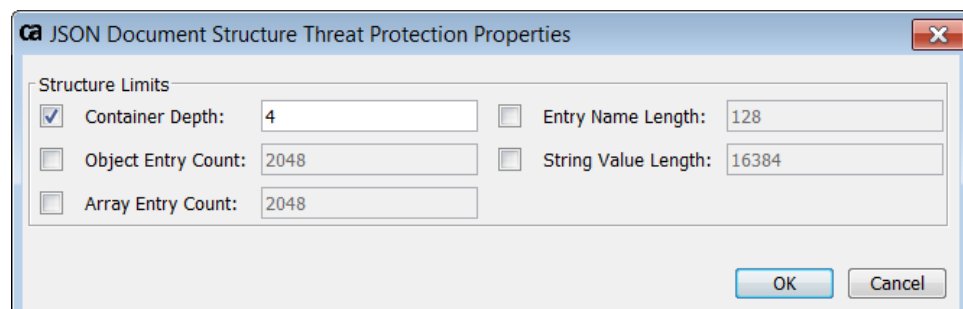


Figure 237: JSON Document Structure Threat Protection Properties

3. Configure the properties as follows:

Table 221: JSON Document Structure Threat Protection settings

Setting	Description
Container Depth	<p>This is the most nested containers within a JSON document. Example 1 above contains a string, which is the only type of document with a container depth of 0. Every open object or array increases the container depth by one. In Example 2, the "msg" entry has a container depth of 1, and the values in the "color" array have a depth of 2.</p> <p>Select this check box to reject any JSON document with a container depth that exceeds the value entered here. The container depth can be from 0 to 30. Default: 4</p> <p>Note: A value of '0' is valid only for a single typed value document.</p>
Object Entry Count	<p>This is the maximum number of entries (comma delimited string:value pairs) in a single object. Example 1 has a maximum object entry count of 0 (as it is a single typed value, it does not contain an object). Example 2 has a maximum entry count of 3 (the top level entries: "msg", "color", "options").</p> <p>Select this check box to reject any JSON document with an object entry count that exceeds the value entered here.</p>
Array Entry Count	<p>This is the maximum number of entries (comma delimited values) in an array. The array in Example 2 has an entry count of 3. As with the</p>

Setting	Description
	<p>object entry count, the count is done for each array, not all arrays in the document.</p> <p>Select this check box to reject any JSON document with an array entry count that exceeds the value entered here.</p>
Entry Name Length	<p>This is the maximum number of characters in the name of an entry. A setting of 7 would be violated by the string "underline" (which contains 9 characters), but not by "msg", "bold", "color", or "options".</p> <p>Select this check box to reject any JSON document with a name length that exceeds the value entered here.</p>
String Value Length	<p>This is the maximum number of characters in a string value. In Example, 2, the longest string value is 5 ("Hello").</p> <p>Select this check box to reject any JSON document with a string value that exceeds the value entered here.</p>

- Click **[OK]** when done.

Protect Against Message Replay Assertion

The *Protect Against Message Replay* assertion is used to protect the Gateway against possible replay attacks. This replay protection can either be cluster-wide (default) or per node, depending on the setting of the cluster property *cluster.replayProtection.multicast.enabled*.

Note the following important issues when using this assertion:

- Depending on the expiry period set in the assertion, using the Protect Against Message Replay assertion in a Gateway cluster may increase request message processing time and require more memory. To mitigate this, place this assertion after a [Authenticate User or Group](#) or [Authenticate Against Identity Provider](#) assertion to help confine the protection to successfully authenticated messages, thereby reducing system processing and memory requirements.
- This assertion should not be used in any policy that will process messages from JMS destinations that are configured with the "On completion" acknowledgment mode without a specified failure queue.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To learn more about selecting the target identity for this assertion, see "Selecting a Target Identity" on page 152.

Details for Advanced Users

The Protect Against Message Replay assertion uses an internal replay ID. This ID is based on either a WS-Addressing Message ID or the timestamp of the request combined with other information that depends on how the message was signed:

- For a request message signed with a WS-Security one-shot X.509 signature, the replay ID is comprised of the following:
 - The SHA-1 of the WS-Addressing MessageID, if present, or the timestamp creation date
 - The signing certificate's subject and issuer DNs
 - The signing certificate's subject key identifier
- For a request message signed with a key derived from a [WS-SecureConversation](#) security context, the replay ID is the MessageID or timestamp created date and the security context identifier.
- For a request message signed with a key derived from an EncryptedKey, the replay ID is the MessageID or timestamp created date and the EncryptedKeySHA1 value.
- For a request message signed with a WS-Security Kerberos token, the replay ID is the MessageID or timestamp created date and the SHA-1 of the Kerberos token.

In all cases, the granularity of the timestamps is determined by the message sender. While the Securespan XML VPN Client always uses at least millisecond-granular timestamps (with a random count of up to one million nanoseconds, to reduce the chance of an ID collision), many tools will use second-granular timestamps by default, resulting in spurious duplicate IDs if MessageIDs are not used and more than one message is sent per second per signing identity.

The Protect Against Message Replay assertion offers two different modes: *Default* or *Custom*.

Default Mode

The assertion first attempts to use a signed WS-Addressing Message ID in the message as the basis for replay protection. If the Securespan XML VPN Client is deployed, you can enforce the presence of Message IDs by using the "Require WS-Addressing Assertion" on page 477.

Note: A Message ID that is present but not signed will not be used by the Protect Against Message Replay assertion. The assertion will use a signed time stamp instead, if one is available.

If no Message ID is present (and the policy is not configured to enforce the presence of one), the message time stamp is used for replay protection. The Gateway will reject a message as a possible replay if detects any of the following:

- A duplicate creation time stamp in a message
- An expired time stamp is present
- The creation time stamp is more than 30 days old.

In the Default mode, the Protect Against Message Replay assertion behaves exactly the same as the *WSS Replay Protection* assertion found in versions prior to 5.2.

Custom Mode

In this mode, you may specify a context variable that contains the identifier to check and how long the identifier should be saved. This allows you to verify non-SOAP messages. It will not perform signature verification or validate the timestamp.

Note: The Custom mode only deals with checking for replay of the identifier. The policy administrator is responsible for ensuring that the identifier can be trusted and that the current time is within the time stamp created/expires times.

The custom mode allows you to create your own custom replay protection policy fragment when combined with other assertions.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Protect Against Message Replay** in the policy window and select **Message Replay Protection Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

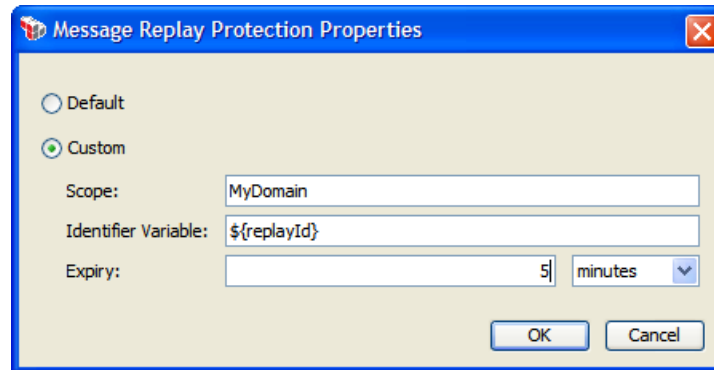




Figure 238: Message Replay Protection Properties

3. Configure the properties as follows:

Table 222: Message Replay Protection settings

Setting	Description
Default Custom	<p>Choose the mode of operation: [Default] or [Custom]. Refer to the introduction to this topic for a description of each mode.</p> <p>The [Default] mode replicates the functionality in the <i>WSS Replay Protection</i> assertion in versions prior to 5.2. This mode requires no further configuration.</p>
Scope 	<p>The replay scope lets you specify a scope for the uniqueness of the message identifier. For example, a message identifier scheme may be global, or per service, or could use some other granularity.</p> <p>Specify a scope for the uniqueness; context variables are permitted. Examples:</p> <p>Service scoped: <code>\${service.oid}</code></p> <p>Customer scoped: <code>Customer 7</code> (maximum 250 chars)</p> <p>Global scope: <code><leave blank></code></p> <p>Tip: The scoping can be performed by the policy author (for example, by specifying an identifier as <code>\${service.oid}/\${myId}</code>) but such an approach risks collisions if other services do not use service-scoped identifiers.</p>
Identifier Variable 	<p>Specify a context variable containing the Message ID to be processed. You can enter the variable in the format <code>\${myVar}</code> or <code>myVar</code>.</p> <p>Ensure that this Message ID has been signed and is unique.</p>
Expiry	<p>Specify how long the identifier should be saved. This expiry time is the lifetime of the message—that is, the amount of time the identifier will be stored in the cache from the time it was received. The default is 5 minutes.</p> <p>Tip: The expiry time should be greater than 0 and less than 25 days.</p>

4. Click **[OK]** when done.

Protect Against SQL Attack Assertion

The *Protect Against SQL Attack* assertion allows you to configure the Gateway to help protect a web service against specific, common SQL injection threats. When added to a policy, this assertion inspects the request message for the specific patterns of characters or keywords that are associated with these potential SQL injection attacks. If a match is found in the message, then the request is blocked from further processing.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Note: When scanning a message element (for example, with the [Evaluate Request XPath](#) or [Evaluate Response XPath](#) assertions), select the "Invasive SQL Attack Protection" option (the "standard" protection is not sufficient). (3) The assertion does not examine attachments to messages. Protection for attachments is provided by the optional "Scan Using Symantec Antivirus Assertion" on page 749.

SQL Injections Detected

The Protect Against SQL Attack assertion will respond as follows to these SQL injections:

Table 223: SQL Attack Protection - response to injections

Injection Type	Desired Response	Assertion Response
Single Tick Only	Detect a parameter value in requests and inbound HTTP headers that consists of only one single tick (''); this can be used to initiate an injection attack.	Select the Invasive SQL Injection Attack Protection check box to check the entire message body (excluding HTTP headers) for these meta characters: ' # -- If any of the non-Invasive protection options are selected, only the text and CDATA portions of a message will be checked for the above characters.
Single Tick Start	Detect a parameter value in requests and inbound HTTP headers that start with a single tick (''); this can be used to initiate an injection attack.	Same as above.
SQL Commands	Detect and block any request message containing standard SQL commands, including ALTER DATABASE, ALTER	The assertion does not search for any of the listed commands.

Injection Type	Desired Response	Assertion Response
	TABLE, ALTER VIEW, CREATE DATABASE, CREATE PROCEDURE, CREATE SCHEMA, CREATE TABLE, CREATE VIEW, DELETE FROM, DROP DATABASE, DROP PROCEDURE, DROP TABLE, DROP VIEW.	
SQLServer 2000	Detect and block any request message containing the default stored procedures included with SQLServer 2000.	The assertion searches for EXEC procedure when MS SQL Server is enabled.
SQLTABLE	Detect SQL Server 2000 Master database default table names in request messages.	The assertion does not search for any specific table names.
Oracle	Detect and block any request message containing the system tables, default stored procedures, or factory passwords included with Oracle.	In the Oracle case, the assertion searches for the strings 'to_timestamp_tz', 'tz_offset', 'bfilename'. No search for other table names and or default stored procedures.
XML Entity Expansion (encoded)	Detect and block any request message containing the "<!ENTITY" XML entity expansion sequence.	SOAP services scan for and deny any DOCTYPE declaration, so the ENTITY item will not pass. However, XML/REST services will let this pass.
XML Entity Expansion (unencoded)	Detects and blocks any message containing an XML "<!ENTITY" element. Run this rule on requests. When a match is found, log an event and reject the message	Same response as above — encoding '<' as '<' does not change the Gateway response.
SQLTABLE_Updated	Allow commonly used words in request messages such as: VIEWS, TABLES, ROUTINES, DOMAINS, PARAMETERS and COLUMNS.	No checking is done for any of these SQL keywords.

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- When adding the assertion, the **SQL Attack Protection Properties** automatically appear; when modifying the assertion, right-click **<target>: Protect against SQL**

Attack in the policy window and select **SQL Attack Protection Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

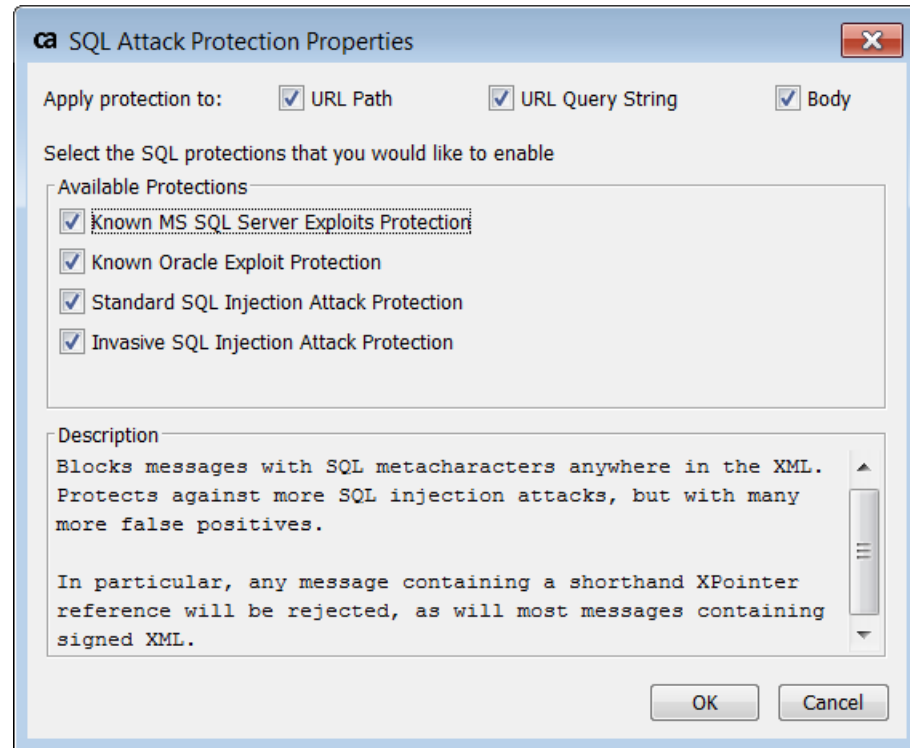


Figure 239: SQL Attack Protection Properties

3. Select the protection.

W A R N I N G

The following options are a minimal starting point for protecting against common SQL attacks. More effective protection with fewer false positives can be obtained by customizing the policy to the specific vulnerabilities of the service being protected. For example, the [Validate XML Schema](#) assertion can be used to block SQL metacharacters from only those elements that are known to be at risk of being misused by the backend service. Each threat protection option requires a separate inspection of the request message; thus, selecting multiple options may increase the message processing time. Do not select the product-specific protections—MS SQL Server or Oracle exploit protection—unless the associated product is used by the protected service.

Table 224: SQL Attack Protection settings

Setting	Description
Apply Protection to:	Specify where to apply the protection: <ul style="list-style-type: none"> • URL Path: Select this to protect the URL Path. • URL Query String: Select this to protect the query parameters in the URL. • Body: Select this to protect the body of the message.
Known MS SQL Server Exploits Protection	Block messages that contain patterns recognized as potential MS SQL Server exploits.
Known Oracle Exploit Protection	Block messages that contain patterns recognized as potential Oracle SQL exploits
Standard SQL Attack Protection	<p>Block messages that contain a single-quote ('), hash mark (#), or string (--) inside the element text or CDATA section (the characters are permitted in message attributes).</p> <p>This option effectively protects against many SQL injection attacks but may result in a small number of false positives. For example, a message containing the name "O'Reilly" will be rejected as a possible attack due to the "'" character.</p> <p>IMPORTANT: If the service being protected is potentially vulnerable to altered XML attributes (for example, it uses XML attributes in SQL statements), then the Standard SQL Attack Protection may not be sufficient.</p>
Invasive SQL Attack Protection	<p>Block messages that contain a single-quote ('), hash mark (#), or string (--) anywhere within the message.</p> <p>This option effectively protects against many SQL injection attacks, but will result in a large number of false positives that cause messages to be incorrectly rejected. For example, any message containing a shorthand XPointer reference will be rejected, as will most messages containing signed XML (e.g., WSS Signature).</p> <p>IMPORTANT: The Invasive SQL Attack Protection option is a "catch all" approach that should only be used when a potentially vulnerable service must be well-protected and where lower performance and a higher false positive rate are acceptable.</p>

4. Click **[OK]** when done.

Scan Using ICAP-Enabled Antivirus Assertion

The *Scan Using ICAP-Enabled Antivirus* assertion lets the Gateway connect to an antivirus server that supports the ICAP protocol, such as McAfee®, Sophos®, or Symantec™.

Ensure your antivirus server is enabled for the ICAP protocol in order to use this assertion. Please consult the documentation for your antivirus server for more information.

Tips: (1) The cluster properties *icap.channelIdleTimeout* and *icap.threadPoolSize* can be used to further adjust the assertion behaviour, if necessary. (2) You can monitor the number of connections to the antivirus server by using this “netstat” command on the Gateway machine: **netstat -an -t 1 | grep ":1344"**. If it is necessary to further limit the number of requests, use the "Apply Rate Limit Assertion" on page 573.

Note for McAfee Users: When using the Scan Using ICAP Antivirus assertion with McAfee VirusScan, the McAfee server must be configured to add virus information to the ICAP response headers. Please contact your McAfee administrator or CA Technical Support if you require assistance.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Context Variables Created by This Assertion

The Scan Using ICAP-Enabled Antivirus assertion sets the following context variables with information about a detected virus. **Note:** The context variables are not set if no viruses are found.

Table 225: Context variables created by the ICAP-Enabled Antivirus assertion

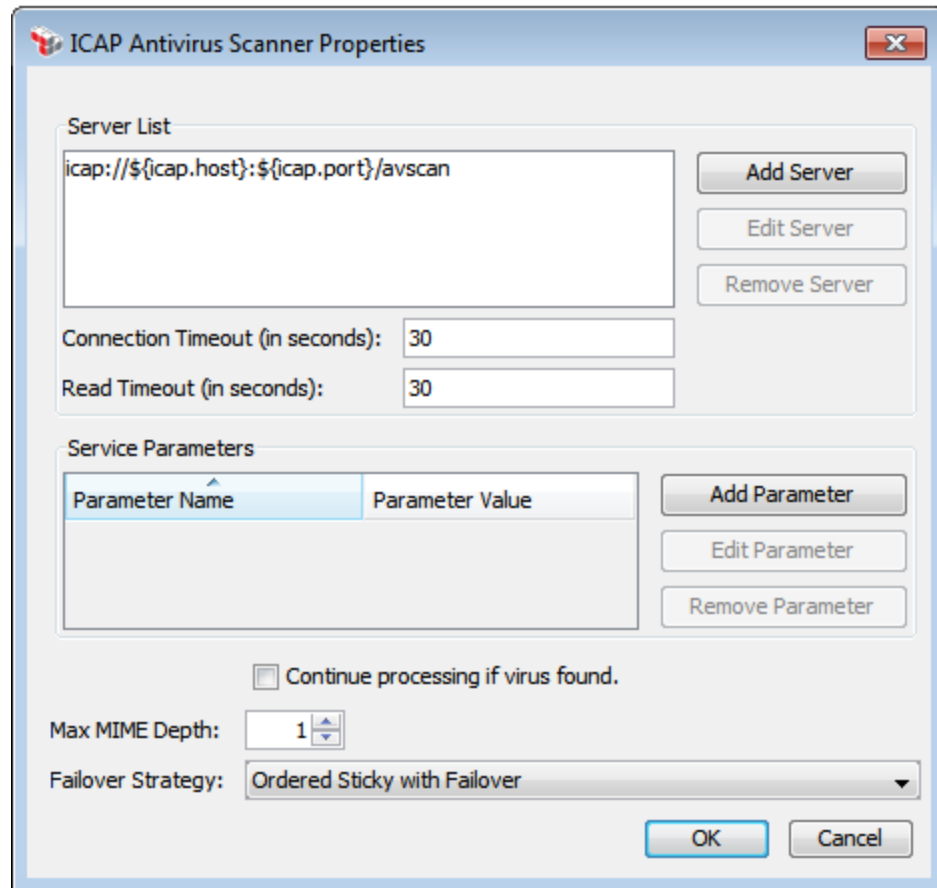
Variable	Description
icap.response.infected	Lists the infected part ID, content ID, filename or context variable name.
icap.reponse.header.names.X	Header names as returned by the ICAP server, where 'X' is an index that corresponds to the index of the infected part.
icap.reponse.header.values.X	Header values as returned by the ICAP server, where 'X' is an index that corresponds to the index of the infected part.
icap.reponse.header.value.X.headerName	The value of the specified header name for the infection part 'X'.

All the ICAP-Enabled context variables are multivalued, to accommodate multiple viruses found.

Using the Assertion

1. Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Scan Using ICAP-Enabled Antivirus** in the policy window and select **ICAP Antivirus Scanner Properties** or double-click the assertion in the policy window. The assertion properties are displayed.




The dialog box titled "ICAP Antivirus Scanner Properties" contains the following sections:


- Server List:** A text area containing the URL `icap://{icap.host}:{icap.port}/avscan`. To the right are buttons for "Add Server", "Edit Server", and "Remove Server".
- Timeouts:** Two input fields: "Connection Timeout (in seconds):" with value 30, and "Read Timeout (in seconds):" with value 30.
- Service Parameters:** A table with two columns: "Parameter Name" and "Parameter Value". To the right are buttons for "Add Parameter", "Edit Parameter", and "Remove Parameter".
- Options:** A checkbox labeled "Continue processing if virus found." which is currently unchecked.
- Max MIME Depth:** A spinner box set to 1.
- Failover Strategy:** A dropdown menu set to "Ordered Sticky with Failover".
- Buttons:** "OK" and "Cancel" buttons at the bottom right.

Figure 240: ICAP Antivirus Scanner Properties

3. Configure the **Server List** as follows:

Table 226: Configuring the ICAP Antivirus server list

To...	Do this...
Add a server 	<p>At least one server must be defined.</p> <ol style="list-style-type: none"> 1. Click [Add Server]. 2. Enter the URL of the ICAP Server. You may also reference context variables. 3. Optionally, click [Test Connection] to test the connection to

To...	Do this...
	<p>the specified server.</p> <p>Note: If context variables are used, the connection cannot be tested and the URL cannot be validated.</p> <p>4. Click [OK].</p>
Edit a server 	<p>1. In the Server List, select the server you wish to modify.</p> <p>2. Click [Edit Server].</p> <p>3. Modify the fields as required. See <i>Add a server</i>, above, for more information.</p> <p>4. Click [OK].</p>
Remove a server	<p>1. In the Server List, select the server you wish to remove.</p> <p>2. Click [Remove Server].</p>




4. Configure the timeout settings: 
- **Connection Timeout (in seconds):** Enter the number of seconds (between 1 and 3600) before the connection times out. You may reference context variables.
 - **Read Timeout (in seconds):** Enter the number of seconds (between 1 and 3600) for the read timeout. You may reference context variables.
5. Optionally, configure any **Service Parameters** required by the antivirus server:

Table 227: Configure the ICAP antivirus service parameters

To...	Do this...
Add a parameter 	<p>1. Click [Add Parameter].</p> <p>2. Configure the dialog as follows:</p> <ul style="list-style-type: none"> • Parameter Name: Enter the name of the parameter as per the antivirus server documentation. You may reference context variables. • Parameter Value: Enter a value for the parameter as per the antivirus server documentation. You may reference context variables. • Parameter Type: Specify whether the parameter is a Header or a Query. <p>3. Click [OK].</p>
Edit a parameter 	<p>1. Select the parameter to modify.</p> <p>2. Click [Edit Parameter].</p> <p>3. Configure the dialog as required. See <i>Add a parameter</i>, above, for more information.</p>

To...	Do this...
	4. Click [OK] .
Remove a parameter	1. Select the parameter to remove. 2. Click [Remove Parameter] .

6. Specify how the assertion should behave when a virus is found:
 - Select the **Continue processing if virus found** check box to allow the assertion to continue when a virus is detected.
 - Clear the **Continue processing if virus found** check box to fail the assertion when a virus is found. This setting is the default.
7. In the **Maximum MIME Depth** field, specify how deep the assertion should traverse in the event of nested multipart.
8. Choose the failover strategy:

Table 228: Failover Strategies for ICAP antivirus

Failover Strategy	Description
Ordered Sticky with Failover	<p>The Gateway sends service messages to the first server in the list until that server does not respond (fails). When this occurs, the next server in the list is used.</p> <p>Tip: The cluster property <i>io.failoverServerRetryDelay</i> controls the delay before the Gateway retries a failed server. The default is to wait 15 minutes when using the "Ordered Sticky with Failover" strategy.</p>
Random Sticky with Failover	<p>The Gateway chooses a server randomly at the beginning of each session and uses it for the duration of the session. If the chosen server fails, another server is chosen at random.</p>
Round Robin	<p>The Gateway rotates through the server list on a request-by-request basis (round-robin) from the first server, to the second server, and so on. When the end of the server list is reached, the cycle continues from the top of the list.</p> <p>Tip: The cluster property <i>io.failoverServerRetryDelay</i> controls the delay before the Gateway retries a failed server. The default is to wait 5 minutes when using the "Round Robin" strategy.</p>

9. Click **[OK]**.

Scan Using Sophos Antivirus Assertion

The *Scan Using Sophos Antivirus* assertion is an optional assertion that can be added to the Gateway. When installed, this assertion enables the Gateway scan all message attachments in the request using Sophos Antivirus running on another machine. This assertion has multiple failover strategies to support installations with multiple machines running Sophos Antivirus.

When this custom assertion package is installed, the Scan Using Sophos Antivirus assertion appears in the Policy Manager.

The Administrator is responsible for installing and configuring the Sophos Custom Assertion package on the Gateway. For more information, refer to the *Custom Assertion Installation Manual*.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Tip: Consider using the Scan Using ICAP-Enabled Antivirus assertion for greater flexibility in checking messages for viruses.

Context Variables Created by This Assertion

The Scan Using Sophos Antivirus assertion sets the following context variables with information from the Sophos response.

Table 229: Context variables created by the Sophos Antivirus assertion

Variable	Description
sophos.scan.virus.name	Name of the virus detected
sophos.scan.virus.type	Type of virus detected
sophos.scan.virus.location	Location where virus was detected
sophos.scan.virus.disinfectable	Indicates whether the virus can be disinfected
sophos.scan.virus.count	Number of viruses detected

All the Sophos context variables (with the exception of *sophos.scan.virus.count*) are multivalued, to accommodate multiple viruses found.

Using the Assertion

1. Do one of the following:

- To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Scan using Sophos Antivirus** in the policy window and select **Sophos Antivirus Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

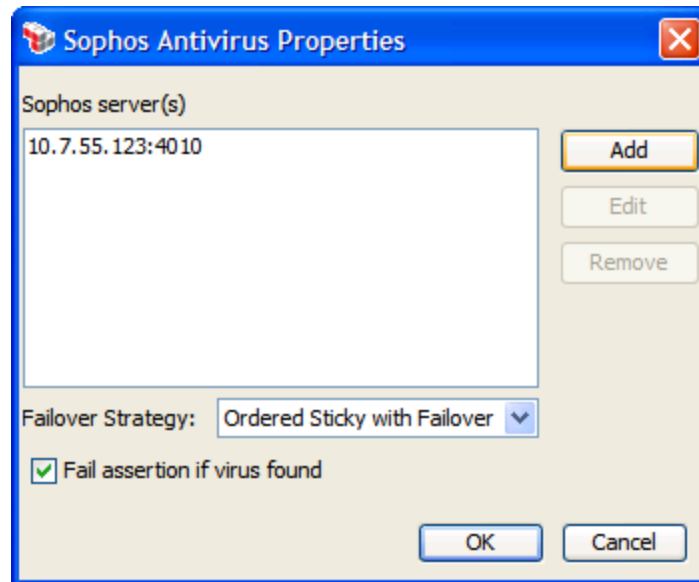


Figure 241: Sophos Antivirus Properties

3. Configure the dialog as follows:

Table 230: Sophos Antivirus settings

Setting	Description
Sophos server(s)	<p>The list of Sophos servers and their associated port numbers.</p> <ul style="list-style-type: none"> • Click [Add] to add a new server to the list. In the Add Host/Port dialog that appears, enter the Host name or IP address. Specify a Port number, between 0 and 65535. The default port is 4010. • Click [Edit] to modify the details for the selected server. • Click [Delete] to remove a server from the list.
Failover Strategy	<p>When there are multiple Sophos servers, choose a failover strategy to use in case a server fails to respond:</p> <ul style="list-style-type: none"> • Ordered Sticky with Failover: The assertion uses the first Sophos server in the list until that server does not respond (fails). When this occurs, the next server in the list is used. This setting is the default.

Setting	Description
	<ul style="list-style-type: none"> • Random Sticky with Failover: The assertion chooses a Sophos server randomly at the beginning of each session and uses it for the duration of the session. If the chosen server fails, another server is chosen at random. • Round Robin: The assertion rotates through the list of Sophos servers beginning with the first server, to the second server, and so on. When the end of the server list is reached, the cycle continues from the top of the list.
Fail assertion if virus found	<p>Select this check box to fail the assertion if a virus is found. Clear this check box to not fail the assertion if a virus is found.</p> <p>IMPORTANT: If you choose not to fail the assertion, ensure there is policy logic to check the context variables created by this assertion. Otherwise, viruses will be detected but not stopped.</p>

4. Click **[OK]**.

Validate or Change Content Type Assertion

The *Validate or Change Content Type* assertion can be used to validate or change the Content-Type of any target message. You can target specific parts of a multi-part MIME message.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **<target>: Validate Content Type** or **<target>: Change Content Type to <ContentType>** in the policy window and select **Content Type Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

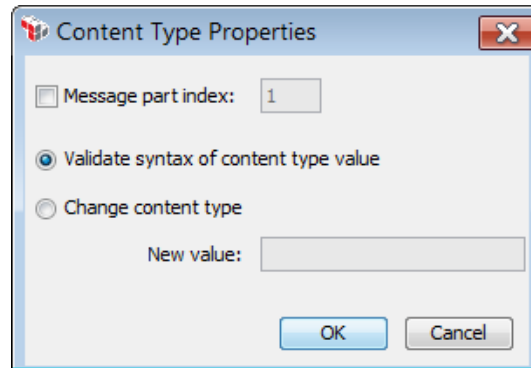



Figure 242: Content Type Properties

3. Configure the properties as follows:

Table 231: Rate Limit settings

Setting	Description
Message part index	<p>Select this check box to target the Content-Type in a specific MIME part in the message.</p> <p>Clear this check box to target the Content-Type in the main/root MIME part of the message (i.e., MIME part '1').</p>
Validate syntax of content type	<p>Select this option to validate the syntax of the value of the Content-Type of the targeted message or message part.</p> <ul style="list-style-type: none"> If the validation succeeds, the assertion succeeds. If the validation fails, the assertion fails and returns the assertion status code 601 ("Error in assertion processing.")
Change content type	<p>Select this option to modify the Content-Type value of the targeted message or message part.</p> <p>Note: The modification takes effect immediately, but any previous message processing based on the old Content-Type will not be undone.</p>
New value 	<p>When changing the Content-Type, enter the new value here. You can also enter a context variable to set the value at the time of policy execution.</p> <p>Note: The value must be a valid and complete MIME type as defined by RFC 2045 and 2046.</p>

4. Click **[OK]** when done.

Validate JSON Schema Assertion

The *Validate JSON Schema* assertion is used to validate JSON (JavaScript Object Notation) data against a JSON schema. Specifically it will:

- validate JSON data structure
- validate JSON data property types
- validate JSON data property values

The JSON schema can either be defined within the assertion, or the assertion can monitor a URL or extracted the URL from a Content-Type or Link header.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

To adjust how JSON schemas are cached, refer to these cluster properties:

```
json.schemaCache.maxAge
json.schemaCache.maxDownloadSize
json.schemaCache.maxEntries
```

Tips: (1) If the JSON schema validation fails, the reason is stored in the `$(jsonschema.failure)` context variable. (2) Place a "Protect Against JSON Document Structure Threats Assertion" on page 678 before this assertion to protect against DOS attacks

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the JSON Schema Validation Properties automatically appear; when modifying the assertion, right-click **<target>: Validate JSON Schema** in the policy window and select **JSON Schema Validation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

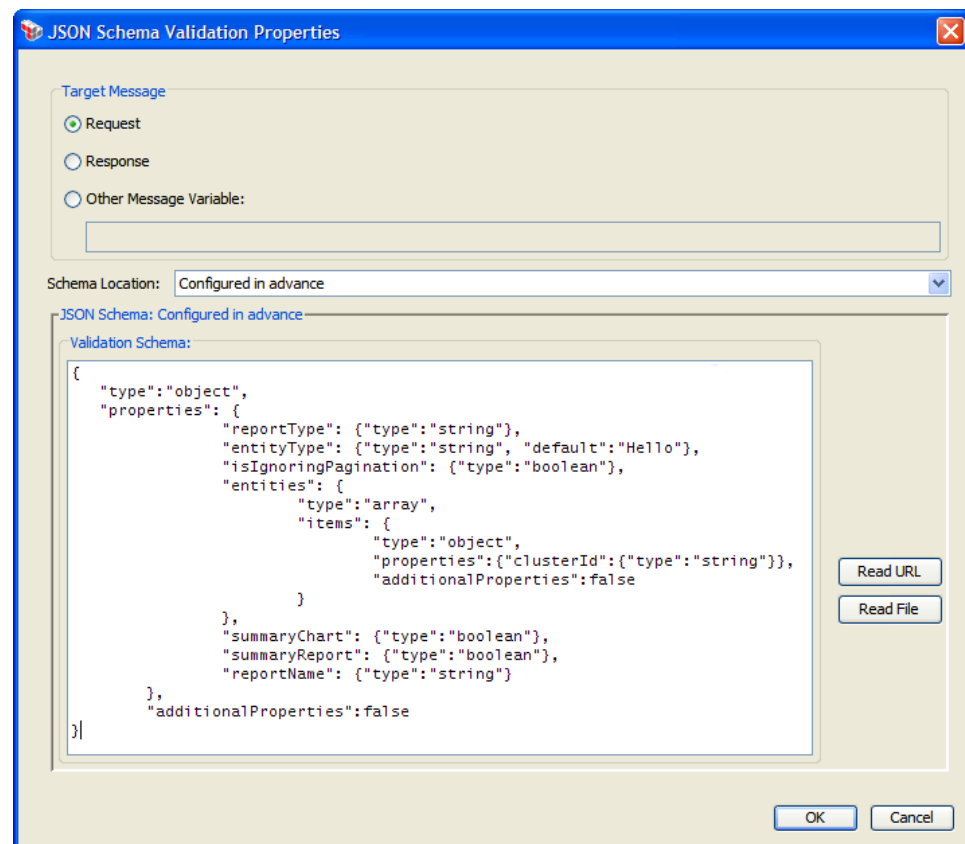




Figure 243: JSON Schema Validation Properties

3. Specify the target message to be validated:
 - **Request:** Select this to validate the request message. This is the default setting if the assertion is positioned before the routing assertion in the policy.
 - **Response:** Select this to validate the response message. This is the default setting if the assertion is positioned after the routing assertion in the policy.
 - **Other Variable:** Select this to validate JSON content stored in a context variable. This variable will normally be either a String or a Message variable with Content-Type 'application/json' or another Content-Type that allows text. This variable must be predefined or has been set in the policy prior to the Validate JSON Schema assertion. For more information on Message variables, see Context Variables in the *Layer 7 Policy Manager User Manual*.

Tip: The message target can also be set outside of the assertion properties. For more information, see "Selecting a Target Message" on page 153.

4. From the **Schema location** drop-down list, specify where the schema is coming from:

Table 232: Configuring the JSON schema based on location

Setting	Description
Configure in advance	<p>Select this option to define a JSON schema directly.</p> <ol style="list-style-type: none"> 1. Specify the JSON schema using any of the following methods: <ul style="list-style-type: none"> • Manually type the code into the Validation Schema box or copy and paste the code from another source. Variables may be used. The assertion will check the input for correct JSON structure, but it will not validate any variables entered. <p></p> <p>Tip: You can use the ".mainpart" suffix on variables of type Message and with Content-Type 'application/json'. For more information about this suffix, see "Context Variable Data Types" under Context Variables in the <i>Layer 7 Policy Manager User Manual</i>.</p> • Load the schema from a URL by clicking [Read URL] and then specifying the URL. <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> <ul style="list-style-type: none"> • Load the schema from a local file by clicking [Read File] and then browsing to the appropriate file. <p>Tip: The schema maximum size is controlled by the <i>schemacache.maxSchemaSize</i> cluster property.</p> 2. Review the content of the Validation Schema box and edit if necessary.
Monitor URL for latest value 	<p>Select this option to specify a URL for the JSON schema. The Gateway monitors the external resources for changes over time.</p> <p>Type the address in the URL to monitor field. The URL may contain context variables that will be resolved at run time. By default, Gateway will issue an <i>If-Modified-Since</i>: HTTP request for this URL approximately every 5 minutes while the schema is in use.</p> <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> <p>Tip: The monitor time interval is controlled by the <i>json.schemaCache.maxAge</i> cluster property.</p>
Retrieve Schema URL from Content-Type or Link Header	<p>Select this option to retrieve the JSON schema URL from either a Content-Type profile parameter in the header or from a Link header.</p> <p>Example of a MIME type parameter: 'profile':</p>

Setting	Description
	<p><i>Content-Type: application/json; profile=http://json.com/my-hyper-schema</i></p> <p>Example of a "describedby" HTTP header:</p> <p><i>Link: <http://json.com/my-hyper-schema>; rel="describedby"</i></p> <ul style="list-style-type: none"> • Use [Add] to add as many regular expressions as necessary to determine if a URL belongs to the set of white-listed URLs. • Use [Edit] to modify any of the regular expressions. • Use [Delete] to remove a regular expression from the list. • Select the Skip validation... check box to allow the assertion to succeed if there is no schema URL in the message. Clear this check box to always check for a schema URL (the assertion will fail if not found). <p>Note: The Content-Type parameter is checked first; if a URL is not found, then the header values are checked next.</p>

5. Click **[OK]** when done.

Validate OData Request Assertion

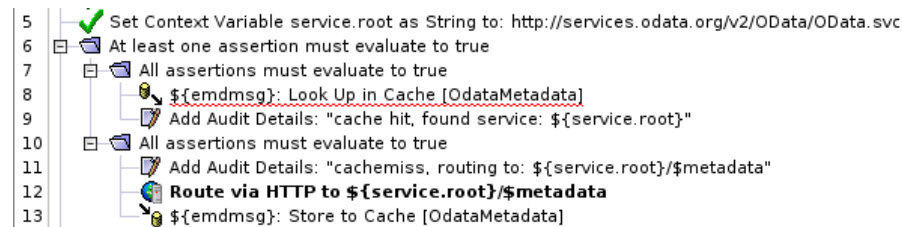
The *Validate OData Request* assertion is used to validate OData (Open Data Protocol) request messages using the Service Metadata Document (SMD) exposed by an OData service. The resource URI, query string, and (optionally) the payload of the request are analyzed to ensure they are well-formed, adhere to the OData v2.0 specifications, and apply to the target service.

The Validate OData Request assertion supports OData version 2.0.

The OData request may be stored in the default Gateway request, response, or in a custom context variable. To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Retrieving the Service Metadata Document

The following sample policy provides an example on how to retrieve and cache the Service Metadata Document:



Notes and Limitations

Observe the following notes about this assertion:

- The assertion will test JSON payloads to ensure their content is suitable for the request type (for example, the request resource URI for a create entry operation points to collection "X", but the entry type described in the message payload is of type "Y") and will fail if it is not suitable. This test is not performed for Atom payloads.
- JSON payloads containing open type entries will fail to validate. This validation failure does not occur with Atom payloads.
- Batch request payloads cannot be validated. Attempting to validate a batch request will cause the assertion to fail.
- Payloads for function import requests cannot be validated.
- All HTTP methods are considered valid for function import requests.
- The Service Metadata Document must be made available in a context variable.
- Matrix parameters in request URIs is not supported and will fail to validate.
- OData versions 3.0 and 4.0 are not supported.
- Validation of requests using method tunnelling is not supported.

Context Variables Created by This Assertion

The Validate OData Request assertion sets the following context variables. **Note:** The default <prefix> is "odata" and can be changed in the assertion properties (Figure 244).

Table 233: Context variables created by Validate OData Request assertion

Context variable	Description
<prefix>.query.count	Returns a Boolean value indicating the presence of the count option; example: "true"
<prefix>.query.top	Returns the top option value; example, "10"
<prefix>.query.filter	Returns the filter expression in a multivalued variable; example: "length, CompanyName, 19, eq"
<prefix>.query.skip	Returns the skip option value; example: "10"
<prefix>.query.orderby	Returns the Orderby expression in a multivalued context variable; example: "Rating, Category, Name, desc"
<prefix>.query.expand	Returns the Expand expression; example: "Category, Suppliers"
<prefix>.query.format	Returns the format media type; example: "json"
<prefix>.query.inlinecount	Returns the Inlinecount setting; example: "allpages"
<prefix>.query.select	Returns the Select expression; example: "Rating, Category, Name"
<prefix>.query.customoptions	Returns the custom query options in a multivalued variable; example: ["x=y", "a=b", "f=g"]
<prefix>.query.pathsegments	Returns the resource path segments in a multivalued variable; example: ["Categories(1)", "\$links", "Products"]




Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click **<target>: Validate OData Request** in the policy window and select **OData Request Validation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 244: OData Validate Request Properties

3. Configure the properties as follows:

Table 234: Odata Validate Settings

Setting	Description
Service Metadata 	Specify a context variable that contains the Service Metadata Document to use for validating the OData request. For more information, see " Retrieving and Caching the Service Metadata Document " on page 700
Resource 	Specify the resource URI to validate against the Service Metadata Document, including the query string. You may reference context variables. Note: Ensure the resource URI is correctly encoded from the client.
HTTP Method 	Choose the HTTP method to use during payload validation. The "<Automatic>" option attempts to locate the method in the HttpRequestKnob in the target message. You may reference a context variable.
Actions	For improved security, following request types are disallowed by default:

Setting	Description
	<ul style="list-style-type: none"> • Allow \$metadata request: Select this check box to allow the client to retrieve the metadata document from the service by requesting the <i>\$metadata</i> URI. • Allow \$value requests: Select this check box to allow the client to retrieve the raw value of the request target by calling the <i>\$value</i> operation. <p>Tip: When the assertion will return "falsified" if it encounters a request type that has been disallowed (see Assertion Status Codes in the <i>Layer 7 Policy Manager User Manual</i>.)</p>
Allowed Operations	<p>Select which OData operations are permitted:</p> <ul style="list-style-type: none"> • GET: Allow or deny the OData retrieve operation. • POST: Allow or deny the OData create operation. • PUT: Allow or deny the OData update operation. • DELETE: Allow or deny the OData delete operation. • MERGE: Allow or deny the OData partial update operation. • PATCH: Allow or deny the OData partial update operation. This method is synonymous with MERGE.
Validate Payload	<p>Select this check box to validate the message payload against the request URI and the Service Metadata Document.</p> <p>Clear this check box to not validate the message payload.</p>
Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>Default: odata</p>

4. Click **[OK]** when done.

Validate XML Schema Assertion

The *Validate XML Schema* assertion allows you to specify a schema for validating a web service or XML application request or response messages. This assertion can be used to protect backend web services against the following threats:

- **XML Parameter Tampering:** All XML parameters in the request are validated to ensure conformance with the XML schema specifications. This is to prevent injection of malicious scripts or content into the request parameters.
- **XDoS Attacks:** The message structure and content are examined to ensure that they are correct.

A message schema is provided by the Gateway administrator. If the service's WSDL contains a schema, then that schema can be extracted to serve as the starting point for the schema used in the Validate XML Schema assertion. This WSDL schema can be extracted in whole or in request or response message-specific parts.

If the schema contains import statements that refer to external schemas, the Policy Manager will attempt to fetch all unresolved schemas in an import tree (for example, a schema referencing another schema) and add them to the global schema table. You can view these imported schemas using the [Manage Global Resources](#) task. If the Policy Manager is unable to resolve a schema (for example, because of a bad URL or URI), you will be prompted to manually add the schema.

Tip: The format of the import statement can affect how it is received by the Gateway. A full URL path is most preferable and is always resolvable (e.g., "http://schema.example.com/test.xsd"). Just the file name is acceptable, provided that the exact name can be located in the Global Schemas stored in the Gateway (e.g., "test.xsd"). Not acceptable are paths containing a specific drive letter (e.g., "f:\test.xsd"), or relative paths such as "../test.xsd".

A policy can contain multiple Validate XML Schema assertions. The runtime application of a schema is determined by its placement in the policy path. If routing has already occurred when the Validate XML Schema assertion initiates, then the schema will be applied to the response message. If routing has not yet occurred, then the schema will be applied to the request message.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Schema Failure in Context Variable

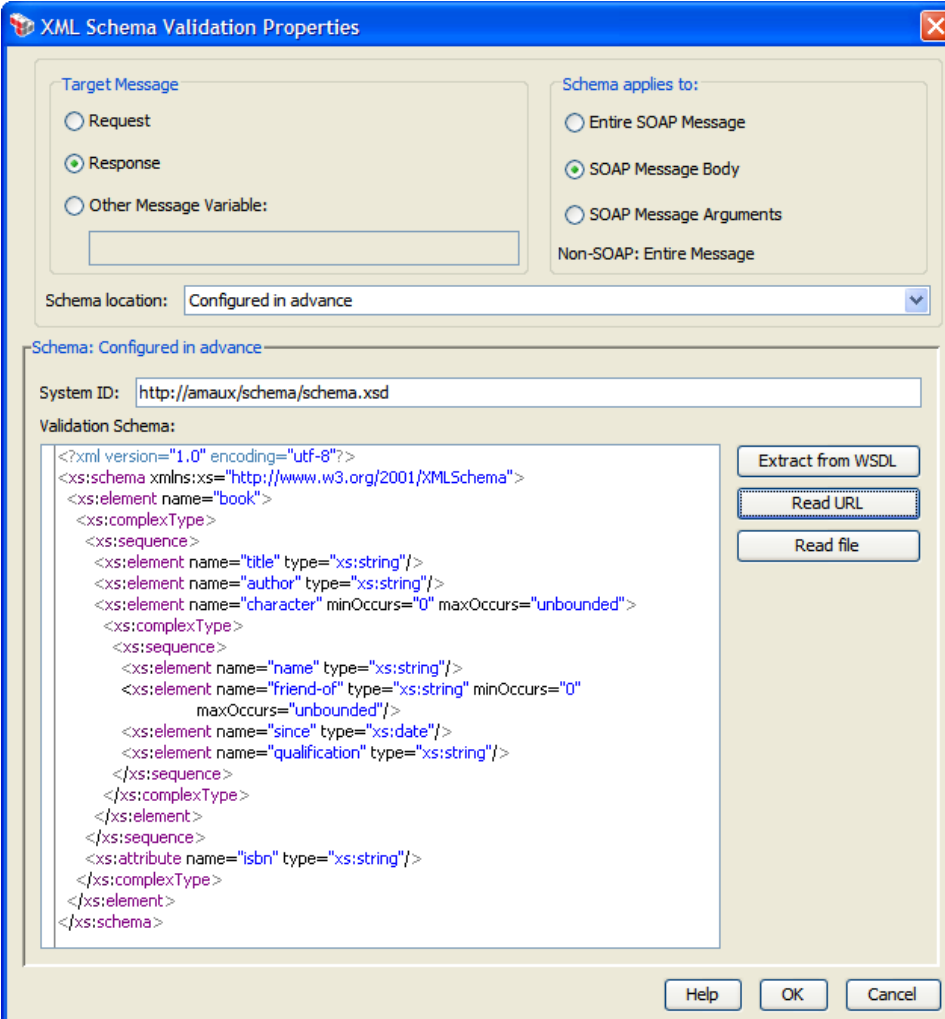
When a schema validation fails, an audit record is created and the reason for failure is placed in the context variable `${schema.failure}`. This makes it possible to reference the failure later in the policy (for example, inclusion in the [Return Template Response to Requestor](#) assertion).

Schemas with Circular References

A "circular reference" occurs when a schema references other schemas that ultimately point back to the original schema. The Policy Manager will fetch all schemas from a destination, circular or not, and add them to the global schemas table.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. When adding the assertion, the Schema Validation Properties automatically appear; when modifying the assertion, right-click **<target>**: **Validate XML Schema** in the policy window and select **XML Schema Validation Properties** or double-click the assertion in the policy window. The assertion properties are displayed.



The dialog box titled "XML Schema Validation Properties" contains the following sections:

- Target Message:**
 - ☐ Request
 - ☒ Response
 - ☐ Other Message Variable:
- Schema applies to:**
 - ☐ Entire SOAP Message
 - ☒ SOAP Message Body
 - ☐ SOAP Message Arguments
 - Non-SOAP: Entire Message
- Schema location:** Configured in advance
- Schema: Configured in advance**
 - System ID:**
 - Validation Schema:**

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="author" type="xs:string"/>
        <xs:element name="character" minOccurs="0" maxOccurs="unbounded"/>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="friend-of" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="since" type="xs:date"/>
            <xs:element name="qualification" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:sequence>
    <xs:attribute name="isbn" type="xs:string"/>
  </xs:sequence>
</xs:schema>
```
 - Buttons:** Extract from WSDL, Read URL, Read file
- Footer:** Help, OK, Cancel

Figure 245: XML Schema Validation Properties

3. Specify the target message to be validated:

- **Request:** Select this to validate the request message. This is the default setting if the assertion is positioned before the routing assertion in the policy.
- **Response:** Select this to validate the response message. This is the default setting if the assertion is positioned after the routing assertion in the policy.
- **Other Message Variable:** Select this to validate a message stored in a context variable of type 'Message'. This variable must be predefined or has been set in the policy prior to the Validate XML Schema assertion. For more information on Message variables, see Context Variables in the *Layer 7 Policy Manager User Manual*.

Tip: The message target can also be set outside of the assertion properties. For more information, see "Selecting a Target Message" on page 153.

4. For SOAP messages, specify the portion of the message that will be validated by the schema. For non-SOAP messages, the schema will be applied to the entire message.

- **Entire SOAP Message**

Schema validation is performed on the entire SOAP envelope.

The schema configured by the policy author in this case should be based on the SOAP envelope schema. It may optionally include definitions that cover the payload of the SOAP headers and/or the SOAP body.

If you need to validate a schema against the SOAP message including any security elements in the header (for example, signature element), you should additionally import the WS-Security schema in your custom schema (for example, <http://schemas.xmlsoap.org/ws/2002/04/secext/secext.xsd>).

- **SOAP Message Body**

Apply the schema to each element under the *soap:Body* element in a SOAP message. This setting is the default.



Note: When importing an RPC/literal-style WSDL using this option, the system will prompt you with: *"The WSDL style seems to indicate that the schema validation should be applied to the body 'arguments' rather than the entire body. Would you like to change the setting accordingly?"* Answer 'Yes' only if you are certain that the web service is RPC/literal-style.

- **SOAP Message Arguments**

Apply the schema to the children elements under the first child element under the *soap:Body*. This is typically used in RPC/literal-style web services where the argument elements themselves are not declared in the schema.

- From the **Schema location** drop-down list, specify where the schema is coming from:

Table 235: Configuring the schema based on location

Setting	Description
Configure in advance	<p>Select this option to define a root schema and all dependencies directly.</p> <ol style="list-style-type: none"> Specify the schema using any of the following methods: <ul style="list-style-type: none"> Manually type the code into the Validation Schema box or copy and paste the code from another source. If the Gateway can detect a schema in the WSDL document, you can click [Extract Schema from WSDL] to import the schema from the WSDL document. A WSDL-based schema is typically only included in document-style web services. Complete the Extract Schema from WSDL dialog in step 6 below. Load the schema from a URL by clicking [Read URL] and then specifying the URL.  <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> <ul style="list-style-type: none"> Load the schema from a local file by clicking [Read File] and then browsing to the appropriate file. <p>The System ID field is automatically populated when opening a resource (from the WSDL, a URL, or a file).</p> <p>Notes: (1) The schema maximum size is controlled by the <i>schemacache.maxSchemaSize</i> cluster property. (2) If the cluster property <i>schema.allowDoctype</i> is set to "true", then the "Configure in advance" XML schema may contain a document type definition (DTD); otherwise, DTDs are not permitted (default).</p> Review the content of the Validation Schema box and edit if necessary. You can right-click within the box for some useful tools to help you edit. For more information, see Using the XML Editor in the <i>Layer 7 Policy Manager User Manual</i>.
Monitor URL for latest value 	<p>Select this option to specify a URL for the root schema. The Gateway loads all the dependencies and then monitors the external resources for changes over time.</p> <p>Type the address in the URL to monitor field. The URL may contain context variables that will be resolved at run time. By default, Gateway</p>

Setting	Description
	<p>will issue an <i>If-Modified-Since</i>: HTTP request for this URL approximately every minute while the schema is in use.</p> <p>Tip: To configure options for the URL (for example, to specify the credentials, SSL, or proxy options), click [HTTP Options] to open the Manage HTTP Options dialog.</p> <p>Note: The schema maximum size is controlled by the <i>schemacache.maxSchemaSize</i> cluster property.</p>
Pick XML Schema from global resources	<p>Select this option to pick the validation schema from the global resources table. Choose the global schema to use from the Selected schema drop-down list. If the schema you require is not listed, click [Manage Global Resources] to define it, or to modify or remove other global resources defined in the system. For more information, see "Managing Global Resources" on page 72.</p>

6. If you chose to extract the schema from a WSDL document, the following dialog appears:

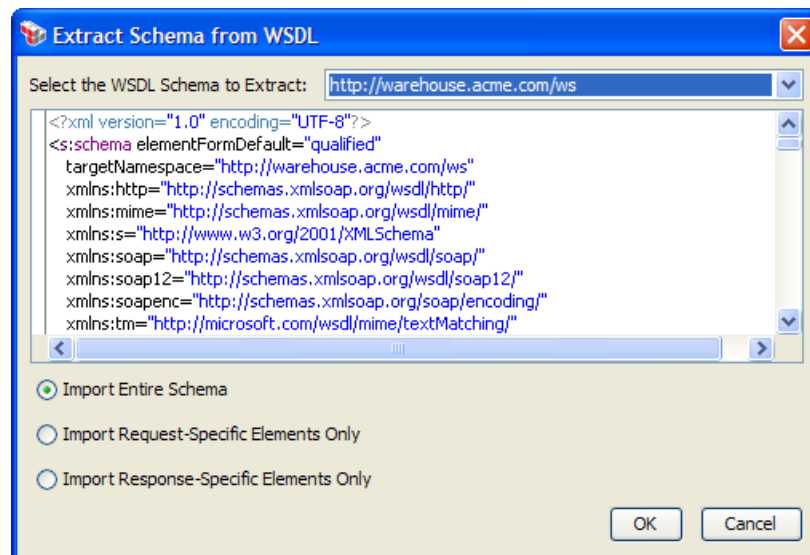


Figure 246: Extract Schema from WSDL dialog

Configure the dialog as follows and then click **[OK]** when done:

Table 236: Extract Schema from WSDL settings

Setting	Description
Select the WSDL Schema to Extract	<p>If the WSDL document contains more than one schema, select the schema to use from the drop-down list. The schema code is displayed in the box below.</p> <p>Note: The Validate XML Schema assertion only takes a single schema</p>

Setting	Description
	as input. If the WSDL contains multiple schemas, it may be necessary to reorganize those schemas into one root schema that references other schemas through import statements. The Policy Manager attempts to retrieve the schemas referenced by the import statements and add them to the global schema table. To view these schemas, see "Managing Global Resources" on page 72.
Import Entire Schema	Extract the entire schema. This setting is the default.
Import Request-Specific Elements Only	Extract only the schema elements particular to request messages.
Import Response-Specific Elements Only	Extract only the schema elements particular to response messages.

7. When a resource with dependencies is opened, you are prompted to confirm whether to import the schema's dependencies as global resources. Select **[Import]** to import the dependencies or **[Skip]** to exclude the dependencies. Select **[Cancel]** to cancel the loading of the resource (whether from the WSDL, a URL, or a file).
8. If you chose **[Import]** in the previous step, all the schema dependencies that will be processed and potentially added as global resources (Figure 247) are listed. Review the list carefully and note the *Action* column for each resource:
 - *Ignore*: The resource will not be imported.
 - *Update*: The resource will update an existing global resource.
 - *Create*: A new global resource will be created for the resource.

Select **[Import]** to update the global resources or **[Skip]** to not update the global resources. Select **[Cancel]** to cancel the import.

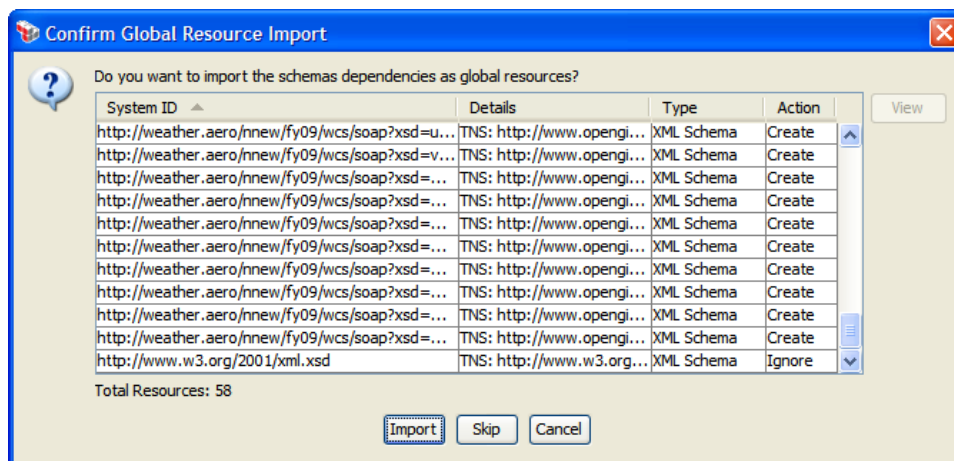


Figure 247: Confirming importing schema dependencies

9. During import, if there are issues that require manual intervention, you will be prompted with a dialog similar to Figure 248

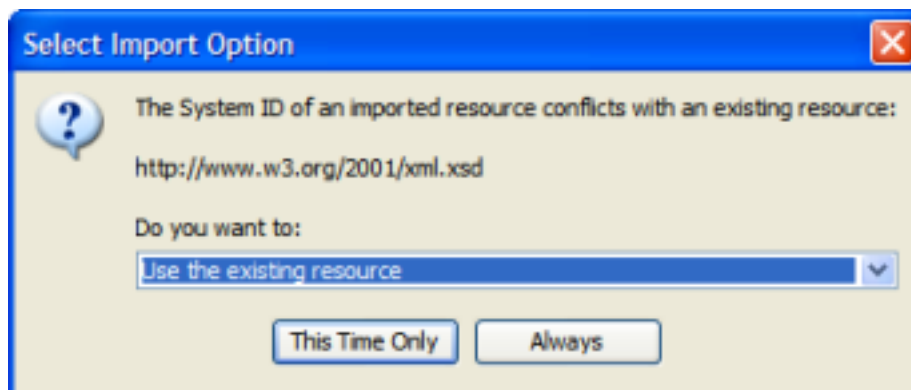


Figure 248: Select Import Option dialog

Select a resolution from the drop-down list, then specify whether:

- **[This Time Only]**: Use the selected action only for this occurrence of the conflict. When another similar conflict occurs, you will be asked again how to resolve it.
 - **[Always]**: Use the selected action for all the conflicts of this type. You will not be prompted for a resolution if another similar conflict occurs during this import.
10. On the XML Schema Validate Properties, click **[OK]** when done. If the dependencies of a configured in advance XML Schema are found then the assertion is added to the policy development window. If the Policy Manager is unable to validate the dependencies, you are prompted to manually add the unresolved schema(s).

Chapter 13: Internal Assertions

In the Policy Manager, the following assertions are available in the Internal Assertions category of the [Assertions] tab:

Collect WSDM Metrics Assertion	711
Convert Audit Record to XML Assertion	713
Handle UDDI Subscription Notification Assertion	714
Manage Gateway Assertion	715
Context Variables Created by Assertion	715
REST Manage Gateway Assertion	716
Context Variables Used by Assertion	717
Subscribe to WSDM Resource Assertion	717

The Internal Assertions are used in Internal Services. For more information, see Working with Internal Services in the *Layer 7 Policy Manager User Manual*.

Collect WSDM Metrics Assertion

The *Collect WSDM Metrics* assertion collects metrics for a specified resource that is interoperable with the Web Services Distributed Management (WSDM) specification. This assertion is automatically added to a policy when the WSDM QosMetrics internal service is published. This assertion forwards *GetMultipleResourceProperties* requests to the CA implementation of the WSDM service.

Note: The Collect WSDM Metrics assertion requires that the cluster property *serviceMetrics.enabled* be set to *true* (default setting).

Using the Assertion

- This assertion is automatically added to a policy when you publish an WSDM QosMetric internal service. You can also manually add it to a policy if it has been deleted. For more information, see "Adding an Assertion" on page 112.

There are no properties or user-definable settings for this assertion. When the Collect WSDM Metrics assertion detects a request directed at a specific WSDM QosMetrics resource, it will collect the information requested and return it to the caller.

Supported Metrics

The following table summarizes the QosMetrics supported by the Gateway.

Table 237: QosMetrics properties supported in Gateway

Property	Description
muws2:OperationalStatus	This relates to the published service's enabled state. If you disable a published service using the Policy Manager for example, the corresponding managed resource will have a value of <i>Unavailable</i> for this property.
mows:NumberOfRequests	The total number of requests classified for the published service corresponding to the managed resource for the period in question.
mows:NumberOfFailedRequests	The number of requests which resulted in a policy violation for the published service corresponding to the managed resource for the period in question.
mows:NumberOfSuccessfulRequests	The number of requests which resulted in a policy success for the published service corresponding to the managed resource for the period in question.
mows:ServiceTime	This is sum of all the total response times (whether or not a request was successful) for a service, across all cluster nodes in a gateway cluster during the period in question.
mows:MaxResponseTime	The maximum value recorded for the corresponding published service to respond to a requestor. If the policy being enforced includes routing the message to one or more backend services, the routing time will be included in this response time.
mows:LastResponseTime	The last value recorded for the corresponding published service to respond to a requestor. If the policy being enforced includes routing the message to one or more backend services, the routing time will be included in this response time.
qosm:Throughput	The number of requests processed for a specific time unit. The time unit is chosen depending on the traffic being handled. For example, instead of indicating 0.0833 req/sec, the Gateway will indicate 5 req/min. The time unit is indicated by the <i>Duration</i> attribute.
qosm:AvgResponseTime	The average value recorded for the corresponding published service to respond to a requestor. If the policy being enforced includes routing the message to

Property	Description
	one or more backend services, the routing time will be included in this response time.

Convert Audit Record to XML Assertion

The *Convert Audit Record to XML* assertion converts the current audit record into XML code as an in-memory DOM tree, overwriting the targeted message.

This assertion is designed to populate the request in an audit sink policy with some XML. The resulting XML is not enclosed in a SOAP envelope.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Note: The Convert Audit Record to XML assertion works only within an audit sink policy. It has no effect when placed in any other policy.

The following are examples of XML code that will be created for each [type](#) of audit record:

System Audits

```
<audit action="Checking" componentId="1290010" id="0" level="800"
  levelStr="INFO" name="Trusted Certificate Store" sequenceNumber="4"
  type="system" xmlns="http://17tech.com/audit/rec">
  <nodeId>node1</nodeId>
  <time>1253670817171</time>
  <message>One or more trusted certificates has expired or is expiring
  soon</message>
  <ipAddress>192.168.1.42</ipAddress>
  <thrown>java.lang.RuntimeException: main record throwable (rest of stack
  trace)</thrown>
</audit>
```

Message Summary Audits

```
<audit type="message" id="0" level="800" levelStr="INFO" name="ACMEWarehouse"
  sequenceNumber="2" xmlns="http://17tech.com/audit/rec">
  <nodeId>node1</nodeId>
  <requestId>req4545</requestId>
  <time>1253669933078</time>
  <message>Message processed successfully</message>
  <ipAddress>3.2.1.1</ipAddress>
  <user id="41123" identityProviderGoid="-2" name="alice"/>
  <details>
    <detail componentId="0" messageId="6" ordinal="0" time="1253669933078">
      <params>
        <param>foomp</param>
      </params>
      <exception>java.lang.IllegalArgumentException: Exception for foomp
      detail
      (rest of stack trace)</exception>
    </detail>
  </details>
  <thrown>java.lang.RuntimeException: main record throwable (rest of stack
  trace)</thrown>
```

```

<authType>HTTP Basic</authType>
<mappingValuesOid>49585</mappingValuesOid>
<operationName>listProducts</operationName>
<requestContentLength>4833</requestContentLength>
<requestSavedFlag>false</requestSavedFlag>
<responseContentLength>9483</responseContentLength>
<responseSavedFlag>false</responseSavedFlag>
<responseHttpStatus>200</responseHttpStatus>
<routingLatency>232</routingLatency>
<serviceOid>8859</serviceOid>
<status>0</status>
</audit>

```

Administrative Audits

```

<audit type="admin" action="U" id="0" level="800" levelStr="INFO" name="testuser"
sequenceNumber="0" xmlns="http://17tech.com/audit/rec">
<nodeId>node1</nodeId>
<requestId>00000000000000003-22b</requestId>
<time>1253669933046</time>
<message>updated</message>
<ipAddress>2.3.4.5</ipAddress>
<user id="1111" identityProviderGoid="-1" name="admin"/>
<details>
<detail componentId="0" messageId="6" ordinal="0" time="1253669933046">
<params>
<param>foomp</param>
</params>
<exception>java.lang.IllegalArgumentException: Exception for foomp
detail
(rest of stack trace)</exception>
</detail>
</details>
<thrown>java.lang.RuntimeException: main record throwable (rest of stack
trace)</thrown>
<entity class="com.17tech.identity.User" oid="1234"/>
</audit>

```

Using the Assertion

- Add the assertion to an audit sink policy as described in "Adding an Assertion" on page 112.

The assertion is added to the policy window; no further configuration is required.

Handle UDDI Subscription Notification Assertion

The *Handle UDDI Subscription Notification* assertion processes subscription results messages from a UDDI registry. If successful, the assertion will initialize an empty SOAP response message. Gateway services can be updated when the related UDDI Business Service changes. For more information, see the [UDDI] tab under Service Properties in the *Layer 7 Policy Manager User Manual* for the UDDI control options.

Note: The Handle UDDI Subscription Notification assertion works only within an internal UDDI notification policy. You will be warned if you attempt to use it in any other policy.

Using the Assertion

Do one of the following:

- Publish a UDDI Notification Service. For more information, see Publish Internal Service Wizard in the *Layer 7 Policy Manager User Manual*.

The assertion is added to the policy window; no further configuration is required.

Manage Gateway Assertion

The *Manage Gateway* assertion processes the request as a management SOAP message and will populate the response message. This assertion is automatically added to a policy when the "Gateway Management" internal service is published. For more information, see Working with Internal Services in the *Layer 7 Policy Manager User Manual*.

This assertion is similar to the "REST Manage Gateway Assertion" on page 716, except it calls the SOAP API service instead of processing the request as a REST message.

Note: The Manage Gateway assertion is only intended for use with the Gateway Management internal service. CA does not support its use in any other scenario.

Context Variables Created by Assertion

The Manage Gateway assertion can optionally set the following context variables. **Note:** The `<prefix>` is defined in the assertion properties.

IMPORTANT: There is no default prefix—if no prefix is specified in the properties, then no context variables will be set by this assertion.

Table 238: Context variables created by Manage Gateway assertion

Variable	Description
<code><prefix>.action</code>	Contains the action that was performed on the entity: Create, Read, Update, Delete, Enumerate
<code><prefix>.entityType</code>	The type of entity receiving the action: Identity Provider, Published Service, Trusted Certificate, etc
<code><prefix>.entityId</code>	The identifier for the entity (if applicable for the operation).
<code><prefix>.message</code>	A message describing the outcome of the operation.

Using the Assertion

1. Do one of the following:

- To add the assertion to the policy development window, publish the internal Gateway Management Service.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Manage Gateway** in the policy window and select **Gateway Management Properties** or double-click the assertion in the policy window. The assertion properties are displayed.

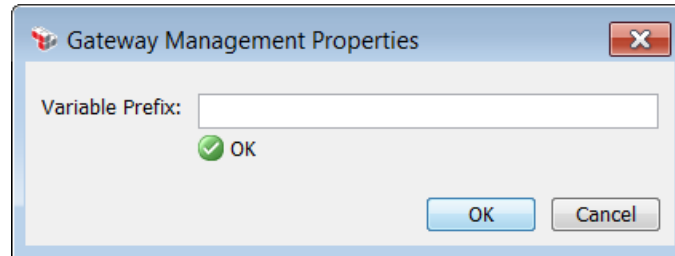


Figure 249: Gateway Management Properties

3. If you want to create the context variables in [Table 1](#), enter a prefix. Leave the field blank if you do not want to set the context variables. You may enter a new prefix or one that has been used before to overwrite existing content. Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.

For an explanation of the validation messages displayed, see Context Variable Validation in the *Layer 7 Policy Manager User Manual*.

4. Click **[OK]**.

REST Manage Gateway Assertion

The *REST Manage Gateway* assertion can let you use REST-style inputs to manage the Gateway. This assertion is automatically added to a policy when the "REST Gateway Management" internal service is published. For more information, see Working with Internal Services in the *Layer 7 Policy Manager User Manual*.

This assertion is similar to the "Manage Gateway Assertion" on page 715, except it calls the REST API service instead of processing the request as a SOAP message.

To learn about selecting the target message for this assertion, see "Selecting a Target Message" on page 153.

Note: The REST Manage Gateway assertion is only intended for use with the REST Gateway Management internal service. CA does not support its use in any other scenario.

Context Variables Used by Assertion

The REST Manage Gateway assertion uses the following context variables only if a Message variable is specified as the target. These variables are not used if the target message is Request or Response.

Table 239: Context variables used by REST Manage Gateway assertion

Variable	Description
restGatewayMan.action	Contains the HTTP method that the assertion will process (for example, "POST").
restGatewayMan.uri	Contains the URI of the entity, including query parameters (for example, "/1.0/storedPasswords?name=MyPassword").

Using the Assertion

- Publish the "REST Gateway Management" internal service.

Complete documentation on how to use the REST Management API can be found online at this location:

https://

<GatewayHostName>:<port>/<GatewayRESTRoutingURI>/1.0/doc/home.html

Log in with your Gateway credentials when prompted.

Subscribe to WSDM Resource Assertion

The *Subscribe to WSDM Resource* assertion allows you to send subscription requests to a specified resource that is interoperable with the Department of Defence Joint Web Services Distributed Management (WSDM) Specification. This assertion is automatically added to a policy when the WSDM Subscription internal service is published. This assertion recognizes the methods: *Subscribe*, *Renew*, *Unsubscribe*.

You can optionally specify an outbound policy if you wish to apply a security policy and/or provide access to the trust store for outbound messages. Eligible outbound policies are those of type "Internal Use Policy", with the tag "WSDM-notification". For more information, see "Creating a Policy" on page 21.

Tip: WSDM subscription notifications can be enabled or disabled using the *WSDM.notification.enabled* cluster property.

The following are some technical limitations to subscribing to an WSDM resource:

- IRIs are not specifically supported and validations are not performed.
- SOAP and WSA namespaces are not accepted in endpoint reference parameters.
- Endpoint reference parameters are limited to 16KB.

Using the Assertion

1. This assertion is automatically added to a policy when you publish an WSDM Subscription internal service. You can also manually add it to a policy if it has been deleted; see "Adding an Assertion" on page 112.
2. When manually adding the assertion, the **WSDM Subscription Properties** automatically appear. You can also access the properties by right-clicking the assertion in the policy window and selecting **WSDM Subscription Properties** or double-click the assertion in the policy window.

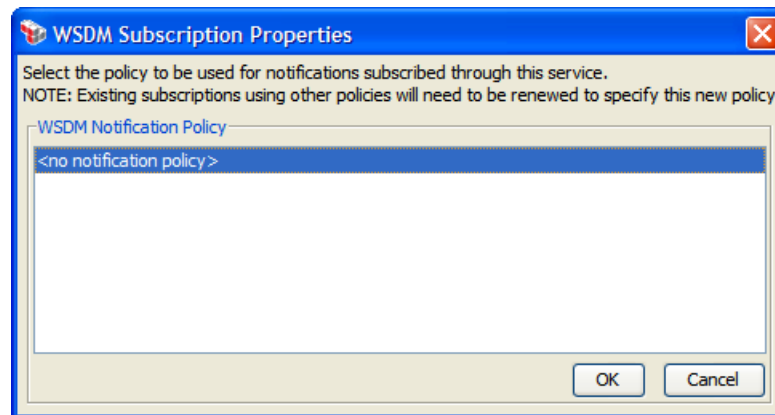


Figure 250: WSDM Subscription Properties

3. Select an outbound policy to be applied to notification messages. If an outbound policy is not required, select "**<no notification policy>**" and the notification will be routed to the subscriber without further processing. To add policies to this list, [create](#) a policy of type "Internal Use Policy", with the tag "WSDM-notification".

Notes: If you change notification policies, this will be applied only to new subscriptions. To update existing subscriptions to use the new policy, you must send a "Renew" message to renew the existing subscriptions. If a policy used by the Subscribe to WSDM Resource assertion is deleted, you must select another policy to use, otherwise the assertion will fail at runtime. Be sure to send a "Renew" message to existing subscriptions after selecting a new policy.

4. Click **[OK]** when done.

Chapter 14: Custom Assertions

The Custom Assertions category lists any supplemental custom assertions that have been purchased from CA. It also lists any custom assertions that were created by third parties.

Tip: Please contact CA Technologies if you are interested in acquiring a custom assertion or to obtain the Custom Assertion SDK that you can use to create your own custom assertions.

The following custom assertions are available as optional add-ons to the Policy Manager:

Access Resource Protected by JSAM Assertion	720
Context Variables Created by This Assertion	720
Access Resource Protected by Oracle Access Manager Assertion	722
Context Variables Created by This Assertion	722
Authenticate using Tivoli Access Manager Assertion	727
Usage Rules	727
Using the Assertion	728
Troubleshooting	730
Authenticate with SiteMinder R12 Protected Resource Assertion	730
Context Variables Created by This Assertion	731
Execute Salesforce Operation Assertion	734
Context Variables Created by This Assertion	735
Using the Assertion	737
Creating Objects	738
Updating Objects	740
Retrieving Objects	742
Retrieving Modified Objects	744
Retrieving Deleted Objects	745
Executing Queries	747
Searching Objects	748
Exporting/Importing Policies	749
Scan Using Symantec Antivirus Assertion	749

Each custom assertion is purchased separately and must be installed on the Gateway before they are available from the Policy Manager. Detailed instructions for installing and configuring the custom assertions are provided in the *Custom Assertions Installation Manual*.


For information on purchasing a custom assertion, please contact CA Technologies.

Access Resource Protected by JSAM Assertion

Installing and configuring the Sun Java System Access Manager Custom Assertion package in the Gateway enables the *Access Resource Protected by JSAM* assertion in the Policy Manager. This assertion allows a policy to use the Single Sign-On (SSO) and Policy Service from an existing *Sun® Java™ System Access Manager 7.0 or 7.1* deployment.

The Administrator is responsible for installing and configuring the Sun Java System Access Manager Custom Assertion package on the Gateway. For more information, refer to the *Custom Assertion Installation Manual*.

Notes: (1) You may receive an HTTP Basic authentication warning when the Access Resource Protected by JSAM assertion is used with these assertions: [Require XPath Credentials](#), [Require FTP Credentials](#), or [Require WS-Security UsernameToken Profile Credentials](#). You may ignore this policy validation warning. (2) If the incoming request is coming through a Securespan XML VPN Client, be sure the Pass Through HTTP Cookies check box has been set on the [XML VPN Client Policy] tab of the Gateway Account properties.

Note: When running this assertion in the browser client, a triangular warning icon () may appear next to the dialog box when the assertion properties is displayed. You may ignore this icon.

Context Variables Created by This Assertion

The user attributes for a successfully authenticated user are available through the following context variables:

Table 240: Context variables created by Access Resource Protected by JSAM assertion

Attribute	Context Variable
UID	<code>\${jsam.attributes.uid}</code>
User Password	<code>\${jsam.attributes.userpassword}</code>
DN	<code>\${jsam.attributes.dn}</code>
CN	<code>\${jsam.attributes.cn}</code>
SN	<code>\${jsam.attributes.sn}</code>
Inet User Status	<code>\${jsam.attributes.inetuserstatus}</code>
Given Name	<code>\${jsam.attributes.givenname}</code>
Object Class	<code>\${jsam.attributes.objectclass}</code>

Policy Example

The following illustrates how this custom assertion might be used in a policy:

"At least one assertion must evaluate to true"

Require HTTP Basic Credentials

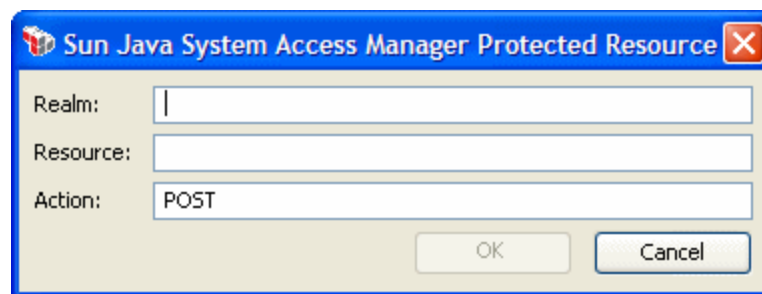
Require HTTP Cookie: iPlanetDirectoryPro

Access Resource Protected by JSAM

Route via HTTP(S) to URL

Using the Assertion

- Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
- Right-click **Access Resource Protected by JSAM** in the policy window and select **JSAM: Access Resource Protected by JSAM** or double-click the assertion in the policy window. The assertion properties are displayed.



The image shows a dialog box titled "Sun Java System Access Manager Protected Resource". It contains three input fields: "Realm:" with a cursor in the text box, "Resource:" with an empty text box, and "Action:" with the text "POST" entered. At the bottom right, there are "OK" and "Cancel" buttons.

Figure 251: Access Resource Protected by JSAM Properties

- Configure the dialog as follows:

Table 241: Access Resource Protected by JSAM settings

Setting	Description
Realm	Enter the name of the realm defined on the Java System Access Manager Server.
Resource	Enter the protected resource URL defined in a JSAM policy. Be sure to include the port number. For example: <i>http://server:80/WebApp/Service1.asmx</i>

Setting	Description
Action	Enter the allowed action, as defined in a JSAM policy. For example, POST or GET . The default is POST.

- Click **[OK]** when done.

Access Resource Protected by Oracle Access Manager Assertion

The *Access Resource Protected by Oracle Access Manager* assertion is used to delegate authentication and authorization to an Oracle Access Manager 10g or 11g server.

This assertion will also populate context variables with the values and attributes set for the action on Authorization Rules (on the OAM server).

You can control whether the obSSO cookie is excluded in the outbound request and set its contents in a custom context variable.

The Administrator is responsible for installing and configuring the Access Resource Protected by Oracle Access Manager Assertion on the Gateway. For more information, refer to the *Layer 7 Custom Assertion Installation Manual*.

Context Variables Created by This Assertion

The Access Resource Protected by Oracle Access Manager assertion sets the following context variables with details of the verification.

Table 242: Context variables created by the Access Resources Protected by OAM assertion

Variable	Description
<code>\${<prefix>.actions.type}</code>	A multivalued context variable that returns all action types.
<code>\${<prefix>.actions.<type>.names}</code>	A multivalued context variable that returns all names related to a certain type, specified by " <code><type></code> ".
<code>\${<prefix>.actions.<type>.<name>}</code>	Returns a values or attribute related to a certain type and name, specified by " <code><type></code> " and " <code><name></code> ".
<code>oam.ssoCookie</code>	Returns the session cookie. Note: This is only the default name. The actual name is specified in the assertion properties.

Variable	Description
oam.ssoCookie.name	Returns the name of the OAM cookie.
oam.ssoCookie.value	Returns full information of the session cookie, such as name, value, and attributes.

The default variable prefix is **oamResponse**. This can be changed in the assertion properties.


Using the Assertion




1. Do one of the following:
 - To add the assertion to the policy development window, drag and drop the assertion from the Access Control category in the *Assertion* tab into the policy window.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Access Resource Protected By Oracle Access Manager** in the policy window and select **Oracle Access Manager Protected Resource**. The properties dialog appears.

Figure 252: Oracle Access Manager Protected Resource Properties

3. Configure the properties as follows:

Table 243: Oracle Access Manager Protected Resource settings

Setting	Description
Protected Resource 	<p>Enter the full name of the resource being protected. This should be in the format:</p> <p style="text-align: center;"><i>//<host>:<port><resource_URL></i></p> <p>Where:</p> <ul style="list-style-type: none"> <i><host></i> is the hostname of the server that is servicing <i><resource_URL></i> <i><port></i> is the port number on the server (optional) <i><resource_URL></i> is the resource being serviced; this resource definition should follow the guidelines set by Oracle Access Manager <p>You can also specify a context variable that contains the resource.</p> <p>Note the following:</p>

Setting	Description
	<ul style="list-style-type: none"> The <code><host></code> and <code><port></code> must be defined in one of the host identifiers via the OAM Admin Console. The <code><resource_URL></code> must be defined in one of the Resources in the Application Domains via OAM Admin Console. The <code><host></code> does not need to be an actual hostname, as long as it is defined in a host identifier via the OAM Admin Console.
Type 	<p>Enter the type of the resource. This can be a built-in type, such as HTTP or EJB, or a custom type defined through the Access System Console. For custom resource types, custom operations are defined using the Oracle Access Manager system console when the resource type is defined.</p> <p>You can also specify a context variable that contains the type.</p> <p>Default: HTTP</p>
Method 	<p>Enter the action to be performed against the protected resource, as dictated by the resource type. Examples are GET and POST for HTTP resources, and EXECUTE for EJB resources. For custom resources, operations are defined through the Access System Console when the resource type is defined.</p> <p>You can also specify a context variable that contains the method.</p>
Response Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p> <p>Default prefix: oamResponse</p> <p>For an explanation of the validation messages displayed, see Context Variable Validation in the <i>Layer 7 Policy Manager User Manual</i>.</p>
Set SSO cookie in context variable: 	<p>Specify a context variable to save the SSO cookie and value, including all specified cookie attributes.</p> <p>Default: oam.ssoCookie</p> <p>You can access specific cookie information using these suffixes:</p> <ul style="list-style-type: none"> <code><variable>.name</code>: Returns the name of the OAM SSO cookie. <code><variable>.value</code>: Returns the value of the OAM SSO cookie.
Omit SSO cookie in Outbound Request	<p>Select this check box to exclude the ObSSOCookie from the outbound request.</p> <p>Tip: You can exclude the cookie to avoid certain problems. For example, this assertion is used to authenticate and authorize consumer access to a service, but the CA API Gateway needs to identify itself with a different set of credentials to an OAM-protected service endpoint.</p> <p>Clear this check box to not exclude the cookie in the outbound request.</p>

Setting	Description
	<p>Note: Clearing the check box does not guarantee that the outbound request will contain an SSO Cookie or an updated SSO cookie.</p> <p>Tip: To ensure that the outbound request contains an SSO Cookie, use either the Manage Transport Properties/Headers or Route via HTTP (S) assertions to add the cookie using the <i>oam.ssoCookie</i> context variable.</p>
Set cookie attributes	<p>Select this check box to set the cookie attributes.</p> <ul style="list-style-type: none"> • Domain: Enter the domain for which the cookie is valid (optional). • Path: Type the full path that specifies the subset of URLs to which this cookie applies (optional). • Expiry: Enter the lifetime of the cookie, in seconds (optional). A negative value indicates that the cookie is not stored persistently and will be deleted when the browser is closed. The default is -1, which the cookie will persist until the browser is closed. • Version: Enter the required version of the state management specifications to which the cookie conforms. The default is 0, which means the cookie is using the Netscape cookie format. Any other versions (for example, 1+) mean that the cookie is using the RFC 2109 cookie format. • Comment: Enter any comments (optional). Note: As cookies can contain private information about a user, the Cookie attribute allows an origin server to document its intended use of a cookie. You can then inspect the information to decide whether to initiate or continue a session with this cookie.
HTTP only	<p>Select this check box to direct browsers to use cookies via the HTTP or HTTP(S) protocols. This setting is the default.</p> <p>Clear this check box to allow browsers to use cookies via other protocols.</p>
Secure	<p>Select this check box to direct browsers to use cookies only via encrypted/secure connections.</p> <p>Clear this check box to allow browsers to use cookies in unsecured connections.</p>

Note: The values used will be determined by the configuration of the Oracle Access Manager System. For more information, refer to the *Oracle® Access Manager User Guide*.

4. Click **[OK]**.

When a successful authorization call is made to Oracle Access Manager using this assertion, the obSSOCookie is added to the response HTTP header (unless suppressed). If the cookie is available in the request HTTP header on subsequent calls to the CA API Gateway policy using the Access Resource Protected by Oracle Access Manager assertion, the cookie will be used as the authorization credentials for the user.


When a user is authenticated by the Access Resource Protected by Oracle Access Manager assertion, the authorization action information is made available in the policy using the context variable described under "[Context Variables Created by This Assertion](#)".

Note: Failed authorization action information still requires a user to be authenticated, but not authorized to access the specified resource.

Authenticate using Tivoli Access Manager Assertion

Installing and configuring the TAM (Tivoli Access Manager) Custom Assertion package in the Gateway installs and enables the *Authenticate using Tivoli Access Manager* assertion in the Policy Manager. This assertion instructs the Gateway to delegate the authentication and authorization tasks required to gain access to a protected service to the IBM® Tivoli® Access Manager (version 6.0) server.

The Administrator is responsible for installing and configuring the TAM Custom Assertion package on the Gateway. For more information, refer to the *Custom Assertion Installation Manual*. If you encounter authentication errors during the execution of a policy, refer to the [Troubleshoot Errors](#) section below.

Notes: (1) You may receive an HTTP Basic authentication warning when the Tivoli Access Manager assertion is used with these assertions: [Require XPath Credentials](#), [Require FTP Credentials](#), or [Require WS-Security UsernameToken Profile Credentials](#). You may ignore this policy validation warning. (2) When running this assertion in the browser client, a triangular warning icon () may appear next to the dialog box when the assertion properties is displayed. You may ignore this icon.

Usage Rules

Note the following rules when using the Authenticate using Tivoli Access Manager assertion:

- You cannot use this assertion with:
 - Authentication assertions that encrypt passwords, such as the [Require SSL or TLS Transport with Client Authentication](#) assertion (a clear text password is required)

- The [Sign Element](#) and [Encrypt Element](#) assertions
- The [Authenticate User or Group](#) assertion .
- You can use this assertion with:
 - The [Require HTTP Basic Credentials](#) assertion
 - Username Token (including the [Require Encrypted UsernameToken Profile Credentials](#) assertion)
 - The [Require XPath Credentials](#) assertion
 - The [Require SSL or TLS Transport](#) assertion
 - Any other assertion not listed in the exclusion list above.
- A policy can only contain a single Authenticate using Tivoli Access Manager assertion per authentication scheme. For complex policies that contain more than one authentication scheme, multiple instances of this assertion may be used.
- In a policy, the Authenticate using Tivoli Access Manager assertion must appear before the [routing](#) assertion and after the Require SSL or TLS Transport assertions.

Note: You can use XML encryption/signing if the [Require Encrypted UsernameToken Profile Credentials](#) assertion is also present in the policy.

Using the Assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Authenticate using Tivoli Access Manager** in the policy window and choose **Authenticate using Tivoli Access Manager** or double-click the assertion in the policy window. The properties are displayed.

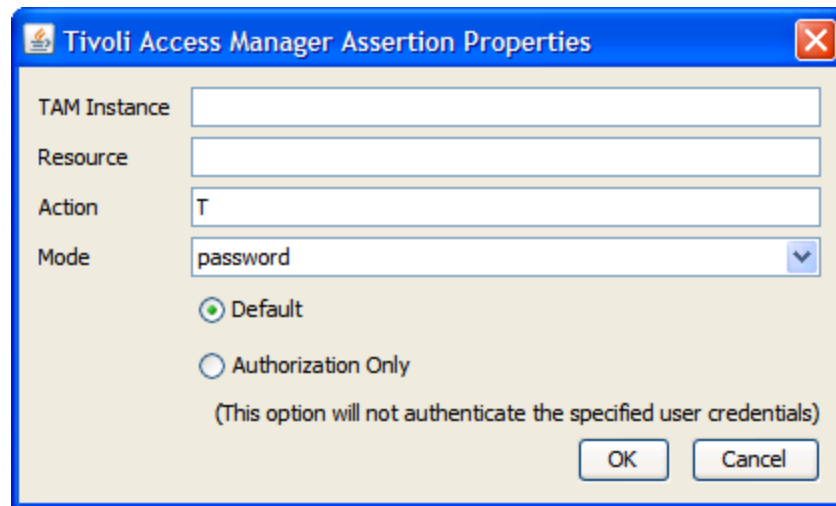



Figure 253: Tivoli Access Manager Authentication Properties

3. Configure the dialog as follows:

Table 244: Tivoli Access Manager assertion settings

Setting	Description
TAM Instance	<p>Specify the TAM instance to use:</p> <ul style="list-style-type: none"> Leave this field blank to use the default setting, which sets the TAM instance to the same value as <i>tam.pd.config.file.name</i> in the <i>tam_agent.properties</i> file on the Gateway. Enter the TAM instance name, as configured in the <i>tam_agent.properties</i> file on the Gateway. Specifically, this value is the "<instanceName>" part of the <i>tam.pd.config.file.name</i> property. <p>You can also reference a context variable containing the instance name.</p> <p>For more information on TAM instances, see <i>Installing the Tivoli Access Manager Assertion</i> in the <i>Custom Assertions Installation Manual</i>.</p>
Resource 	Enter the protected resource defined in the Tivoli Access Manager. You may reference context variables.
Action	Enter the requested action (such as "T" or "B") to be applied to resource for the given user.
Mode	Choose how user credentials are passed to the Tivoli Access Manager: password or iv-creds .

Note: The action and resource values are determined by the TAM (Tivoli Access Manager) settings used by the Gateway. The action value is taken from a list of allowable actions defined in the permission setting of the TAM Access Control List, and the resource value is the resource specified in the path in the configured TAM object space. Consult your TAM Administrator for information about the action and resource properties.

4. Click **[OK]** when done.

Troubleshooting

If configuration errors exist in the Tivoli Access Manager server or the CA API Gateway, the following error messages may appear in the Policy Manager Gateway Audit Events window when the Tivoli Access Manager assertion is used in a policy. For information about the Gateway Audit Events window, see Gateway Audit Events in the *Layer 7 Policy Manager User Manual*.

Contact your Administrator if you encounter authentication errors. If you require additional assistance, contact CA Technical Support.

Table 245: Tivoli Access Manager errors


Error Message	Description
SEVERE: Not init or failed	<p>This error message appears in the Gateway Audit Events window when:</p> <ul style="list-style-type: none"> • The TAM server is down • The TAM process is not running • The Gateway is not properly configured to connect to the TAM server. <p>Verify the Gateway and TAM server connection settings.</p>
WARNING: Authorization (access control) failed	<p>This error message appears in the Gateway Audit Events window when the Gateway connection credentials are not authenticated or authorized by the TAM server. A Log on to Gateway dialog prompts you to re-enter your user name and/or password. Ensure that the user name and password entered in SecureSpan XML VPN Client match those configured in the user database used by the TAM server to authenticate and authorize users.</p>

Authenticate with SiteMinder R12 Protected Resource Assertion

Installing and configuring the SiteMinder R12 Custom Assertion package in the Gateway installs and enables the *Authenticate with SiteMinder R12 Protected Resource* assertion in the Policy Manager. This assertion instructs the Gateway to delegate the authentication and authorization tasks required to gain access to a protected service on the CA

SiteMinder Policy Server version 12.0 running in FIPS-only mode.

The Administrator is responsible for installing and configuring the SiteMinder R12 Custom Assertion package on the Gateway. For more information, refer to the *Custom Assertion Installation Manual*. If you encounter authentication errors during the execution of a policy, refer to ["Troubleshooting"](#) below.

Notes: (1) You may receive an HTTP Basic authentication warning when the SiteMinder R12 Protected Resource assertion is used with these assertions: [Require XPath Credentials](#), [Require FTP Credentials](#), or [Require WS-Security UsernameToken Profile Credentials](#). You may ignore this policy validation warning. (2) When used in a policy that includes the [Require HTTP Basic Credentials](#) and [Require HTTP Cookie](#) assertions, ensure that the "HTTP Basic" assertion comes *after* the "HTTP Cookies" assertion. (3) When running this assertion in the browser client, a triangular warning icon () may appear next to the dialog box when the assertion properties is displayed. You may ignore this icon.

Context Variables Created by This Assertion

See "Authenticate with SiteMinder R12 Assertion" under Context Variables for SiteMinder in the *Layer 7 Policy Manager User Manual*.

Usage Rules

Note the following rules when using the SiteMinder R12 Protected Resource assertion:

- The Authenticate with SiteMinder R12 Protected Resource assertion cannot be used with:
 - Authentication assertions that encrypt passwords, such as the [Require SSL or TLS Transport with Client Authentication](#) assertion
 - The [Sign Element](#) and [Encrypt Element](#) assertions
 - The [Authenticate User or Group](#) assertion
- The Authenticate with SiteMinder R12 Protected Resource assertion can be used with:
 - The [Require HTTP Basic Credentials](#) assertion
 - [Require SSL or TLS Transport](#) assertion
 - Any other assertion not listed in the above list.
- A policy should contain only a single Authenticate with SiteMinder R12 Protected Resource assertion per authentication scheme. However, multiple occurrences of this assertion is possible in complex policies that contain multiple authentication schemes.

Note: You may receive a SiteMinder R12 Protected Resource warning when the assertion is used multiple times on one policy path ("Warning: You already have an access control Custom Assertion in this path.") You may ignore this policy validation warning.

- In a policy, the Authenticate with SiteMinder Protected Resource R12 assertion should appear before the [routing](#) assertion and after the "Require SSL or TLS Transport Assertion" on page 267 and authentication method assertions.

Using the assertion

1. Do one of the following:
 - To add the assertion to the Policy Development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to step 2 below.
2. Right-click **Authenticate with SiteMinder R12 Protected Resource** in the policy window and choose **Authenticate with SiteMinder R12 Protected Resource** or double-click the assertion in the policy window. The assertion properties are displayed.

Figure 254: SiteMinder R12 Custom Assertion Properties

3. Configure the dialog as follows:

Table 246: SiteMinder R12 Custom Assertion settings

Setting	Description
Agent ID	Enter the name of the CA SiteMinder Agent to use. The name may be omitted when only one agent is configured.
Protected Resource	Enter the name of the resource being protected by the CA SiteMinder Policy Server.

Setting	Description
Action	Enter the action (such as "POST" or "GET") for the protected resource. The default action is POST .
Authorize via SiteMinder Cookie	<p>Specify how authorization should occur:</p> <ul style="list-style-type: none"> Select this check box to have the assertion attempt to gather a valid SiteMinder cookie and place it in the HTTP Response. Clear this check box to not add a SiteMinder cookie to the HTTP Response. <p>If authorizing via SiteMinder Cookie, specify how to obtain the cookie:</p> <ul style="list-style-type: none"> Use cookie from request: Choose this option to have the assertion attempt to gather the SiteMinder cookie from the HTTP Request and add it to the HTTP Response with the name specified in the adjacent field. <p>Default SiteMinder cookie name: SMSESSION</p> <ul style="list-style-type: none"> Use cookie from variable: Choose this option to have the assertion attempt to gather a valid SiteMinder cookie from the context variable specified in the adjacent field (in the format "\${cookieName}"). <p>Note: The Gateway will log audit message code 8001 if a valid cookie could not be found.</p>

Note: The action and resource values are determined by the settings in the realm that is used by the Gateway custom agent in the CA SiteMinder Policy Server. Consult your Administrator for information about the action and resource properties.

- Click **[OK]** when done.

Troubleshooting

If configuration errors exist in the CA SiteMinder Policy Server or the Gateway, then one of the following error messages will appear in the Gateway Audit Events window when the SiteMinder R12 Protected Resource assertion is used in a policy.

Contact your Administrator if you encounter authentication errors.

Table 247: SiteMinder R12 errors

Error Message	Description
SEVERE: Unable to connect to the SiteMinder Policy Server	<p>This error message appears when:</p> <ul style="list-style-type: none"> The CA SiteMinder Policy Server is down The Gateway is not properly configured to connect to the CA SiteMinder Policy Server The connection credentials cannot be read properly because

Error Message	Description
	<p>the hashed cookie that is presented to the CA SiteMinder Policy Server cannot be decrypted.</p> <p>An error message indicating a SiteMinder Agent initialization failure is also displayed. Verify the CA API Gateway and CA SiteMinder Policy Server connection settings.</p>
SEVERE: The SiteMinder Agent name and/or the secret is incorrect	This error message appears when the agent name and/or the secret is not configured correctly.
WARNING: Authorization (access control) failed	<p>This error message appears when the Gateway connection credentials are not authenticated or authorized by the CA SiteMinder Policy Server. You will be prompted to re-enter your user name and/or password. Ensure that the user name and password entered in Securespan XML VPN Client match those configured in the user database used by the CA SiteMinder Policy Server to authenticate and authorize users.</p>
<p>The following error messages relate to port numbers defined in the <i>siteminder12.agent.configuration</i> cluster property. For detailed information about this cluster property, see "Installing the SiteMinder Assertion" in the <i>Custom Assertions Installation Manual</i>.</p>	
SEVERE: Siteminder configuration error: authentication port not defined	This error message appears when the authentication port is not defined properly.
SEVERE: Siteminder configuration error: authorization port not defined	This error message appears when the authorization port is not defined properly
SEVERE: Siteminder configuration error: accounting port not defined	This error message appears when the accounting port is not defined properly

Execute Salesforce Operation Assertion

The custom *Execute Salesforce Operation* assertion allows CA API Gateways to integrate with the SaaS data APIs provided by Salesforce.com.

The assertion supports the following Salesforce operations:

- Create Objects
- Update Objects
- Retrieve Objects

Retrieve Modified Objects
 Retrieve Deleted Objects
 Execute Query
 Search Objects

Note: This is an extra-cost assertion that requires separate licensing. For more information, please contact CA Technologies.

Context Variables Created by This Assertion

The Execute Salesforce Operation assertion sets the following context variables for each action. **Note:** The default `<prefix>` is "sfdc" and can be changed in the wizard.

Table 248: Context variables created by the Execute Salesforce Operation assertion

Action & Variable	Description
<i>Action: Create Objects</i>	
<code>\${<prefix>.sessionURL}</code>	Returns the session URL used for the call.
<code>\${<prefix>.sessionID}</code>	Returns the session ID used for the call.
<code>\${<prefix>.count}</code>	Returns the number of records processed.
<code>\${<prefix>. objectID}</code>	On success, returns the object IDs.
<code>\${<prefix>.error}</code>	Returns error messages, if any.
<code>\${<prefix>. warning}</code>	Returns warning messages for specific records, if any (for example, "Unknown picklist" or "Field truncated").
<i>Action: Update Objects</i>	
<code>\${<prefix>.sessionURL}</code>	Returns the session URL used for the call.
<code>\${<prefix>.sessionID}</code>	Returns the session ID used for the call.
<code>\${<prefix>.count}</code>	Returns the number of records processed.
<code>\${<prefix>. objectID}</code>	On success, returns the object IDs; multivalued.
<code>\${<prefix>.error}</code>	Returns error messages, if any; multivalued.
<code>\${<prefix>. warning}</code>	Returns warning messages, if any; multivalued.
<i>Action: Retrieve Objects</i>	
<code>\${<prefix>.</code>	Returns the session URL used for the call.

Action & Variable	Description
sessionURL	
\${<prefix>.sessionID}	Returns the session ID used for the call.
\${<prefix>.sObjectIDs}	On success, returns the object IDs; multivalued.
\${<prefix>.sObjects.<fieldName>}	On success, returns multivalued context variables; (optional).
<i>Action: Retrieve Modified Objects</i>	
\${<prefix>.sessionURL}	Returns the session URL used for the call.
\${<prefix>.sessionID}	Returns the session ID used for the call.
\${<prefix>.ObjectID}	On success, returns the object IDs; multivalued.
\${<prefix>.lastDateCovered}	On success, returns the last date covered.
\${<prefix>.sObjects.<fieldName>}	On success, returns multivalued context variables; (optional).
\${<prefix>.sObjects}	On success, returns XML representing the objects (optional); multivalued.
<i>Action: Retrieve Deleted Objects</i>	
\${<prefix>.sessionURL}	Returns the session URL used for the call.
\${<prefix>.sessionID}	Returns the session ID used for the call.
\${<prefix>.sObjects}	On success, returns list of objects.
\${<prefix>.lastDateCovered}	On success, returns the last date covered.
<i>Action: Execute Query</i>	
\${<prefix>.sessionURL}	Returns the session URL used for the call.
\${<prefix>.sessionID}	Returns the session ID used for the call.
\${<prefix>.queryLocator}	On success, returns the query locator of the query executed.

Action & Variable	Description
<code>\${<prefix>.count}</code>	Returns the number of records processed.
<code>\${<prefix>.done}</code>	Returns true or false based on the number of records processed.
<code>\${<prefix>.sObjects.<fieldName>}</code>	On success, returns multivalued context variables; (optional).
<code>\${<prefix>.sObjectXML}</code>	On success, returns XML representing the objects (optional); multivalued.
<i>Action: Search Objects</i>	
<code>\${<prefix>.sessionURL}</code>	Returns the session URL used for the call.
<code>\${<prefix>.sessionID}</code>	Returns the session ID used for the call.
<code>\${<prefix>.count}</code>	Returns the number of records processed.
<code>\${<prefix>.sObjects}</code>	On success, returns multivalued context variables.

Installing and configuring the Salesforce Custom Assertion package in the Gateway installs and enables the Execute Salesforce Operation assertion in the Policy Manager. The Administrator is responsible for installing and configuring the Salesforce Custom Assertion package on the Gateway. For more information, refer to the *Custom Assertion Installation Manual*.

Note: When running this assertion in the browser client, a triangular warning icon (⚠) may appear next to the dialog box when the assertion properties is displayed. You may ignore this icon.

Using the Assertion

- Do one of the following:
 - To add the assertion to the policy development window, see "Adding an Assertion" on page 112.
 - To change the configuration of an existing assertion, proceed to the next step.
- In the Policy Manager's policy window, right-click **Executive Salesforce Operation** and click **Execute Salesforce Operation Properties**. Alternatively, double-click the assertion in the policy window. The Execute Salesforce Operation Wizard is displayed.

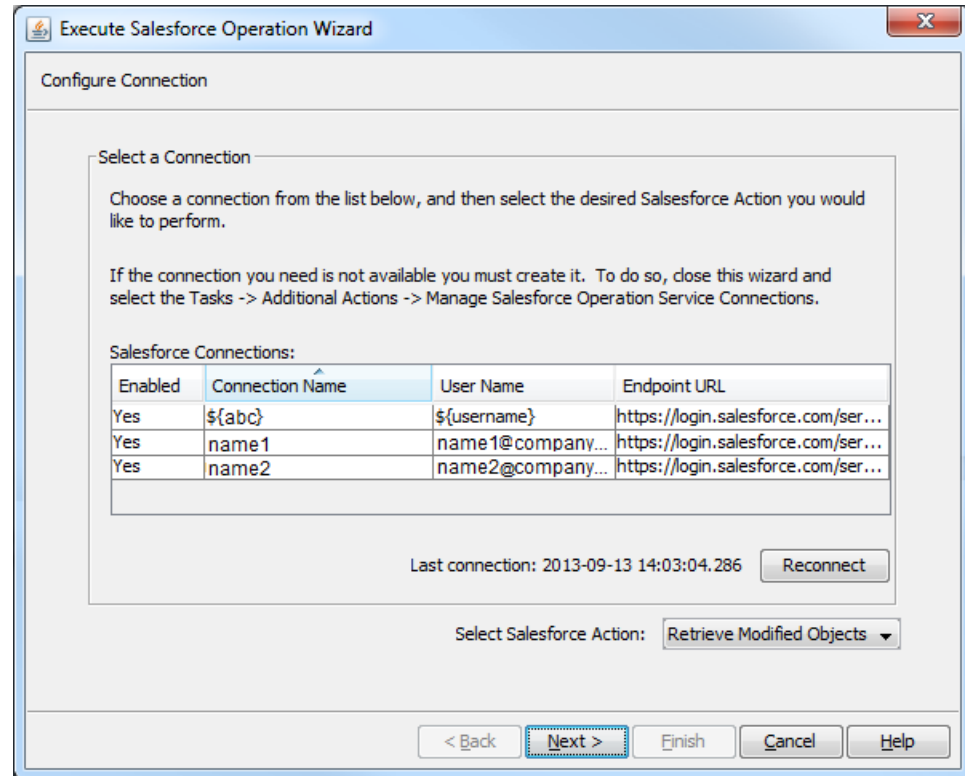
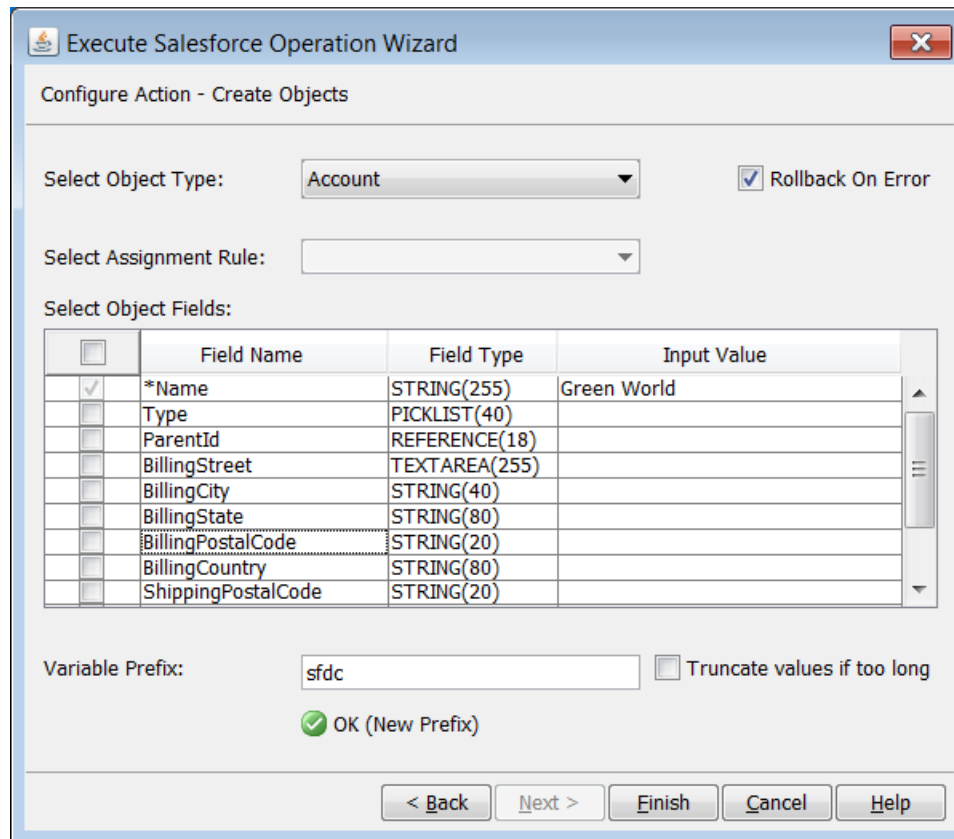


Figure 255: Execute Salesforce Operation Wizard: Configure Connection

3. In the list, choose a Salesforce connection to use and then click **[Reconnect]**. **Tip:** The Salesforce connections are defined using the Manage Salesforce Operation Service Connections task.
4. Select a Salesforce Action (operation) to perform and then click **[Next]**.
5. Configure the Salesforce action. See the appropriate section below for details.
6. Click **[Finish]** to close the wizard when done.

Creating Objects

To create objects in Salesforce, choose the Create Objects action in the Execute Salesforce Operation wizard (see Figure 255). Table 249 below describes the operation's settings.



Execute Salesforce Operation Wizard

Configure Action - Create Objects

Select Object Type: Account ☒ Rollback On Error

Select Assignment Rule:

Select Object Fields:

<input type="checkbox"/>	Field Name	Field Type	Input Value
<input checked="" type="checkbox"/>	*Name	STRING(255)	Green World
<input type="checkbox"/>	Type	PICKLIST(40)	
<input type="checkbox"/>	ParentId	REFERENCE(18)	
<input type="checkbox"/>	BillingStreet	TEXTAREA(255)	
<input type="checkbox"/>	BillingCity	STRING(40)	
<input type="checkbox"/>	BillingState	STRING(80)	
<input type="checkbox"/>	BillingPostalCode	STRING(20)	
<input type="checkbox"/>	BillingCountry	STRING(80)	
<input type="checkbox"/>	ShippingPostalCode	STRING(20)	

Variable Prefix: sfdc ☐ Truncate values if too long

☒ OK (New Prefix)


< Back Next > Finish Cancel Help

Figure 256: Execute Salesforce Operation Wizard: Configure Action - Create Objects

Configure the settings as follows:

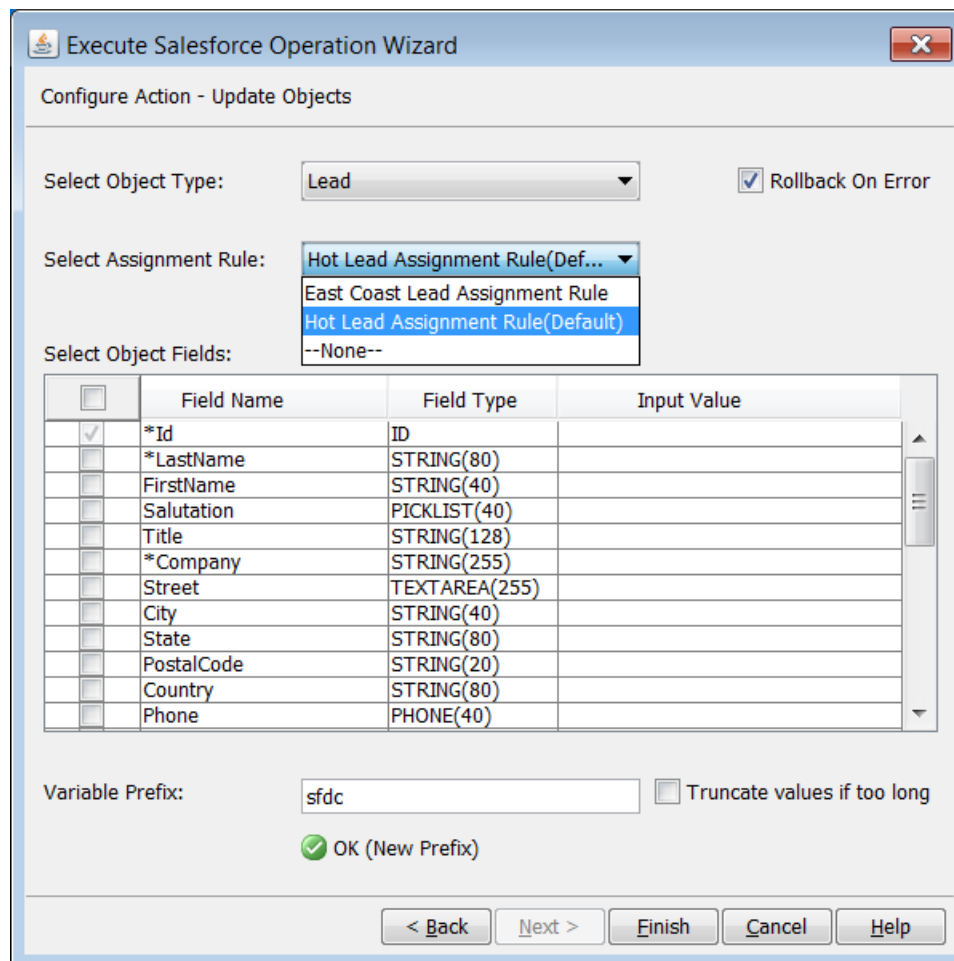
Table 249: Configure Action - Create Objects

Setting	Description
Select Object Type	Choose the object type to be created from the drop-down list. The object type determines which other fields require population.
Rollback On Error	<p>Select this check box so that, if the assertion fails to create an object, the assertion removes (rolls back) the objects it previously created during the call and does not try to create more. This setting is the default.</p> <p>Clear this check box so that, if the assertion fails to create an object, the assertion will continue to create objects and it will not remove any objects. You will have to look at the returned success/failure status for each object and create policy logic to handle the failures.</p> <p>IMPORTANT: Salesforce.com only supports batches up to 200 objects, and the assertion automatically issues multiple calls if required. This means that, if the assertion fails to create the 205th object, then the assertion only rolls back objects 201-204. It will not roll back objects 1-200 because they were part of the previous call.</p>

Setting	Description
Select Assignment Rule	For objects of type “Case” or “Lead”, choose the assignment rule to use from the drop-down list.
Select Object Fields 	<p>Select the fields to populate when the object is created. Then enter a value for each selected field. You may specify context variables. Mandatory fields are preselected and must be completed.</p> <p>Tip: You can create more than one object at a time by specifying multivalued context variables for each field. Ensure that all the multivalued variables have the same number of values, otherwise the assertion will fail.</p>
Variable Prefix	Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.
Truncate values if too long	<p>Select this check box to truncate input values if their length is greater than the maximum length for the given field.</p> <p>Clear this check box to leave long input values at their full length.</p> <p>Note: Records with truncated fields will still be successfully created. The truncation will be noted in the <code>\${<prefix>.warning}</code> context variable.</p>

Updating Objects

To update objects in Salesforce, choose the Update Objects action in the Execute Salesforce Operation wizard (see Figure 255). Table 250 below describes the operation's settings.



Execute Salesforce Operation Wizard

Configure Action - Update Objects

Select Object Type: Lead ☒ Rollback On Error

Select Assignment Rule: Hot Lead Assignment Rule(Default)

Select Object Fields:

<input type="checkbox"/>	Field Name	Field Type	Input Value
<input checked="" type="checkbox"/>	*Id	ID	
<input type="checkbox"/>	*LastName	STRING(80)	
<input type="checkbox"/>	FirstName	STRING(40)	
<input type="checkbox"/>	Salutation	PICKLIST(40)	
<input type="checkbox"/>	Title	STRING(128)	
<input type="checkbox"/>	*Company	STRING(255)	
<input type="checkbox"/>	Street	TEXTAREA(255)	
<input type="checkbox"/>	City	STRING(40)	
<input type="checkbox"/>	State	STRING(80)	
<input type="checkbox"/>	PostalCode	STRING(20)	
<input type="checkbox"/>	Country	STRING(80)	
<input type="checkbox"/>	Phone	PHONE(40)	

Variable Prefix: sfdc ☐ Truncate values if too long

☒ OK (New Prefix)


< Back Next > Finish Cancel Help

Figure 257: Execute Salesforce Operation Wizard: Configure Action - Update Objects

Configure the settings as follows:

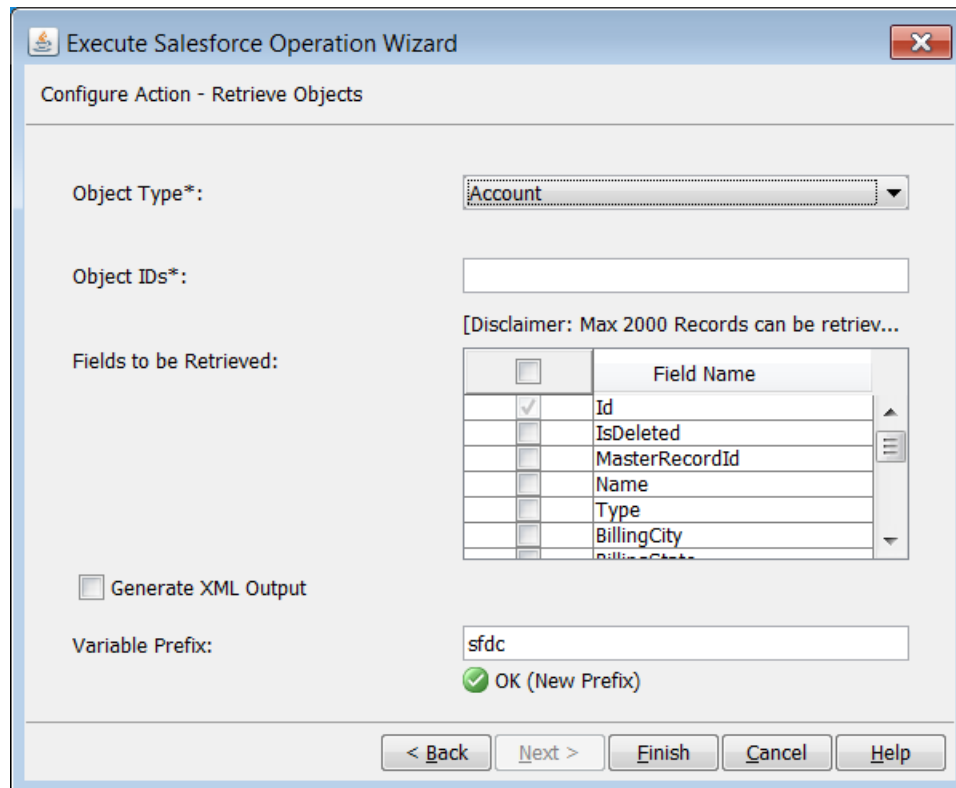
Table 250: Configure Action - Update Objects

Setting	Description
Select Object Type	Choose the object type to be updated. The object type determines which other fields require population.
Rollback On Error	<p>Select this check box so that, if the assertion fails to create an object, the assertion removes (rolls back) the objects it previously created during the call and does not try to create more. This setting is the default.</p> <p>Clear this check box so that, if the assertion fails to create an object, the assertion will continue to create objects and it will not remove any objects. You will have to look at the returned success/failure status for each object and create policy logic to handle the failures.</p> <p>IMPORTANT: Salesforce.com only supports batches up to 200 objects, and the assertion automatically issues multiple calls if required. This</p>

Setting	Description
	means that, if the assertion fails to create the 205th object, then the assertion only rolls back objects 201-204. It will not roll back objects 1-200 because they were part of the previous call.
Select Assignment Rule	This drop-down is enabled only when you are updating an object of type Case or Lead. Select the Assignment Rule from the drop-down list.
Select Object Fields 	Select the fields you want to populate when updating the object. The "Id" field is mandatory. Provide values for the object you are updating by entering text directly or leveraging context variables.
Variable Prefix	Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.
Truncate values if too long	<p>Select this check box to truncate input values if their length is greater than the maximum length for the given field.</p> <p>Clear this check box to leave long input values at their full length.</p> <p>Note: Records with truncated fields will still be successfully created. The truncation will be noted in the <code>\${<prefix>.warning}</code> context variable.</p>

Retrieving Objects

To retrieve objects in Salesforce, choose the Retrieve Objects action in the Execute Salesforce Operation wizard (see Figure 255). Table 251 below describes the operation's settings.



Execute Salesforce Operation Wizard

Configure Action - Retrieve Objects

Object Type*: Account

Object IDs*:


[Disclaimer: Max 2000 Records can be retriev...]

Fields to be Retrieved:

<input type="checkbox"/>	Field Name
<input checked="" type="checkbox"/>	Id
<input type="checkbox"/>	IsDeleted
<input type="checkbox"/>	MasterRecordId
<input type="checkbox"/>	Name
<input type="checkbox"/>	Type
<input type="checkbox"/>	BillingCity
<input type="checkbox"/>	BillingState

☐ Generate XML Output

Variable Prefix:


 OK (New Prefix)

< Back Next > Finish Cancel Help

Figure 258: Execute Salesforce Operation Wizard: Configure Action - Retrieve Objects

Configure the settings as follows:

Table 251: Configure Action - Retrieve Objects

Setting	Description
Select Object Type	Choose the object type to be retrieved. The object type determines which other fields require population.
Object ID(s) 	You can specify one or more Object IDs to retrieve. This field accepts context variables.
Select Fields to be Retrieved	Select the fields to be included in the retrieved object information. By default, the "Id" field will be selected.
Generate XML Output	Select this check box to return the object information in XML format. Clear this check box to return the object information in a multi-valued context variable.
Variable Prefix	Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.

Retrieving Modified Objects

To retrieve modified objects in Salesforce, choose the Retrieve Modified Objects action in the Execute Salesforce Operation wizard (see Figure 255). Table 252 below describes the operation's settings.

The screenshot shows the 'Execute Salesforce Operation Wizard' dialog box with the title 'Configure Action - Retrieve Modified Objects'. The settings are as follows:

- Select Object Type*:** A dropdown menu with 'Contact' selected.
- Timeframe*:** A dropdown menu with 'Past Day' selected. The list includes '---Select---', 'Past Hour', 'Past Day', 'Past Week', and 'Custom'.
- Retrieve Object Information:** A checkbox that is checked.
- Select Fields to be Retrieved:** A table with checkboxes and field names. The fields are: Id, IsDeleted, MasterRecordId, AccountId, LastName, FirstName, Salutation, and Name. The checkboxes for 'Id', 'AccountId', 'LastName', 'FirstName', and 'Salutation' are checked.
- Generate XML Output:** A checkbox that is unchecked.
- Variable Prefix:** A text box containing 'sfdc'.
- Buttons:** '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.


A disclaimer at the bottom right states: '[Disclaimer: Max 2000 Records can be retrieved]'.

Figure 259: Execute Salesforce Operation Wizard: Configure Action - Retrieve Modified Objects

Configure the settings as follows:

Table 252: Configure Action - Retrieve Modified Objects

Setting	Description
Select Object Type	Choose the object type to be retrieved. The object type determines which other fields require population.
Timeframe	Choose a time frame from the drop-down list: Past Hour , Past Day , Past Week , or Custom .

Setting	Description
	<p>The “Custom” option displays two additional text fields with which you can specify context variables for the custom period:</p> <ul style="list-style-type: none"> From UTC Time: From UTC Time is mandatory. It cannot be more than 15 days in the past. The required format of custom time frames is: “YYYY-MM-DD HH:MM:SS”. To UTC Time: If To UTC Time is left blank, it will default to the current time and always be greater than the From UTC Time. <p>Tip: The <code><prefix>.lastDateCovered</code> value from a previous assertion execution can be leveraged here to ensure full coverage.</p>
Retrieve Object Information	<p>Select this check box to retrieve the GUID of matching objects and the object information for each those objects. Selecting this check box also enables the Select Fields to Be Retrieved setting and the Generate XML Output setting.</p> <p>Clear this check box to retrieve only the GUID of matching objects.</p>
Select Fields to Be Retrieved	<p>Select the fields to be included in the retrieved object information. By default, the “Id” field will be selected.</p>
Generate XML Output	<p>Select this check box to return the object information in XML format.</p> <p>Clear this check box to return the object information in a multi-valued context variable.</p>
Variable Prefix	<p>Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.</p>

Retrieving Deleted Objects

To retrieve deleted objects in Salesforce, choose the Retrieve Deleted Objects action in the Execute Salesforce Operation wizard (see Figure 255). Table 253 below describes the operation's settings.

The assertion will return the GUIDs for any objects that meet the criteria specified. Field values cannot be retrieved for deleted objects.

Execute Salesforce Operation Wizard

Configure Action - Retrieve Deleted Objects

Select Object Type*: Account

Timeframe*: Custom

From UTC Time*: 2013-07-23 00:00:00
[Required Format: yyyy-MM-dd hh:MM:ss]

To UTC Time: 2013-07-24 23:59:59
[Required Format: yyyy-MM-dd hh:MM:ss]


Variable Prefix: sfdc
OK (New Prefix)

< Back Next > Finish Cancel Help

Figure 260: Execute Salesforce Operation Wizard: Configure Action - Retrieve Deleted Objects

Configure the settings as follows:

Table 253: Configure Action - Retrieve Deleted Objects

Setting	Description
Select Object Type	Choose the object type to be retrieved. The object type determines which other fields require population.
Enter Timeframe 	<p>Choose a timeframe from the drop-down list: Past Hour, Past Day, Past Week, or Custom.</p> <p>The "Custom" option displays two additional text fields with which you can specify context variables for the custom period:</p> <ul style="list-style-type: none"> From UTC Time: From UTC Time is mandatory. It cannot be more than 15 days in the past. The required format of custom time frames is: "YYYY-MM-DD HH:MM:SS". To UTC Time: If To UTC Time is left blank, it will default to the current time and always be greater than the From UTC Time. <p>Tip: The <code><prefix>.lastDateCovered</code> value from a previous assertion execution can be leveraged here to ensure full coverage.</p>
Variable Prefix	Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.

Executing Queries

To execute queries in Salesforce, choose the Execute Query action in the Execute Salesforce Operation wizard (see Figure 255). Table 254 below describes the operation's settings.

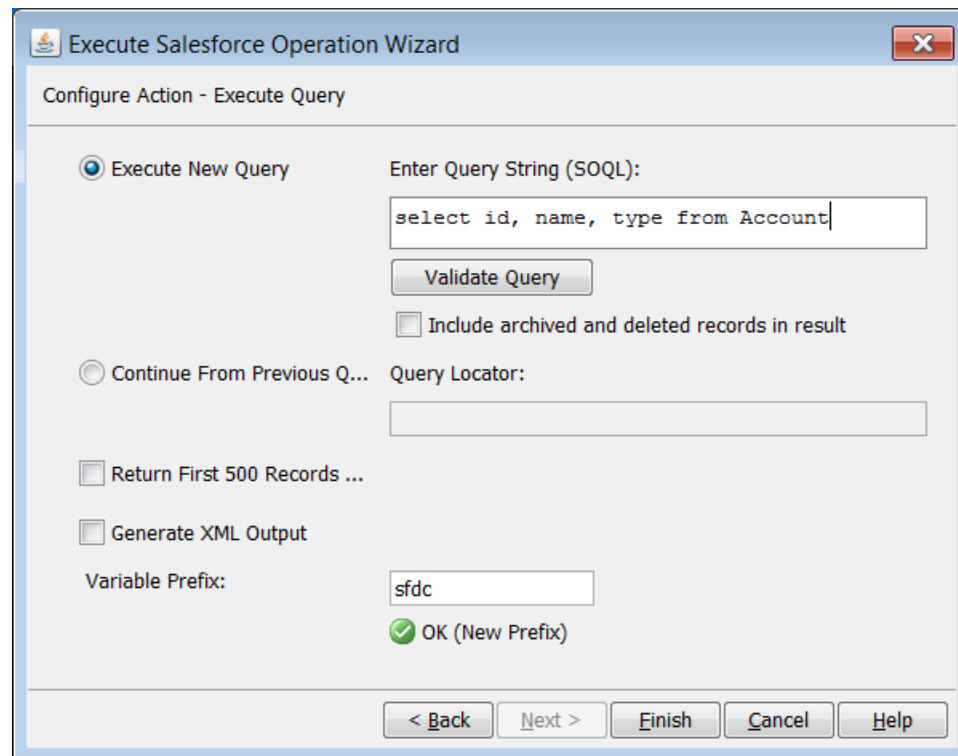




Figure 261: Execute Salesforce Operation Wizard: Configure Action - Execute Query

Configure the settings as follows:

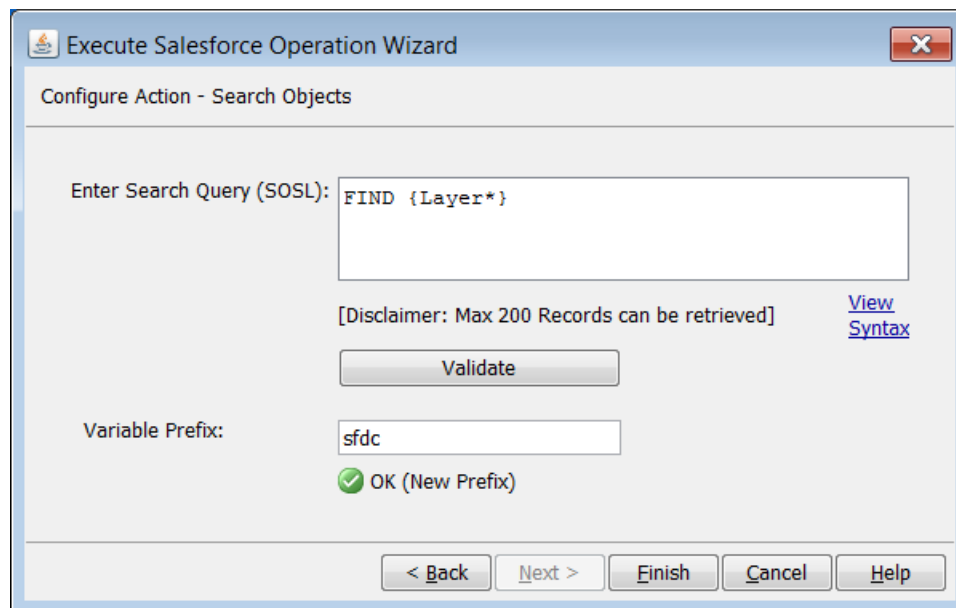
Table 254: Configure Action - Execute Query

Setting	Description
Execute New Query 	Select this option to begin a new query. Enter the SOQL query by typing or pasting text or specifying a context variable.
Validate Query button	Click this button to validate the query entered in the box above.
Include archived and deleted records in result	Select this check box to include archived and deleted records in the results. Clear this check box to exclude archived and deleted records from the results.

Setting	Description
Continue From Previous Query 	Select this option to begin the result set after the previously returned result set. Enter either a static query locator or a context variable query locator that is resolved at run time
Return first 500 records only	Select this check box to return only the first 500 records. Clear this check box to return all the records.
Generate XML Output	Select this check box to return the object information in XML format. Clear this check box to return the object information in a multi-valued context variable.
Variable Prefix	Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.

Searching Objects

To search objects in Salesforce, choose the Search Objects action in the Execute Salesforce Operation wizard (see Figure 255). Table 255 below describes the operation's settings.




Execute Salesforce Operation Wizard

Configure Action - Search Objects

Enter Search Query (SOSL):

[Disclaimer: Max 200 Records can be retrieved] [View Syntax](#)

Variable Prefix:


 OK (New Prefix)

< Back Next > Finish Cancel Help

Figure 262: Execute Salesforce Operation Wizard: Configure Action - Search Objects

Configure the settings as follows:

Table 255: Configure Action - Search Objects

Setting	Description
Enter Search Query (SOSL) 	Enter the SOSL query by typing or pasting text or specifying a context variable. Note: Only a maximum of 200 records can be retrieved with each SOSL query.
Validate button	Click this button to validate the SOSL query.
Variable Prefix	Enter a prefix that will be added to the context variables created by this assertion. This prefix will ensure uniqueness and will prevent the variables from overwriting each other when multiple instances of this assertion appear in a policy.

Exporting/Importing Policies

After importing a policy that uses the custom Execute Salesforce Operation assertion, open the assertion and do one of the following before using the policy:

- If the assertion has a selected connection, no action is needed.
- If the assertion has connections but none is selected, then select a connection and save the assertion.
- If the assertion does not have any connections, use the *Manage Salesforce Operation Service Connection* task to create a connection. Then open the assertion, select the new connection, and save the assertion. For information about creating connections, see *Managing Salesforce Operation Service Connections* in the *Layer 7 Policy Manager User Manual*.


Scan Using Symantec Antivirus Assertion

The Symantec® Custom Assertion package is an optional assertion package that can be added to the Gateway. When installed, this assertion allows the Gateway to direct all messages (and attachments) to the Symantec AntiVirus Engine, where they can be examined for viruses or other potentially malicious code in the message that could compromise the web service or other back end systems.

If the antivirus engine determines that the message is clean, the assertion passes and the Gateway continues to process the policy. If the antivirus engine detects a potential threat, then the assertion fails.

When this custom assertion package is installed, the *Scan Using Symantec Antivirus* assertion appears in the Policy Manager.

The Administrator is responsible for installing and configuring the Symantec Custom Assertion package on the Gateway. For more information, refer to the *Custom Assertion Installation Manual*.

Note: When running this assertion in the browser client, a triangular warning icon () may appear next to the dialog box when the assertion properties is displayed. You may ignore this icon.

Tip: Consider using the Scan Using ICAP-Enabled Antivirus assertion for greater flexibility in checking messages for viruses.

Using the Assertion

1. Add the assertion to the policy development window as described in "Adding an Assertion" on page 112.

The assertion is added to the policy window.

2. Position the assertion before the [routing](#) assertion to scan the request message; after the routing assertion to scan the response message.

Index

A

Access control	
assertions	161
Access Resource Protected by JSAM Assertion	720
Access Resource Protected by Oracle Access Manager Assertion	722
Add	
assertion to policy	112
fragments to policy	103
global schemas	75
listen port	133
UDDI registry	88
user or group to policy	170
Add "All" Assertion	619
Add "One or More" Assertion	619
Add Audit Detail Assertion	600
Add Comment to Policy Assertion	618
Add or Remove WS-Security Assertion	273
Add or Remove XML Element(s) Assertion	414
Add Security Token Assertion	277
Add Timestamp Assertion	283
Add WS-Addressing Assertion	416
Adding a Comment	40
Adding a New Global Resource	75
Adding a Policy Fragment to a Service Policy	103
Adding an Assertion	112
Administrative audits	598
Aliases	15
changing security zone	17
All Assertions Must Evaluate to True	2, 619
AMF policy	34
Annotating a policy	40
Antivirus	
Sophos	687, 692
Symantec	749
Apply JSON Transformation Assertion	419
Apply Rate Limit Assertion	573
Apply Throughput Quota Assertion	578
Apply to each member of a context variable	653
Apply XSL Transformation Assertion	424
Assertion	
adding	112
deleting	119
differences	30
disabling	119
enabling	121
evaluate to true	619
latency	122
properties	30
repositioning in policy	18
running concurrently	651
selecting target identity	152
selecting target message	153

Assertions Must Evaluate to True	619
Assertions tab	112
At Least One Assertion Must Evaluate to True	2, 619, 625
Audit Assertion	602, See also Message Auditing
Audit details	600
Audit message	
administrative	598
message summary	598
system	598
Audit Message Filter policy	34
Audit Messages in Policy Assertion	602
Audit records	713
converting to XML	713
Audit Viewer policy	36
Auditing	597, 602
Authenticate Against Identity Provider Assertion	163
Authenticate Against Radius Server Assertion	164
Authenticate Against SiteMinder Assertion	167
Authenticate Users or Group Assertion	170
Authenticate using Tivoli Access Manager	727
Authenticate with SiteMinder Protected Resource	730
Authenticating a Client via Kerberos	145
Authentication credentials	251
Authentication Credentials	217
Authentication in a Policy	14
Authentication method	231
Authorize via SiteMinder Assertion	173
Automatic Threat Protection	666
Automatic WS-Security	275
AV policy	36

B

Base64 encoding	435
Basic Security Profile 1.0	441, 443
BSP compliance	441
Build RST SOAP Request Assertion	285
Build RSTR SOAP Response Assertion	288
Build SAML Protocol Request Assertion	291
Build SAML Protocol Response Assertion	299
Builder, SAML	291

C

CA eTrust SiteMinder Policy Server	733
Cache	
JDBC metadata	206
look up in	585
storing to	594
Cancel Security Context Assertion	306
Capture Identity of Requestor Assertion	604
Certificates	
extracting attributes	180
Validation	486
Change	
content type	495, 694
WSS assertion recipient	146

Change WSS Assertion Recipient	146
Character encoding	412
Check Protected Resource Against SiteMinder	175
Check Results from XML Verification Assertion	372
Circular References	500, 704
Client	
authenticating via Kerberos	145
Client certificate	246
authenticating	238, 267
Clone	
listen port	133
UDDI registry	88
Code editor	159
Code injection protection	670
Coercive parsing attack	666
Collect WSDM Metrics Assertion	711
Comment	618
adding	40, 618
deleting	42
editing	41
Compare	
Expression	621
Expressions	621
Variable	621
Variables	621
Compare Expression Assertion	621
Comparing Policies	28
Compress Messages to/from XVC Assertion	429
Compression	429
Configure	
internal use policy	21
policies	18
policy fragment	21
Configure Authentication form	262
Configure SAML Browser/Artifact form	259
Configure WS-Security Decoration Assertion	309
Configure WS-Trust Credential Exchange form	177
Configuring a Policy	18
Configuring Message Buffering Assertion	508
Constrained Delegation	252
Content type	
validate or change	495, 694
Context Variables	
Comparing	621
data types	656
debug trace policy	71
joining	636
making available in fragments	632
retrieve credentials	250
splitting	661
Continue Processing Assertion	625
Convert Audit Record to XML Assertion	713
Cookies	215, 512
Copy Request Message to Response Assertion	510
Create	
internal use policy	21
policy fragment	21, 102
Create Include Fragment	102

Create routing strategy	
adding	626
cloning	626
deleting	628-629
editing	626
Create Routing Strategy	
cloning	628
editing	628
Create Routing Strategy Assertion	626
Create SAML Token Assertion	315
Create Security Context Token Assertion	328
Create XACML Request Assertion	330
Creating a Computer Account for NTLM Authentic-	
ation	221
Creating a Policy or Policy Fragment	9, 21
Credentials	
authenticating	14, 163
FTP	214
HTTP basic	215
SSH	237
WS-Federation	263
WS-Trust	177
XPath	248
Credentials from Context Variable	250
Cross-site request forgery	672
Custom Assertions	
SiteMinder R12 Protected Resource	730
Sun Java System Access Manager	720
Tivoli Access Manager	727
Customer identity	604
Customize Error Response Assertion	430
Customize SOAP Fault Response Assertion+	607
Customized SOAP faults	612

D

Debug trace policy	66-67
context variables	71
deleting	68
permissions	71
Debugger	56
Debugging a Policy	56
Decode	432, 435
Decrypt XML Element Assertion	374
Delete	
debug trace policy	68
global schemas	77
policy fragment	23, 105
policy template	142
Deleting a Comment	42
Deleting a Global Resource	77
Deleting a Policy Fragment	23, 105
Deleting an Assertion	119
Disable	
assertion	119
Disabling a Policy	24
Disabling an Assertion	119
Document structure threats	675, 678

Domain identity injection	226	Exporting a Policy	42
E		Exporting/Importing a Policy	42
Echo routing	510	External entity attack	667
Edit		Extract Attributes for Authenticated User Assertion ..	185
encapsulated assertion	133-134	Extract Attributes from Certificate Assertion	180
global schemas	77	Extract schema from WSDL	503, 707
policies	22	F	
policy fragment	104	Failover strategy	626
policy template	142	Fault level	607
Editing a Comment	41	Feedback List	630, 648
Editing a Global Resource	77	Folders	12
Editing a Policy	22	Forgery protection	672
Editing a Policy Fragment	104	Form data	488
Email alert	612	Fragments	21, 23, 104, 106, 635
Email Alert Properties form	613	Export variables	632
Enabling		FTP credentials	214
assertion	121	FTP(S) Routing	520, 563
Enabling a Policy	24	G	
Encapsulated assertion		Generate OAuth Signature Base String Assertion ..	360
adding	133	Generate Security Hash Assertion	365
cloning	133	Generate UUID Assertion	634
editing	133-134	GIF publishing	94, 99
exporting	133	Global policies	106
properties	134	Global resources	72
removing	133	adding	75
role	130	analyzing	84
Encapsulated Assertion Configuration Properties ..	134	deleting	77
Encode	412, 435, 437	editing	77
Encode to MTOM Format Assertion	437	importing	77
Encode/Decode Data Assertion	435	Global schemas	72
Encrypt Element Assertion	346	adding	75
Encrypt XML Element Assertion	376	deleting	77
Encrypted Username Token Assertion	213	editing	77
Enforce WS-I BSP Compliance Assertion	441	Groups	
Enforce WS-I SAML Compliance Assertion	443	authenticating	163, 170
Enforce WS-Security Policy Compliance Assertion ..	441	H	
Error messages	26	Handle UDDI Subscription Notification Assertion ..	714
Error response	430	Hash, generating	365
ESM		Header	
metrics	See WSDM metrics	adding	515
subscription	See WSDM subscription	host headers	535
Establish Outbound Secure Conversation Assertion	348	HTML Form Data Assertion	488
Evaluate JSON Path Expression	445	HTTP basic credentials	215
Evaluate Regular Expression Assertion	449	HTTP compression	429
Evaluate Request XPath Assertion	458	HTTP cookies	215
Evaluate Response XPath Assertion	461	HTTP form	482, 484
Evaluate SAML Protocol Response Assertion	353	HTTP header	
Evaluate WSDL Operation Assertion	465	adding	515
Evaluate XACML Policy Assertion	356	removing	515
Exchange Credentials using WS-Trust Assertion ..	177	replacing	515
Execute Routing Strategy Assertion	630	HTTP host headers	535
Execute Salesforce Operation Assertion	734	HTTP(S) Routing Properties	529
Explicit SSL	524		
Export			
policies	42		
Export Variables from Fragment Assertion	632		

I

Identity attributes	185
Implicit SSL	524
Import WS-Policy from URL in UDDI Registry	47
Importing a Policy from a File	44, 51
Importing a Policy via UDDI Registry	46
Include Policy Fragment Assertion	635
Index lookup	640
Inject domain identity	226
Injection protection	670
Internal	
assertion	711
Internal Services	
publish	91
WSDM QosMetric	711
WSDM Subscription	717
Internal use policies	33
Internal use policy	
creating	21
IP address range	592
ISO8859-1	413
Items by index position, looking up	640
Items by value, looking up	641

J

Java System Access Manager	720
JDBC connection	
editing	628
JDBC metadata, caching	206
JDBC query	187
JMS property	
adding	515
removing	515
replacing	515
JMS routing	541
Join Variable Assertion	636
JSON	
schema	490, 696
JSON document threats	678
JSON Path Expression	
evaluating	445
JSON transformation	419

K

KDC	144
Kerberos	143, 243
authenticating client	145
authentication credentials	251
Kerberos Configuration dialog	143
keyedReference meta data	99
Keytab	143

L

Latency, assertion	122
--------------------------	-----

LDAP query	209
Limit Availability to Time/Days Assertion	584
Limit Request Size Assertion	668
Limits	573, 668
Look Up Certificate Assertion	367
Look Up Context Variables	637
Look Up in Cache Assertion	585
Look Up Item by Index Position Assertion	640
Look Up Item by Value Assertion	641
Look Up Secure Conversation Assertion	370

M

Malicious code injection	684
Manage Cookie Assertion	512
Manage Gateway Assertion	715
Manage Gateway via REST	716
Manage Transport Headers/Properties Assertion	515
Management SOAP messages	715
Managing Global XML Schemas	72
Managing JMS Queues	544
Managing Kerberos Configuration	143
Managing Meta Data	99
Managing UDDI Registries	87
Manipulate Multivalued Variable	642
Target Multivalued Variable	643
Variable to append	643
Map	156
Map Value Assertion	644
Message Auditing	597
Message Buffering	508
Message Context	604
Message encoding	412
Message size	669
Message Summary audits	598
Messages	
audit	600
non-XML	37
template response	518
Meta data	99
Migrating Namespaces	158
MIME	482, 484
MIME multipart messages	433, 454, 494, 671
MQ Native	
routing	551
MS SQL server exploits protection	687
MTOM Message	
encoding	437
validating	493
Multiple identities in message	17
Multiple signatures	
enabling	17
Multivalued Context Variables	
create by joining	636
splitting	661

N

Namespace	
map	156
migrating	158
Netegrity SiteMinder Policy Server	733
New WSS Recipient Wizard	146, 151
Non-SOAP Check Results from XML Verification	372
Non-SOAP Decrypt XML Element Assertion	374
Non-SOAP Encrypt XML Element Assertion	376
Non-SOAP Sign XML Element Assertion	377
Non-SOAP Validate SAML Token	380
Non-SOAP Verify XML Element Assertion	391
Non-XML messages	37
Non SOAP Sign XML Element Assertion	377
Non SOAP Verify XML Element Assertion	391
Note	
adding to policy	40
NTLM	217, 241, 532
NTLM Authentication	217

O

OAM	722
OAuth	
Client	360
Server	360
OAuth Signature Base String	
generate	360
OData message	
validating	699
Oracle Access Manager Assertion	722
Oracle exploit protection	686
Organizing Services and Policies into Folders	12

P

Partial download, SFTP	563
Password	
digest credentials	244
Payloads	37
PDP	356
Perform JDBC Query Assertion	187
PHP eval injection	670
Policy	
alias	15
authenticating	14
comparing	28
configuring	18
creating	21
debugging	56
differences	28
disabling	24
editing	22, 32
enabling	24
error	26, 609
exporting	42
global	106

importing	44, 46
internal use	33
multiple	32
organizing	2
overview	1
revision	6
rolling back	6
validating	25
Policy Debug Trace	66
Policy Debugger	56
Policy Decision Point	356
Policy folders	12
Policy fragments	104
adding	103, 635
creating	21, 102
deleting	23, 105
Policy Organization	2
hints and tips	3
Policy Properties	9
Policy tag	10
Policy Templates	142
deleting	142
editing	142
exporting	42
importing	44, 46, 52
renaming	142
Policy Type	10
Preemptive compression	429
Prefix	156, 460
Private key	
selecting custom	530
Process Routing Strategy Result Assertion	648
Process RSTR Response Assertion	395
Process SAML Attribute Query Assertion	466
Process SAML Authentication Request Assertion	472
Protect Against Code Injection Assertion	670
Protect Against Cross-Site Request Forgery	672
Protect Against Document Structure Threats	675
Protect Against JSON Document Structure Threats	
Assertion	678
Protect Against SQL Attack Assertion	684
Protect Against WS-Security Replay Assertion	397, 680
Protected Resource	722
Protocol Transition	251
Publish to UDDI Settings	92

Q

QosMetrics	711
Query JDBC	187
Query LDAP Assertion	209
QueryThroughput Quota Assertion	587, 589
Queue	
JMS	541
Quota	578

R

Radius Server	164
Rate limit	573
Raw TCP	560
Recipient (WSS Assertion)	146
Regex	449
Regular Expression	412, 449
Reject	
based on nesting depth	675
based on SOAP request	675
based on XML length	675
Remove	
encapsulated assertion	133
UDDI registry	88
Renaming	
policy template	142
Replace Tag Content Assertion	475
Replay protection	397, 680
Request	
size limit	668
XACML	330
Request element	461, 668
Request forgery	672
Request Security Token (RST)	285, 288, 395
Requestor identity	604
Require Encrypted Element Assertion	400
Require FTP Credentials Assertion	214
Require HTTP Basic Credentials Assertion	215
Require HTTP Cookie Assertion	215
Require NTLM Authentication Credentials	217
Require Remote Domain Identity Assertion	226
Require Signed Element Assertion	402
Require SSH Credentials Assertion	237
Require SSL or TLS Transport Assertion	238, 267
Require Timestamp in Request Assertion	405
Require Windows Integrated Auth Credentials	241
Require WS-Addressing Assertion	477
Require WS-Secure Conversation Assertion	242
Require WS-Security Kerberos Token Profile Creds	243
Require WS-Security Password Digest Credentials Assertion	244
Require WS-Security Signature Credentials	246
Require WS-Security UsernameToken Profile Creds	248
Require XPath Credentials Assertion	248
Resolve External Dependencies Wizard	51
Resolve Service Assertion	590
Response element	465
REST Manage Gateway Assertion	716
Restrict	
day	584
IP addresses	592
time	584
Restrict Access to IP Address Range Assertion	592
Retrieve Credentials from Context Variable	250
Retrieve Kerberos Authentication Credentials Assertion	251
Retrieve SAML Browser Artifact Assertion	258

Return message to requestor	518
Return Template Response to Requestor Assertion	518
Revision	
policy	6
Route Strategy	
Execute	630
Route Variable Name	630
Route via FTP(S) Assertion	520
Advanced Tab	528
Authentication Tab	526
Connection Tab	523
Route via HTTP(S) Assertion	529
Authentication tab	531
Connection tab	536
Headers tab	533
HTTP tab	538
Other tab	539
Proxy tab	539
Route via JMS Assertion	541
Request tab	548
Response tab	550
Security tab	546
Target tab	544
Route via MQ Native Assertion	551
Route via Raw TCP Assertion	560
Route via SSH2 Assertion	563
Routing	510, 520, 529, 541, 563
Routing Strategy	
Create	626
RST SOAP request	285
RSTR response message	395
RSTR SOAP response	288
Run All Assertions Concurrently Assertion	651
Run Assertions for Each Item Assertion	653

S

Salesforce operation	734
SAML	
attribute query	466
authentication	472
compliance	443
endpoint	258
issuer	316
protocol	291, 299, 353
response status code	480
token	228, 315, 380
wizard	231
SAML authentication	472
SAML browser artifact	258
SAML Constraints Wizard	230-231
SAML Token Creation	317
SAML Token Creation Wizard	315, 317
SAML Token Profile Wizard	231
SAMLP	
builder	291, 299
evaluator	354
Scan Using Sophos Antivirus Assertion	687, 692

Scan using Symantec Antivirus Assertion	749
Schema	72
adding	75
circular references	500, 704
deleting	77
editing	77
poisoning	667
validating	499, 703
Searching	
UDDI Registry	48
Secure conversation	
cancelling	306
establishing	348
looking up	370
requiring	242
Security context	306
Security context token	328
Security hash	365
Security token	277, 328
Selecting	
target identity	152
target message	153
WSS recipient	147
XPath	154
Send Email Alert Assertion	612
Send SNMP Trap Assertion	615
Service	
alias	15
folders	12
Service Debugger	56
Service resolution	590
Services	
tab	112
Set Context Variable Assertion	656
Set SAML Response Status Code Assertion	480
SFTP partial downloads	563
Sign Element Assertion	407
Sign XML Element Assertion	377
Signature	246
multiple	17
SignatureConfirmation elements	409
Signed element	402
Signed Timestamp to Response	283
SiteMinder	
authenticate	167
authorize	173
check protected resource	175
SiteMinder policy server	733
SiteMinder R12 Protected Resource Assertion	730
SNMP Trap	615
SOAP attachment	497
SOAP fault	607, 611
SOAP management messages	715
SOAP request with attachment	497
Sophos Antivirus Assertion	687, 692
Split Variable Assertion	661
SQL attack protection	684
SQL code injection	684
SSH credentials	237
SSH Routing	563
SSL transport	238, 267
SSL with Client Certificate Authentication	238, 267
SSO Cookie	722
Stop Processing Assertion	664
Store to Cache Assertion	594
Structure threats	675
Stylesheets	425
Subfolders	12
Subscribe to WSDM Resource Assertion	717
Subscription notification	
UDDI	714
Sun Java System Access Manager Assertion	720
Symantec Virus Scanning Assertion	749
System audits	598
Systinet UDDI Registry	48
T	
Tabs	
policy	32
Tag content	475
TAM Assertion	727
Target identity	
selecting	152
Target message	153
TCP/IP attacks	666
Template response	518
Threat protection	666, 678
Throughput quota	578
Query	587, 589
Time/Day availability	584
Timestamp	283, 397, 405, 681
Tivoli Access Manager Assertion	727
TLS transport	238, 267
Trace policy	66-67
Transaction limits	573
Transformation	419, 424
Translate HTTP Form to MIME Assertion	482
Translate MIME to HTTP Form Assertion	484
Trusted Certificates	
looking up	367
U	
UDDI Notification Service	91
UDDI registry	46-48, 89, 92
adding	88
cloning	88
configuring	87
properties	89
removing	88
viewing	88
UDDI subscription notification	714
Understanding Assertion Latency	122
URL/URI encoding	435

Use WS-Federation Credential Assertion	263
Use WS-Security version 1.1 Assertion	409
UsernameToken	213, 248
Users	
authenticating	163, 170
context variable from	185
Using the XML Editor	159
UTF-8	412
UUID	634

V

Validate	
content type	495, 694
HTML form data	488
JSON schema	490, 696
MIME type	497
MTOM message	493, 699
policy	25
SAML	380
SOAP attachments	497
XML schemas	499, 704
Validate Certificate	486
Validate Certificate Assertion	486
Validate HTML Form Data Assertion	488
Validate JSON Schema Assertion	490, 696
Validate MTOM Message Assertion	493
Validate OData Messages Assertion	699
Validate or Change Content Type Assertion	495, 694
Validate SOAP Attachments Assertion	497
Validate XML Schema Assertion	499, 703
Value lookup	641
Value, mapping	644
Variables	
Comparing	621
Verify XML Element Assertion	391
Version	
policy	6

W

Warning messages	25
Web SSO	480
WebSphere MQ	551
Windows integrated authentication	241
Wizards	47, 52, 151, 231, 292, 315, 317
Working with Aliases	15
Working with Comments	40
Working with Global Policies	106
Working with Internal Use Policies	33
Working with Multiple Policy Tabs	32
Working with Multiple Signatures	17
Working with Non-XML Messages	37
Working with the Debug Trace Policy	67
Working with the Service Debugger	56
WS-Addressing	416, 477
WS-Federation passive credentials	263
WS-Secure Conversation	242

WS-Security	149, 309, 409
adding	273
automatic	275
clearing automatic	273
enforcing compliance	441
password digest credentials	244
removing	273
WS-Trust credential exchange	177
WSDL	
scanning	667
WSDL Operation	465
wsdm-notifications	34
WSDM metrics	711
WSDM subscription	717
WSI-BSP compliance	441
WSI-SAML compliance	443
WSS	
header handling	539
Kerberos	243
password digest credentials	244
recipient	146, 151
replay protection	397, 680
signature	246
WSS UsernameToken	248

X

X.509 BinarySecurityToken	246
X.509 certificate	
attributes	180
XACML PDP	356
XACML Request Builder	330
XDoS attack	499, 703
XML	
assertion	31
bomb attack	666
editor	159
namespace	156
parameter tampering	499, 703
routing detour	668
XML Element	
adding	414
decrypting	374
encrypting	376
removing	414
signing	377
verifying	391
XML verification	
check results	372
XPath	458, 461
acceleration	346
credentials	248
expressions	248
selecting	154
stylesheet	419, 424
XSL transformation	424