CA™ SOA Security Manager

Policy Configuration Guide

r12.1

Second Edition

CA Product References

This document references the following CA products:

- CA SOA Security Manager
- CA SiteMinder® Web Access Manager

Contact CA

Contact Technical Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is also available on the CA Support website, found at http://ca.com/docs.

Contents

Chapter 1: Introducing SOA Security Manager 19

SOA Security Manager Overview 19
SOA Security Manager Architecture and Components 19
Web Service Request Processing 25
Authentication Schemes 26
Authentication Service Models 27
How the Single-Step Authentication Model Works 28
How the Multistep Authentication Model Works 28
How the Chain Authentication Service Model Works 30
SiteMinder Session Ticket Support 32
How to Develop and Deploy SOA Security Manager Protected Web Services 33

Chapter 2: SOA Security Manager Configuration Overview 35

Policy Server Configuration and Management Interfaces 35
Policy Server Object Types 36
Infrastructure Objects 36
Policy Objects 38
Global Objects 39
Policy Management Methods 40

Chapter 3: Administrative User Interface Management 41

Administrative UI Overview 41
Start the Administrative UI 42
Manage Policy Server Objects 42
Duplicate Policy Server Objects 43
View Policy Server Object Properties 44
Modify an Existing Policy Server Object 44
Delete a Policy Server Object 45
SiteMinder Administrators 46
Default Administrator Account for the Administrative UI 47
Create an Administrator 47
Disable an Administrator Account 49
Administrator Use Case 49
Create a Legacy Administrator 53
Recreate a Deleted UI Administrator 56
How a SOA Agent and Policy Server Calculate Time 57

Chapter 4: Agents and Agent Groups 59

How to Configure Policy Server Objects for SOA Agents . . . 59
Trusted Hosts for SOA Agents . . . 59
Register a Trusted Host with the Policy Server . . . 59
Trusted Host Configuration Settings . . . 60
Delete Trusted Host Objects . . . 63
Host Configuration Objects for Trusted Hosts . . . 63
Copy a Host Configuration Object . . . 64
Create a Host Configuration Object . . . 64
Add Multiple Policy Servers to the Host Configuration Object . . . 65
Configure Policy Server Clusters for a Host Configuration Object . . . 66
SOA Agent Configuration Overview . . . 68
Advantages of Centrally Configuring SOA Agents . . . 68
SOA Agent Components . . . 69
Policy Server Objects Related to SOA Agents . . . 69
How to Configure a SOA Agent . . . 70
Configure SOA Agents Centrally . . . 71
Configure SOA Agents Locally . . . 71
Combined Central and Local Configuration . . . 72
Create an Agent Object to Establish a SOA Agent Identity . . . 73
Set the Configuration Parameters in the Agent Configuration Object . . . 73
Agent Configuration Object Overview . . . 74
Copy an Agent Configuration Object . . . 74
Create an Agent Configuration Object . . . 75
Required Agent Configuration Object Parameters . . . 76
Modify Agent Configuration Parameters . . . 78
Enable a SOA Agent . . . 79
Agent Groups . . . 79
Configure an Agent Group . . . 79
Add Agents to an Agent Group . . . 80
Custom Agents . . . 81
Configure a Custom Agent Type . . . 81
Create a Custom Agent Object for the Agent Identity . . . 82

Chapter 5: User Directories 85

User Directory Connections Overview . . . 85
LDAP Overview . . . 86
ODBC Database Overview . . . 96
Windows Directory Overview . . . 97
Active Directory Overview . . . 98
Custom Directory Overview . . . 99

Directory Attributes Overview 99
How to Configure a CA Directory User Directory Connection 101
Ping the User Store System 101
Configure CA Directory User Directory Connections 102
Enable User Store DSA Parameters 103
Enable Caching for a CA Directory User Store 103
Verify the CA Directory Cache Configuration 104
How to Configure a Sun Java System User Directory Connection 105
Ping the User Store System 105
Configure Sun Java System User Directory Connections 105
How to Configure a IBM Directory Server User Directory Connection 106
Ping the User Store System 106
Configure IBM Directory Server User Directory Connections 107
How to Configure a Domino User Directory as a User Store 108
Verify that a Domino User Directory Meets Policy Server Requirements 108
Ping the User Store System 108
Configure Domino Directory Connections 109
How to Configure a Novell eDirectory LDAP Directory Connection 110
Configure NetWare 110
Configure Anonymous LDAP Access on Novell eDirectory 111
Special Access for the SiteMinder Administrator 113
Create a Novell eDirectory User Account for SiteMinder Administration 113
Ping the User Store System 113
Configure Novell eDirectory LDAP Directory Connections 114
How to Configure an ADAM User Directory Connection 115
Create an ADAM Instance and User Store Connection 115
Create a SiteMinder User For Connecting to the ADAM User Store 116
Assign Administrator Privileges to the SiteMinder User 116
Load Users into the ADAM User Store 117
Configure ADAM User Store Directory Connections 117
How to Configure an Active Directory Directory Connection 118
Active Directory Considerations 119
LDAP Namespace for an Active Directory Connection 121
AD Namespace for an Active Directory Connection 123
Ping the User Store System 123
Configure Active Directory Connections 123
How to Configure an Active Directory Global Catalog User Directory Connection 125
Ping the User Store System 125
Configure Active Directory Global Catalog Directory Connections 125
Configure a Global Catalog User Store With an SSL Connection 127
How to Configure an Oracle Internet Directory User Directory Connection 127
LDAP Referral Limitation for Oracle Internet Directory User Directory 127

Ping the User Store System 127
Create an Organizational Unit for an OID Directory 128
Configure Oracle Internet Directory Connections 128
How to Configure an ODBC User Directory Connection 129
Ping the User Store System 129
Configure ODBC Directory Connections 130
SQL Server User Store Case Insensitivity and Extra Trailing Spaces Password Issues 131
SQL Server User Store Case Insensitivity and Extra Trailing Spaces Password Issues 134
How to Configure a Windows Directory Connection 138
WinNT Domain Connection Requirements 139
Ping the User Store System 140
Configure a Windows Directory Connection 140
How to Configure a Custom User Directory Connection 143
Ping the User Store System 143
Configure Custom Directory Connections 143
How to Configure an LDAP User Directory Connection over SSL 144
Before You Configure a Connection over SSL 145
Install the NSS Utility 146
Create the Certificate Database Files 146
Add the Root Certificate Authority to the Certificate Database 147
Add the Server Certificate to the Certificate Database 149
List the Certificates in the Certificate Database 150
Configure the User Directory Connection for SSL 151
Point the Policy Server to the Certificate Database 152
Verify the SSL Connection 152
LDAP Load Balancing and Failover 153
Port Number Considerations 154
Configure Failover 154
Configure Load Balancing 155
Configure Load Balancing and Failover 155
Use Case - Load Balancing and Failover 156
Configure ODBC Data Source Failover 157
SQL Query Schemes 158
Configure a SQL Query Scheme 158
Add SQL Query Schemes to ODBC User Directory Connections 160
How to Configure SQL Query Schemes for Authentication via Stored Procedures 160
Asynchronous Call Support During Failover and Connection Pooling 163
Define the Same User Directory Connection in Multiple Policy Stores 166
View User Directory Contents 167
Search User Directories 167
Universal IDs 168
How SiteMinder Uses UIDs 168

Named Expressions . . . 169
Benefits of Named Expressions . . . 169
Define Named Expressions . . . 170
User Attribute Mapping . . . 183
User Attribute Mapping Overview . . . 183
How Attribute Mapping Works . . . 185
Define an Attribute Mapping . . . 186
Apply User Attribute Mapping . . . 201

Chapter 6: Directory Mapping 207

Directory Mapping Overview . . . 207
Directory Mapping Requirements . . . 209
Supported Directory Mappings . . . 209
How to Configure an Authentication and Authorization Directory Mapping . . . 209
Configure a Directory Mapping . . . 210
Assign an Authorization Directory to a Realm . . . 210
Configure an AuthValidate Directory Mapping . . . 211
Directory Mapping Examples . . . 211
Employee Accesses an Engineering Realm Resource . . . 212
Temporary Employee Accesses a Quality Assurance Realm Resource . . . 212
Directory Mapping by Universal ID . . . 213
Directory Mapping Case Sensitivity . . . 213
Directory Mapping and Responses . . . 214

Chapter 7: Authentication Schemes 215

Authentication Scheme Overview . . . 215
Authentication Scheme Processing . . . 215
XML Document Credential Collector Authentication . . . 216
How Data Flows During XML DCC Authentication . . . 217
Configure the XML DCC Authentication Scheme . . . 217
XML Digital Signature Authentication . . . 230
How XML Digital Signature Authentication Works . . . 230
Required XML Document Elements for XML-DSIG Authentication . . . 231
Configure the XML DSIG Authentication Scheme . . . 232
SAML Session Ticket Authentication . . . 233
How SAML Session Ticket Authentication Works . . . 234
How Signing Assertions Affects SAML Session Ticket Authentication . . . 235
Benefits of the SAML Session Ticket Authentication Scheme . . . 235
Multistep Authentication Service Model Using SAML Session Tickets . . . 236
Multistep Authentication Using SAML Session Tickets Without Signed XML Documents . . . 239
Chain Authentication Service Model Using SAML Session Tickets . . . 239

Configure the SAML Session Ticket Authentication Scheme 242
WS-Security Authentication 243
How WS-Security Headers Are Produced 243
Supported Authentication Schemes for Producing Each WS-Security Header Type 244
How WS-Security Headers Are Consumed 245
How the Multistep Authentication Service Model Works Using WS-Security 245
How the Chain Authentication Service Model Works Using WS-Security 247
Choose a WS-Security Token Type 248
Additional WS-Security Features 260
Configure the WS-Security Authentication Scheme 264
Federation Use Cases 267
Certificate Mapping 269
Configure a Certificate Mapping 270
Test a Certificate Mapping 270
Confirm the Validity of Certificates 271
Custom Mapping Expressions 275
Custom Certificate Mapping for Multiple Attributes of the Same Type 278

Chapter 8: Additional WS-Security Configuration Requirements 281

XML Signing and Validation Service 281
Smkeydatabase Overview 281
What to Store in Smkeydatabase 283
Smkeydatabase Properties File 283
Use the Smkeytool Utility to Modify the Key Database 286
Configuration Requirements for Generating SAML Assertions 296
SAML 1.x Assertion Generator 297
Configure SAML 2.0 Service Providers 300

Chapter 9: Configure Security Policies from WSDL Files Using Enterprise Policy Management 303

Application Overview 303
Administrative Rights to Create Application Security Policies 304
How to Create Application Security Policies 305
Create an Application 306
(Optional) Configure Application Responses 307
Secure Web Service Resources from WSDL Files 308
Modify the Default Role to Define User Access Rights 310
Create Additional Roles to Define User Access Rights 311
Modify Role Assignments in the Application Policy 312

Chapter 10: Configure Security Policies Using Traditional Policy Management 315

Traditional Policy Management Overview . . . 315
How to Identify a Web Service Resource by Agent, Realm, and Rule . . . 316
How SOA Agent for Web Servers Identifies Web Service Resources . . . 316
How Other SOA Agent Types Identify Web Service Resources . . . 317
Resource Identification Policy Examples . . . 317
Unprotected Realms, Rules, and Policies . . . 319
Guided Example: Create Security Policies from a WSDL File . . . 320

Chapter 11: Domains 323

Policy Domain Overview . . . 323
Domains and User Membership . . . 324
How to Configure a Policy Domain . . . 324
Configure a Policy Domain . . . 325
Assign User Directories . . . 325
Create a Realm . . . 326
Disable Global Policy Processing for a Domain . . . 327
Modify a Domain . . . 327
Delete a Domain . . . 327

Chapter 12: Realms 329

Realms Overview . . . 329
Nested Realms . . . 330
Realms in Request Processing . . . 331
Configure a Realm . . . 331
Modify a Realm . . . 333
Delete a Realm . . . 333
Configure a Nested Realm . . . 333
Flush a Single Realm from the Resource Cache . . . 334

Chapter 13: Rules 337

Rules Overview . . . 337
How Rules Work as Part of a Policy . . . 338
How the Policy Server Processes Rules . . . 338
Rules and Nested Realms . . . 339
Rule Actions . . . 339
Advanced Rule Options . . . 341
Configure a Rule for Web Agent Actions . . . 341
Resource Matching and Regular Expressions . . . 342

Standard Resource Matching . . . 342
Regular Expressions for Resource Matching . . . 342
Enable and Disable Rules . . . 344
Advanced Rule Options . . . 344
Add Time Restrictions to a Rule . . . 344
Configure an Active Rule . . . 345
Delete a Rule . . . 346

Chapter 14: Rule Groups 347

Rule Group Overview . . . 347
Create a Rule Group . . . 347
Add Rules to a Rule Group . . . 348
Modify a Rule Group . . . 349
Modify a Rule Group . . . 349
Delete a Rule Group . . . 350

Chapter 15: Responses and Response Groups 351

Responses Overview . . . 351
Response Attribute Types . . . 352
Web Agent Response Attributes for SOA Agents . . . 353
Responses and Directory Mappings . . . 356
Configure a Response . . . 356
Configure a Response Attribute for SOA Agents . . . 357
Configure Responses to Generate SAML Session Tickets for Outgoing Messages . . . 357
How the SAML Session Ticket Response is Used . . . 358
Configure a SAML Session Ticket Response . . . 359
SAML Session Ticket Response Attribute Variables . . . 360
Use SAML Session Ticket Assertion Variables for a Session Ticket Response . . . 362
Configure Responses for WS-Security Header Production . . . 364
How the WS-Security Response is Used . . . 364
Configure a WS-Security Response . . . 365
WS-Security Response Examples . . . 374
Configure Response Attribute Caching . . . 377
Variable Objects in Responses . . . 377
Select a Variable Using Variable Lookup . . . 378
Configure a Response Attribute that Contains a Variable . . . 378
Edit a Response . . . 380
Delete a Response . . . 380
Response Groups . . . 380
Configure a Response Group . . . 380
Add Responses to a Response Group . . . 382

Add Responses to a Response Group . . . 382
Modify a Response Group . . . 383
Delete a Response Group . . . 383

Chapter 16: Policies 385

Policy Overview . . . 385
Policies Explanation . . . 387
Policy Bindings . . . 387
How to Configure a Policy . . . 388
Create the Policy . . . 388
Add Users to a Policy . . . 389
Add Rules to a Policy . . . 390
Associate a Rule with a Response or Response Group . . . 390
Associate a Rule with a Global Response . . . 391
Exclude a User or Group from a Policy . . . 392
Allow Nested Groups in Policies . . . 393
AND Users/Groups Check Box . . . 393
Specify AND/OR Relationships between Users/Groups . . . 395
Add Users by Manual Entry . . . 396
Enhance Policy Server's LDAP Authorization Performance . . . 397
Add an LDAP Expression to a Policy . . . 398
Enable and Disable Policies . . . 399
Advanced Policy Options . . . 400
Allowable IP Addresses for Policies . . . 400
Time Restrictions for Policies . . . 403
Configure an Active Policy . . . 404
Policy Binding Establishment . . . 404
Policy Bindings for LDAP Directories . . . 405
Policy Bindings for WinNT User Directories . . . 411
Policy Bindings for Microsoft SQL Server and Oracle User Directories . . . 413
Delete a Policy . . . 416
Bind Policies to SQL Queries . . . 416
Expressions in Policies . . . 417
Add an Expression to a Policy . . . 418

Chapter 17: Variables 419

eTelligent Rules . . . 419
SOA Security Manager eTelligent Rules Benefits . . . 419
eTelligent Rules Configuration . . . 420
Variables Overview . . . 423
Variable Types . . . 424

Variable Use in Policies . . . 425
Message-based Authorization Using Variables . . . 426
Variable Use in Responses . . . 427
Create a Variable . . . 427
Create a SAML Assertion Variable . . . 427
Create a Transport Variable . . . 428
Create an XML Agent Variable . . . 429
Create an XML Body Variable . . . 430
Create an XML Envelope Header Variable . . . 432
Create a Static Variable . . . 433
Create a Request Context Variable . . . 434
Create a User Context Variable . . . 435
Create a Form Post Variable . . . 436
Configure Message-based Authorization Using an XPath Query in XmlToolkit.properties . . . 436

Chapter 18: Global Policies, Rules, and Responses 439

Global Policies . . . 439
Global Policy Object Characteristics . . . 440
Global Policy Concept . . . 442
Global Policy Processing . . . 443
How to Configure Global Policies . . . 443
Global Rules . . . 443
Global Response Objects . . . 449
Response Attributes for Global Responses . . . 450
How to Configure Global Policy Objects . . . 451
Enable and Disable Global Policies . . . 453
Configure a Global Active Policy . . . 453
Allowable IP Addresses for Global Policies . . . 454
Specify a Single IP Address for a Global Policy . . . 455
Add a Host Name for a Global Policy . . . 455
Add a Subnet Mask for a Global Policy . . . 456
Add a Range of IP Addresses for a Global Policy . . . 456
Add and Remove Global Policy Time Restrictions . . . 457

Appendix A: LanMan User Directories 459

About LanMan User Directories . . . 459
LanMan Directory Connection Prerequisites . . . 459
Configure a LanMan Directory Connection . . . 460
Configure Registry Keys for a LanMan Directory Connection . . . 460
Configure a LanMan User Directory Connection . . . 461
Failover for Windows User Directories . . . 462

LanMan User Directory Search Criteria . . . 462

Appendix B: Attributes and Expressions Reference 463

Data Types . . . 463
Expression Syntax Overview . . . 466
Pasting . . . 467
Operators . . . 469
- Equality Operators . . . 470
- Inequality Operators . . . 470
- Less-than Operators . . . 471
- Greater-than Operators . . . 472
- Less-than or Equal-to Operators . . . 472
- Greater-than or Equal-to Operators . . . 473
- Begins-with Operators . . . 474
- Ends-with Operators . . . 474
- Containment Operators . . . 475
- Set Inclusion Operators . . . 475
- Pattern Matching Operator . . . 476
- Set Intersection Operators . . . 480
- Set Union Operators . . . 480
- NOT Operator . . . 481
- AND Operator . . . 481
- OR Operator . . . 482
- Exclusive OR Operator . . . 482
- String Concatenation Operator . . . 483
- Arithmetic Addition Operator . . . 483
- Arithmetic Subtraction Operator . . . 484
- Arithmetic Multiplication Operator . . . 484
- Arithmetic Division Operator . . . 485
- Conditional Operator . . . 485
- Indexing Operator . . . 486

Functions Available within Expressions . . . 486
- ABOVE Function--Is User Above Specified LDAP DN . . . 490
- ABS Function--Find the Absolute Value . . . 490
- AFTER Function--Find a String . . . 491
- ALL Function--All Bits Set . . . 492
- ANDBITS Function--Perform a Bitwise AND Operation . . . 493
- ANY Function--Any Bits Set . . . 494
- AT Function--Is User at Specified LDAP DN . . . 495
- BEFORE Function--Find a String . . . 495
- BELOW Function--Is User Below Specified LDAP DN . . . 497
- BOOLEAN Function--Convert to "TRUE" or "FALSE" . . . 497

CHAR Function--Convert an ASCII Value 498
CENTER Function--Pad a Source String 499
COMMONDN Function--Find a Common Root 501
COUNT Function--Count the Elements in a Set 502
DATE Function--Set to Midnight (form 1) 503
DATE Function--Convert Year, Month, Day, Hours, Minutes, and Seconds (form 2) 503
DATEFROMSTRING Function--Convert String to Number 504
DATETOSTRING Function--Convert Number to String 505
DAY Function--Return Day of Month 507
DOW Function--Return Day of Week 508
DOY Function--Return Day of Year 509
ENUMERATE Function--Test Set Elements 509
ERROR Function--Write Error Message to Console Log 511
EVALUATE Function--Evaluate an Expression 511
EXISTS Function--Look Up File Name 512
EXPLODEDN Function--Convert LDAP DN to Set 513
FILTER Function--Test Set Elements 514
FIND Function--Return Position in String 515
GET Function--Locate Attributes in a User Directory 517
HEX Function--Convert to Hexadecimal 518
HOUR Function--Convert to Hour 518
HOUR24 Function--Convert to Hour 519
INFO Function--Write INFO Message to Console Log 520
KEY Function--Look Up Key 520
LCASE Function--Convert to Lowercase 522
LEFT Function--Return Part of a String 522
LEN Function--Return the Length of a String 523
LOG Function--Write a String to a File 524
LOOP Function--Call a Virtual Attribute in a Loop 525
LPAD Function--Pad a Source String on the Left 526
LTRIM Function--Remove Leading Spaces in a String 527
MAX Function--Determine the Larger of Two Values 527
MAYBE Function--Report an Indeterminate Result 528
MID Function--Return Part of a String 530
MIN Function--Determine the Lesser of Two Numbers 531
MINUTE Function--Return the Minutes Component for a Date 532
MOD Function--Return Division Remainder 532
MONTH Function--Return the Month Component of a Date 533
NOTBITS Function--Perform a Bitwise NOT 534
NOW Function--Return Current Time in Seconds 535
NOWGMT Function--Return Current Time in Seconds 535
NUMBER Function--Convert to a Numeric Value 536

ORBITS Function--Perform a Bitwise OR Operation 537
PARENTDN Function--Retrieve Parent in LDAP Tree 538
PCASE Function--Convert a String to Proper Case 538
QS Function--Retrieve Items from a Query String 539
RDN Function--Retrieve First Component of LDAP DN 541
RELATIONDN Function--Compare Two Distinguished Names 542
RIGHT Function--Retrieve Characters from a String 543
RPAD Function--Pad a String on the Right 544
RPT Function--Repeat a String 545
RTRIM Function--Remove Trailing Spaces from a String 546
SECOND Function--Return the Number of Seconds in a Date 546
SET Function--Set the Value of an Attribute 547
SIGN Function--Return the Sign of a Number 548
SORT Function--Sort a Set 549
SPACE Function--Return a String of Spaces 550
STRING Function--Convert to a String 550
THROW Function--Stop Processing and Report Custom Error 551
TRACE Function--Write Trace Entry to Console Log 552
TRANSLATE Function--Replace String Value 552
UCASE Function--Convert to Upper Case 553
URL Function--Returns a Component of a URL String 554
URLDECODE Function--Decode a URL String 557
URLENCODE Function--Encode a String 557
VEXIST Function--Is the Parameter Defined? 558
TRACE Function--Write Trace Entry to Console Log 559
XORBITS Function--Perform a Bitwise XOR Operation 560
YEAR Function--Return the Year Component of a Numeric Date 560
YEAR4 Function--Return the Year Component of a Date (4 digits) 561

Index **563**

Chapter 1: Introducing SOA Security Manager

This section contains the following topics:

SOA Security Manager Overview (see page 19)
Authentication Service Models (see page 27)
How to Develop and Deploy SOA Security Manager Protected Web Services (see page 33)

SOA Security Manager Overview

SOA Security Manager is a policy-based access management system for Service Oriented Architecture (SOA) environments. With SOA Security Manager, you can protect XML transaction-processing web services that are implemented in the following ways:

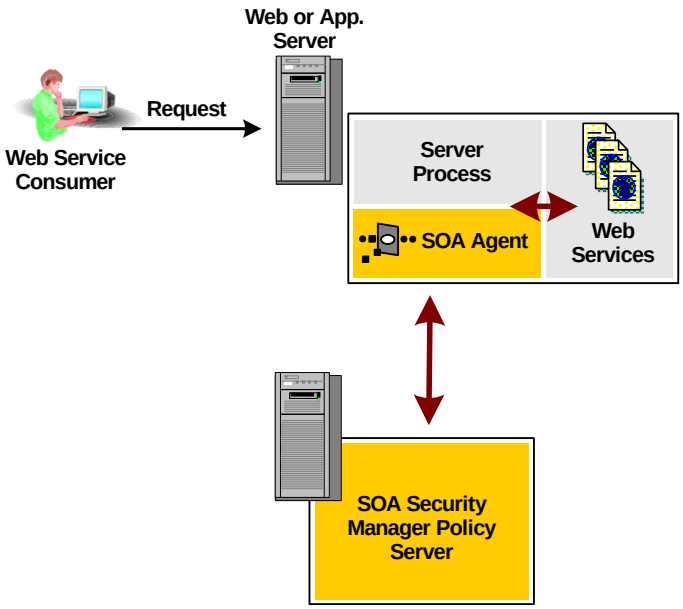
- Implemented as a servlet or active server page (ASP) and exposed by a web server or an application server.
- Implemented using the JAX-RPC binding and deployed on an IBM WebSphere Application Server or BEA WebLogic Server
-
- Exposed using the XML Firewall and web services proxy capabilities of the optional SOA Security Gateway

SOA Security Manager protects XML resources in much the same way as CA SiteMinder protects HTML resources, allowing entitlement data to be obtained from any layer of the XML message, depending upon the authentication and authorization needs of the back-end applications.

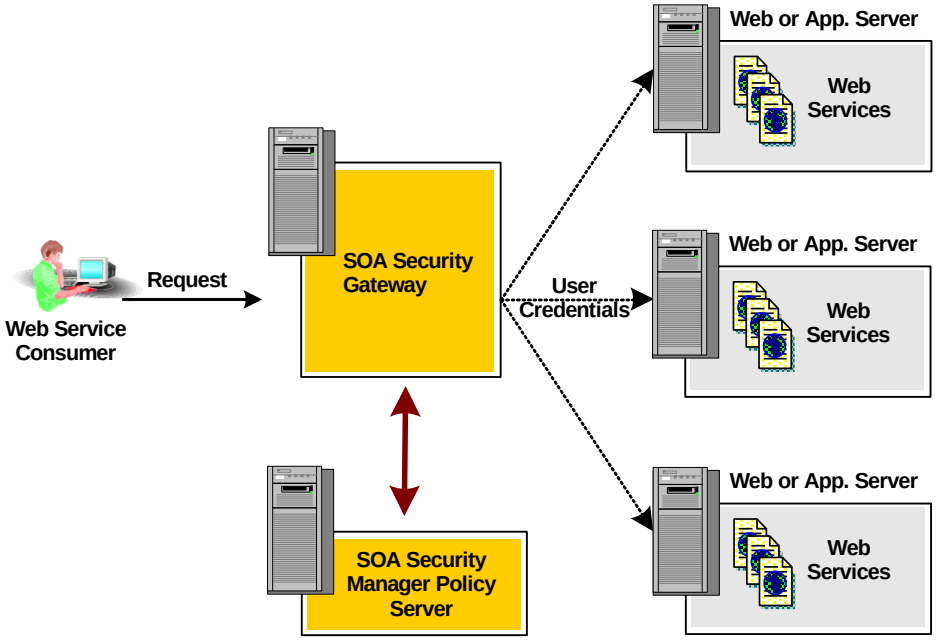
SOA Security Manager Architecture and Components

SOA Security Manager is based on the CA SiteMinder technology, using SOA Agents, the SOA Security Gateway, and an XML-enabled Policy Server to protect web service resources hosted on web and application servers.

The following illustration shows a simple SOA Security Manager environment in which a SOA Agent is deployed into a web or application server that is hosting web services.



The following illustration shows a simple SOA Security Manager environment in which a SOA Security Gateway is deployed in front of web or application servers hosting web services.



More complex architectures can also be configured to support multiple web service implementations or SOA Security Gateway implementations where SOA Agents are optionally deployed on web service endpoints to provide an additional layer of security.

SOA Security Manager Policy Server

The SOA Security Manager Policy Server (the "Policy Server") is an extended version of the CA SiteMinder Web Access Manager r12 SP1 Policy Server that provides a centralized, policy-based security management platform for SOA environments. As such, the Policy Server is the Policy Decision Point (PDP) in the SOA Security Manager environment.

The Policy Server integrates with SOA Agents as well as other CA access and identity management products and agent types to provide a single platform for securely managing every aspect of a company's business.

The Policy Server provides the following:

Authentication

The Policy Server supports a range of authentication methods.

Authorization

The Policy Server is responsible for managing and enforcing access control rules established by the Policy Server administrator. These rules define the operations that are allowed for each protected resource.

Administration

The Policy Server can be configured using the CA SOA Security Manager Web Access Manager Administrative UI. The Administration service of the Policy Server is what allows the Administrative UI to record configuration information in the Policy Store.

Accounting

The Policy Server generates log files that contain auditing information about the events that occur within the system. These logs can be printed in the form of predefined reports, so that security events or anomalies can be analyzed.

Health Monitoring

The Policy Server provides features for monitoring activity throughout a SOA Security Manager deployment.

In a SOA Security Manager implementation, a web service client sends a web service request in the form of an XML/SOAP message. At the target server, that request is intercepted by a SOA Agent. The SOA Agent determines whether or not the resource is protected, and if so, gathers the user's credentials from the request and passes them to the Policy Server.

The Policy Server authenticates the user against native user directories, then verifies if the authenticated user is authorized for the requested resource based on rules and policies contained in the Policy Store. Once a user is authenticated and authorized, the Policy Server grants access to protected resources and delivers privilege and entitlement information.

SOA Agents

SOA Agents are the Policy Enforcement Points (PEPs) in the SOA Security Manager environment, responsible for enforcing the policies defined on the Policy Server. Deployed at the end-points (web and application servers), they protect web services deployed in your SOA infrastructure.

SOA Agent for Web Servers

The SOA Agent for Web Servers is an XML-enabled version of the CA SiteMinder Web Agent. It can integrate with the following components:

- A web server to authenticate and authorize requests for access to web services bound to URLs served by that web server.
- A proxy server that handles requests for an application server to authenticate and authorize requests for access to web services hosted by the application server but associated with URLs served by the proxy server.

  Note: This approach is provided only to support upgraded TransactionMinder deployments that use it. For more advanced use cases requiring XML firewall and routing capabilities, CA recommends that you use the CA SOA Security Gateway, which works as a web services proxy and also provides advance routing and XML firewall capabilities

The SOA Agent recognizes requests that meet the following criteria as web service requests to be handled by SOA Security Manager:

- **Agent action**—POST; all XML message requests are posted. However, SOA Security Manager also provides two other agent actions, ProcessSOAP and ProcessXML, that allow you to create rules that fire for posted requests according to the XML message format.
- **Message MIME type**—text/xml by default; configurable using the XMLSDKMimeTypes Agent parameter.

All other requests are handled using the core Web Agent functionality of the SOA Agent, letting you also protect other resources on a web server, if you have purchased CA SiteMinder.

Note: For more information about protecting web resources using CA SiteMinder, see the *CA SiteMinder Agent Guide*.

SOA Agent for Application Servers

The SOA Agent for Applications Servers is a container-native agent for J2EE application servers that can be used to authenticate and authorize request messages sent over HTTP(S) or JMS transports to JAX-RPC resources hosted on the following application server platforms:

- IBM WebSphere Application Server
- BEA WebLogic Server

The SOA Agent recognizes requests that meet the following criteria as web service requests to be handled by SOA Security Manager:

- **Agent action**—POST; all XML message requests are posted. However, SOA Security Manager also provides two other agent actions, ProcessSOAP and ProcessXML, that allow you to create rules that fire for posted requests according to the XML message format.
- **Message MIME type**—text/xml by default; configurable using the XMLSDKMimeTypes Agent parameter.

SOA Security Gateway

The CA SOA Security Gateway is an XML gateway that manages XML traffic, protecting XML applications from malicious attack and from unauthorized access. Its XML Security Acceleration engine offloads security processing from web services applications and helps ensure optimum performance and throughput.

Note: The SOA Security Gateway requires a separate license in addition to the license for SOA Security Manager. To obtain the CA SOA Security Gateway license, contact the CA Licensing Group.

A SOA Security Gateway deployment includes the following components:

- SOA Security Gateway
- SOA Security Gateway Management Console

Policies can be stored in an XML file, a directory server, a relational database, or an XML database. The SOA Security Gateway Configuration Manager acts as the sole interface to the underlying policy storage system.

The SOA Security Gateway also provides SOA Agent functionality, allowing it to authenticate and authorize XML requests against the SOA Security Manager Policy Server.

Note: When planning and implementing SOA Security Manager security policies, consider the SOA Security Gateway as being the functional equivalent of a SOA Agent. Unless otherwise noted, diagrams and descriptions in this guide that refer to a SOA Agent also apply to the SOA Security Gateway.

Policy Server Configuration and Management Tools

The majority of Policy Server configuration tasks are performed using the Administrative UI, an interface that is generated dynamically based on the administrative privileges of the user.

However, some Policy Server management tasks that you perform using the Policy Server Management Console.

The management tasks controlled by the Policy Server Management Console include the following:

- Starting and stopping Policy Server processes
- Configuring Policy Server Executives
- Cache Management
- Key Management
- Global Settings
- User Management

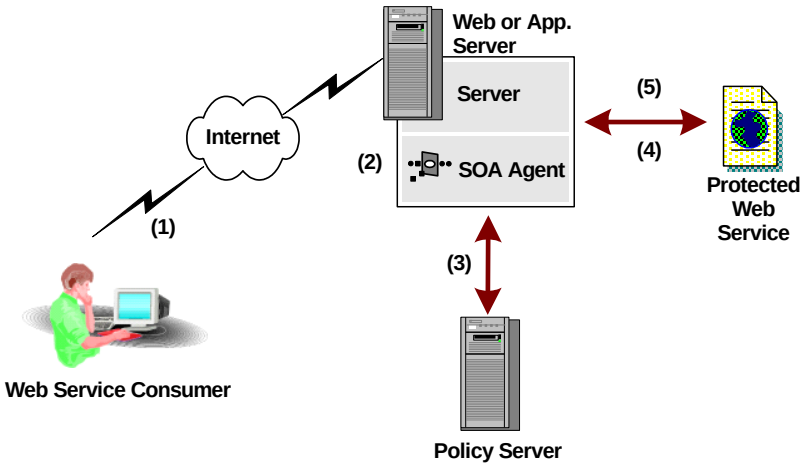
Note: More information on the Policy Server Management Console exists in the *Policy Server Administration Guide.*

More information:

Start the Administrative UI (see page 42)

Web Service Request Processing

SOA Security Manager supports content-level, XML-based security. The following illustration illustrates the flow of data in a simple, single web service implementation secured with SOA Security Manager.



The data in the previous illustration flows as follows:

1. A web service consumer (client) application creates a web service request in the form of an XML document and sends it to the web service provider site. An example document could be a purchase order. Credentials and authorization entitlements can be inserted in the message envelope or message body.
2. At the web service provider's site, the SOA Agent intercepts the request, based on its action and content type in the HTTP header, as shown in the following XML sample:

```
POST /CreditRating HTTP/1.1
Content-Type: text/xml
Content-Length: nnnn
SOAPAction:"someURI:CreditRating#GetCreditRating"

<SOAP-ENV:Envelope>
      <!-- request -->
</SOAP-ENV:Envelope>
```

3. The SOA Agent gathers the sender's credentials from the XML message and passes this information to the CA Policy Server for authentication and authorization.
4. The authorized message is passed to the back-end business application for processing.
5. Optionally, the back-end application returns a response to the web service requester with the status of the payload (for example, indicating that the purchase order has been accepted and is being processed).

Authentication Schemes

Authentication schemes that require user intervention are generally not appropriate for securing web services. SOA Security Manager provides four transport-level and message-level authentication schemes that *do not* require user intervention.

XML Document Credential Collector

Validates XML messages using credentials gathered from the message itself by mapping fields within the document to fields within a user directory.

XML Digital Signature

Validates XML documents digitally signed with valid X.509 certificates.

WS-Security

Validates XML messages using credentials gathered from WS-Security headers in a message's SOAP envelope.

SOA Security Manager can produce and consume WS-Security tokens. This enables you to use the WS-Security authentication scheme to deploy a multiple-web service implementation across federated sites.

SAML Session Ticket

Validates XML messages using credentials obtained from SOA Security Manager synchronized-sessioning SAML assertions (which contain an encrypted combination of a CA SiteMinder session ticket and a CA SiteMinder user's public key) placed in a message's HTTP header, SOAP envelope, or a cookie.

SOA Security Manager can generate and consume SAML Session Ticket assertions. This enables you to use the SAML Session Ticket authentication scheme to deploy a multiple-web service implementation within a single Policy Server domain.

Deciding which authentication scheme or schemes you intend to use to secure your web services is integral to how you design and implement your web services and is best made as part of the broader context of choosing an authentication service model.

More information:

Authentication Service Models (see page 27)
XML Document Credential Collector Authentication (see page 216)
XML Digital Signature Authentication (see page 230)
SAML Session Ticket Authentication (see page 233)
WS-Security Authentication (see page 243)

Authentication Service Models

The ability of SOA Security Manager to obtain security information from XML documents without user interaction and produce WS-Security headers, SAML Session Ticket assertions, and SiteMinder session cookies lets you securely deploy web services using a number of service models.

Single-step Authentication Service Model

All requests are authenticated and handled by a single web service.

Multistep Authentication Service Model

All requests are sent to a web service responsible for authentication, which then returns the message and authentication data back to the web service consumer. The web service consumer application can then send requests containing this authentication data to other related web services within or across domains.

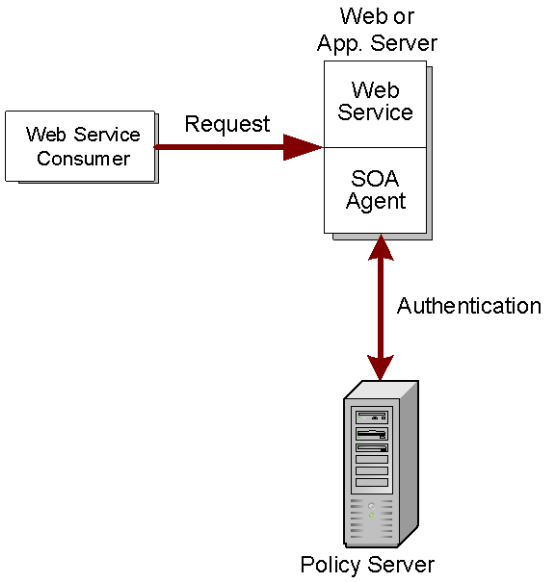
Chain Authentication Service Model

All requests are received by a web service responsible for authentication and then passed, with authentication data, to one or more other web services for handling. That is, message and authentication data always flows from the authentication web service directly to the next required web service, and from there to the next web service and so on, without further interaction from the web service consumer.

Choosing the appropriate authentication service model is the first, and probably most significant, decision you must make when designing a web service implementation. Your choice of service model also plays a significant role in determining the most appropriate SOA Security Manager authentication schemes to use.

How the Single-Step Authentication Model Works

The single-step service model is the simplest possible model for web services—requests from a web service consumer are authenticated and handled by a single web service. The following diagram shows the process by which web services consumers are authenticated using this simple model:



Appropriate authentication schemes for use in the single-step authentication model are as follows:

- XML Document Collector Authentication Scheme
- XML Digital Signature Authentication Scheme

How the Multistep Authentication Model Works

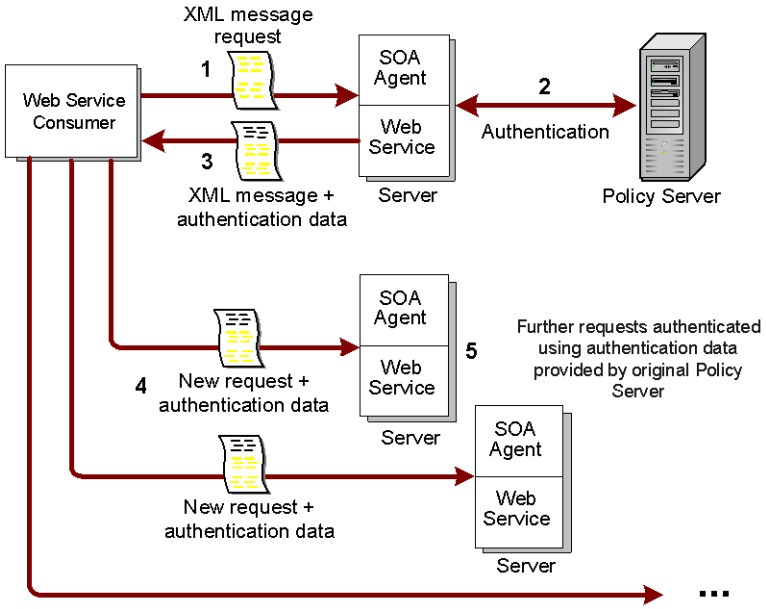
The multistep authentication model is like the CA SiteMinder cookie-based single sign-on implementation, in which WS-Security headers or SAML Session Ticket assertions take the place of the cookie.

In the multistep authentication model, a single web service is responsible for authenticating all incoming web service requests. This authentication service verifies a web service consumer's identity and returns an XML message with authentication data in the form of WS-Security headers or a SAML Session Ticket assertion. The web service consumer can then use this to add to subsequent requests to facilitate authentication by other associated web services.

The process that the web service consumer goes through when making a request has two phases:

1. Obtaining the authentication data
2. Using the authentication data to access other web services

The following illustration shows how request are processed in the multistep authentication service model:



1. The web service consumer sends a request for access to a protected web Service in the form of an XML document.
2. The SOA Agent receives the request, extracts credentials and passes them to the Policy Server, which authenticates the web service request with an appropriate authentication scheme.

   After authentication, the request goes through the authorization process. A response attribute associated with the authorizing policy causes the Policy Server to generate a response which it sends to the SOA Agent, instructing it to return authentication data to the web service.
3. The web service returns the authentication data back to the web service consumer (typically in an XML document, but synchronized sessioning SAML assertions can also be returned in HTTP headers or a cookie).
4. For subsequent requests, the web service consumer passes XML messages that include the authentication data it received from the authentication service to other associated web services.
5. The requests are allowed access without having to reauthenticate because the authentication data is supplied with the request message (in effect, providing single sign-on).

Appropriate authentication schemes for initial authentication by the authentication web service in the multistep authentication model are as follows:

- XML Document Collector Authentication Scheme
- XML Digital Signature Authentication Scheme

The authorizing policy for the authentication web service should trigger one of the following response types:

- WS-Security Responses (appropriate for web services protected by more than one policy store or at multiple sites)
- SAML Session Ticket Responses (appropriate for web services protected by the same policy store)

These responses instruct the SOA Agent to pass authentication data in the form of WS-Security headers or SAML Session Ticket assertions (as appropriate) back to the web service consumer for use in requests to associated web services. The associated web services should be protected using the corresponding authentication scheme:

- WS-Security Authentication Scheme
- SAML Session Ticket Authentication Scheme

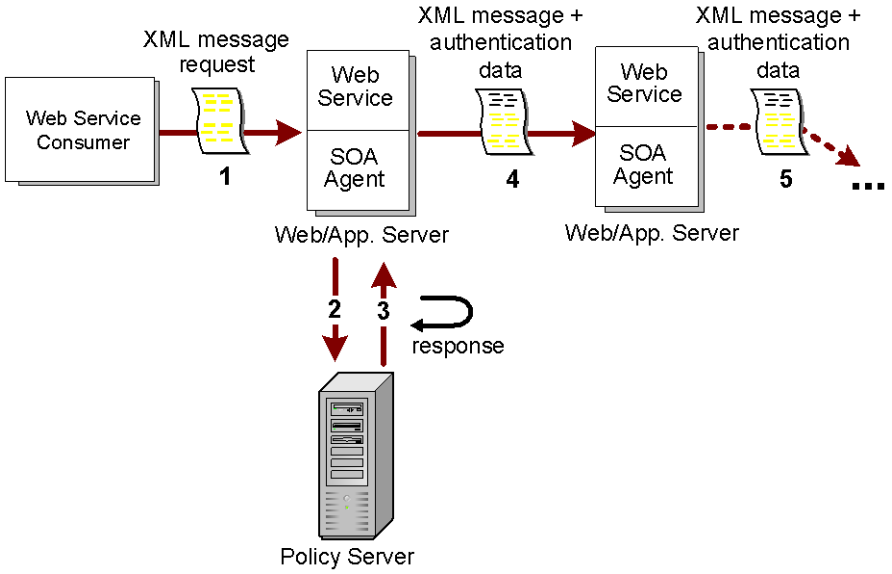
Note: SOA Agents can be configured to accept information from a CA SiteMinder session (SMSESSION) cookie in the HTTP header of a request as a means of authenticating a client and always add such cookies to request headers upon successful authentication and authorization. CA SiteMinder session cookies can therefore be used to implement multistep authentication within an all CA SiteMinder/SOA Security Manager environment.

How the Chain Authentication Service Model Works

The chain authentication model is appropriate for solutions that require XML messages to flow between multiple web services without further intervention from the requesting web service consumer.

In the chain authentication service model, a single web service is responsible for authenticating all incoming web service requests. This authentication service verifies a web service consumer's identity, and then adds authentication data in the form of WS-Security headers or a SAML Session Ticket assertion to the XML message. It then passes the document to downstream web services for processing.

The following illustration shows the flow of data in the chain authentication model.



1. The web service consumer sends a request for access to a protected web Service in the form of an XML document.
2. The SOA Agent receives the request, extracts credentials and passes them to the Policy Server, which authenticates the web service request with an appropriate authentication scheme.
3. After authentication, the request goes through the authorization process. A response attribute associated with the authorizing policy causes the Policy Server to generate a response which it sends to the SOA Agent, instructing it to return authentication data to the authentication web service.
4. The authentication web service sends the XML message and authentication data to the next web service downstream.
5. Downstream web services are configured so that each passes the XML message and authentication data to the next web service in the chain. The requests are allowed access without having to reauthenticate because of the authentication data supplied with the request message.

The most appropriate authentication schemes for initial authentication of requests from the web service consumer by the authentication web service in the chain authentication model are as follows:

- XML Document Collector Authentication Scheme
- XML Digital Signature Authentication Scheme

The authorizing policy for the authentication web service should trigger one of the following responses:

- WS-Security Responses (appropriate for web services protected by more than one policy store or at multiple sites)
- SAML Session Ticket Responses (appropriate for web services protected by the same policy store)

These responses instruct the SOA Agent to add WS-Security headers or SAML Session Ticket assertions (as appropriate) to the XML request passed to the next downstream web service in the chain, which should then be protected using the corresponding authentication scheme:

- WS-Security Authentication Scheme
- SAML Session Ticket Authentication Scheme

Note: SOA Agents can be configured to accept information from a SiteMinder session (SMSESSION) cookie sent in the HTTP header of a request as a means of authenticating a client and always add such cookies to request headers upon successful authentication and authorization. Therefore CA SiteMinder session cookies can therefore be used to implement chain authentication within an all CASiteMinder/SOA Security Manager environment.

SiteMinder Session Ticket Support

Although SOA Security Manager is primarily designed to provide message content-based security for web services, it also provides limited support for SiteMinder session ticket-based session management. A SiteMinder session ticket contains basic information about the user account associated with a request and that user's authentication information; it can be used to identify the user's session across all sites in a single sign-on SiteMinder/SOA Security Manager environment.

SOA Agents that have access to HTTP header information can be configured to accept and maintain SiteMinder sessions obtained from session tickets associated with a web service request received over HTTP transport.

SOA Agents that support SiteMinder session ticket validation accept the XMLSDKAcceptSMSessionCookie configuration parameter. For more information, see the *SOA Security Manager Agent Configuration Guide*.

Note: For more information about SiteMinder user tickets, see the SiteMinder documentation.

How to Develop and Deploy SOA Security Manager Protected Web Services

To develop a web service implementation protected with SOA Security Manager, do the following:

1. Determine how many web services, locally or at federated sites, will be used to perform the required functionality.
2. Choose an authentication service model by determining the following:
   - How security information is to be obtained from a request and, in a multiple-web service environment, how that information is to be passed between web services.
   - In a multiple-web service environment, the flow of data between web services.
3. For each web service in your web service implementation, determine the following:
   a. Define the service interface. The simplest form of interface for a web service can be specified as a set of XML schemas. These schemas dictate the type of XML document to be sent to the web service and what type of document the sender can expect in return.
   b. Build the web service implementation to accommodate an incoming XML document of the type specified in the interface and turn that XML document into a meaningful set of calls to the integrated back-end systems that the web service exposes.
   c. Deploy your web service implementation to a web server, application server, or ESB protected by a SOA Agent or the SOA Security Gateway. You direct consumers of your web service to send their XML message requests to this URI to access the web service.
   d. Configure SOA Security Manager policies to determine how the SOA Agent should authenticate, authorize, and process the XML message before it passes it onto the web service implementation for handling.

      Once it receives a message from the SOA Agent, the web service should return an applicable XML response to the calling web service consumer application or the next.

Chapter 2: SOA Security Manager Configuration Overview

This section contains the following topics:

Policy Server Configuration and Management Interfaces (see page 35)
Policy Server Object Types (see page 36)
Policy Management Methods (see page 40)

Policy Server Configuration and Management Interfaces

The majority of Policy Server configuration tasks are performed by manipulating Policy Server objects using the Administrative UI, use of which is described in the remainder of this book.

Policy Server management tasks that you perform using the Policy Server Management Console.

The management tasks controlled by the Policy Server Management Console include the following:

- Starting and stopping Policy Server processes
- Configuring Policy Server Executives
- Cache Management
- Key Management
- Global Settings
- User Management

Note: For more information on the Policy Server Management Console, see the *Policy Server Administration Guide*.

More information:

Policy Server Object Types (see page 36)

Policy Server Object Types

The Policy Server uses the following three main categories of objects:

- Infrastructure objects
- Policy objects
- Global objects

Infrastructure Objects

You use infrastructure objects throughout a SOA Security Manager deployment. System objects include connections to existing user directories, administrators, Agents, authentication schemes, registration schemes, and password policies.

Infrastructure objects include:

Agents

An Agent is installed on a web server, application server, or other network entities to secure access to resources. Once an Agent is installed on a server, you must configure a SOA Security Manager object for the Agent in the Administrative UI.

Agent Groups

An Agent group is a Policy Server object that points to a group of Agents. The Agents in the group can be installed on different servers, but all of the Agents protect the same resources. Typically Agent groups are configured in SOA Security Manager for groups of servers that distribute the workload for access to a popular set of resources.

Agent Configuration Objects

An Agent Configuration Object holds configuration parameters for one or more Web Agents.

Host Configuration Objects

A Host Configuration Object holds configuration parameters for the Trusted host.

User Directories

A user directory in SOA Security Manager is an object that contains details for connecting to an existing user directory that is external to SOA Security Manager. User directory connections let you configure a connection to an existing user directory, instead of replicating user information within SOA Security Manager.

Policy Domains

A policy domain is a logical grouping of one or more user directories, administrators, and realms. This Policy Server object is the basis for entitlement data. By creating policy domains, an administrator creates a container for entitlements that surround a particular groups of resources (realm), as well as the users who may access the resources, and the administrator who sets up entitlements.

Affiliate Domains

An affiliate domain is a logical grouping of SAML affiliates associated with one or more user directories and administrators.

Note: An affiliate domain must be created using the Federation Security Services Administrative User Interface. More information on affiliate domains exists in the *Federation Security Services Guide*.

Administrators

An administrator is an object that contains profile information for a SOA Security Manager administrator account. Everyone who logs into SOA Security Manager is considered an administrator. The privileges and activities of an administrator account vary by administrative role.

Authentication Schemes

An authentication scheme is a Policy Server object that determines the credentials a user will need to access a protected resource. Authentication schemes are assigned to realms. When a user tries to access a resource in a realm, the authentication scheme of the realm determines the credentials that a user must supply in order to access the resource.

Registration Schemes

A registration scheme is a Policy Server object that allows users to register themselves for access to a group of resources on a network and administrators to manage registered users. Registration schemes simplify the task of managing a large user database.

Agent Types

An Agent Type is a Policy Server object that defines the actions and response attributes supported by a type of Agent, such as Web, Affiliate, RADIUS, or custom.

SQL Query Schemes

A SQL Query Scheme is an object that stores SOA Security Manager SQL queries. These queries are used to retrieve information, such as a list of user groups, from relational databases used as SOA Security Manager user directories.

Password Policies

Password policies are Policy Server objects that contain rules for passwords, including expiration dates, constraints, and composition requirements.

SAML Affiliations

A SAML affiliation is a group of SAML 2.0 entities that share a name identifier for a single principal.

Note: A SAML affiliation must be created using the Federation Security Services Administrative User Interface. More information on SAML affiliations exists in the *Federation Security Services Guide*.

Trusted Hosts

A Trusted Host object represents the client component that connects to the Policy Server.

Policy Objects

Applications

An application is a Policy Server object that defines a complete security policy for one or more related web services. Applications associate users or roles with entitlements (rules) to determine what user accounts can access what web service application resources.

Application objects provide a simplified *enterprise policy management* model that does not require an in-depth knowledge of SOA Security Manager-specific concepts and object types.

Policy Domains

A policy domain is a group of objects that deal with a specific domain of resources. For example, an organization may divide its web service resources by business unit, creating a policy domain for marketing, a separate policy domain for engineering, and so on. Domain objects are those objects that pertain to a specific policy domain. These objects include rules and policies for controlling access to resources.

Policy domains objects are the basis of the traditional SOA Security Manager policy model. They are also the container for the following domain objects that define the security policy for the resources within the domain:

Realms

A realm is a Policy Server object that identifies a group of resources. Realms typically define a directory or folder and possibly its subdirectories.

Rules

A rule is a Policy Server object that identifies a resource and the actions that will be allowed or denied for the resource. Rules can also include actions associated with specific events, such as what to do if a user fails to authenticate correctly when asked for their credentials.

Rule Groups

A rule group is a Policy Server object that contains multiple rules. Rule groups are used to tie together different rules that will be used in a single policy.

Responses

A response is a Policy Server object that determines a reaction to a rule. Responses are included in policies, and take place when a rule is triggered.

Response Groups

A response group is a Policy Server object that contains a logical grouping of responses. Response groups are most often used when many responses will be included in a policy.

Policies

A policy is a Policy Server object that binds users, rules, responses, and optionally, time restrictions and IP address restrictions together. Policies establish entitlements for a SOA Security Manager protected entity. When a user attempts to access a resource, the policy is what SOA Security Manager ultimately uses to resolve the request.

Variables

A variable is an object that can be resolved to a value which you can incorporate into the authorization phase of a request. The value of a variable object is the result of dynamic data and is evaluated at runtime.

Global Objects

In addition to configuring policies for specific resources in a domain, you can also configure global policy objects that apply to all resources.

Global objects include:

Global Rules

A global rule is a Policy Server object that specifies a filter used to apply a global policy to a large group of resources.

Global Responses

A global response is a Policy Server object that determines a reaction to a global rule. Global responses are included in global policies, and take place when a global rule is triggered.

Global Policies

A global policy is a Policy Server object that binds users, global rules, global responses, and optionally, time restrictions and IP address restrictions together. When a user attempts to access a resource, the global policy is what SOA Security Manager ultimately uses to resolve the request.

Policy Management Methods

SOA Security Manager provides two policy management methods for securing your SOA resources:

Enterprise Policy Management Using Application Objects

The recommended method for creating and managing new security policies for your SOA environment is to define application objects that represent one or more related web services and then generate the component and resource settings that define what to protect from associated WSDL files.

Note: Application objects do not support policy expressions using variable objects. Content-based authorization using variables must be implemented using traditional policy management.

Traditional Policy Management Using Policy Domains and Policy Domain Objects

For Policy Server administrators already comfortable with earlier releases of SOA Security Manager or TransactionMinder, traditional policy management — using policy domains and domain objects (realms, rules, responses, polcies, and so on) — can still be used to perform manual configuration of security policies for web service resources.

Traditional policy management must also be used to modify policies created traditionally and migrated from a previous SOA Security Manager deployment or to implement content-based authorization using variables.

Important! While traditional policy management provides all the same policy objects as in previous releases, the user interface is different — you must use the Administrative UI; even if available, you must not use the Policy Server User Interface to create new or manipulate existing policies.

More information:

Traditional Policy Management Overview (see page 315)

Chapter 3: Administrative User Interface Management

This section contains the following topics:

Administrative UI Overview (see page 41)
Start the Administrative UI (see page 42)
Manage Policy Server Objects (see page 42)
SiteMinder Administrators (see page 46)
How a SOA Agent and Policy Server Calculate Time (see page 57)

Administrative UI Overview

The Policy Server is managed through a graphical user interface. The interface is generated dynamically based on the administrative privileges of the user. This chapter discusses how to login to the Administrative UI and the common procedures that you will use while configuring and managing Policy Server objects.

The Administrative UI contains two panes:

- Menu pane - a menu of tasks on the left
- Task pane - the current task on the right

The menu of tasks on the left can be open or closed. If the menu is closed, you can open it by clicking the right-facing arrow. Likewise, if the menu is open, you can close it by clicking the left-facing arrow.

Important! When working on the task pane on the right, always save your changes before opening or closing the menu pane on the left or navigating to another task.

Start the Administrative UI

To use the Administrative UI, you must login as a valid administrator.

To start the Administrative UI

1. Open your browser.
2. Type http://*hostname.domain*/iam/siteminder in the URL field and click Enter.

   hostname

   Specifies the name of the machine on which the Policy Server is installed.

   domain

   Specifies the fully qualified domain name.

   Example: http://siteminder.security.com:81/iam/siteminder

   Note: You must use a fully qualified host name in the URL. If the Web server on the host machine is not running on the standard HTTP port (81), you must append a colon and a port number to the end of the domain.

   The log in screen appears.
3. Enter a valid user name and password in the appropriate fields.
4. Click Login.

   The system displays the relevant tabs for your administrator privileges. The contents of this window differ based on the privileges of the administrator account you use to login to the UI.

   For more information on administrators, see Administrator's Overview.

More information:

Default Administrator Account for the Administrative UI (see page 47)

Manage Policy Server Objects

The Administrative UI lets you view, modify, and delete Policy Server objects. Although the details of each task differ by object, the general methods are similar. For example, the procedure for deleting an Agent is similar to the procedure for deleting a response.

The following sections describe the general tasks for viewing, modifying, and deleting Policy Server objects. Other chapters in this guide describe how to create the Policy Server objects necessary to manage and secure resources.

Duplicate Policy Server Objects

The easiest way to create a new Policy Server object is to copy an existing object and modify its properties. You can use the properties of the existing object as a template, only changing the information that is different for the new object.

Note: The copy option is not available for the following objects:

- Agent Type
- AuthAz Directory Mapping
- AuthValidate Directory Mapping
- Certificate Mapping
- User Directory
- Application
- Application Resource
- Domain
- Policy
- Realm
- Response
- Response Attribute
- Rule
- Global Policy
- Global Response
- Global Rule
- Password Policy
- Administrator

Note: Your administrative privileges determine the objects you can access.

To create a new object by copying and modifying an existing object

1. Click <tab>, <Policy Server category>.

   Example: Click Infrastructure, Agent.

2. Click <Policy Server object>, Create <Policy Server object>.

   The Create Object pane opens.

   Example:

   Click Agent, Create Agent.

3. Select Create a copy of an object, specify search criteria, and click Search.

   A list of objects that match the search criteria opens.

4. Select an object from the list, and click OK.

   The Create Object: *Name* pane opens.

5. Type a new name and description in the fields on the General group box.
6. Modify the properties that are different for the new object, and click Submit.

   The Create Object task is submitted for processing.

View Policy Server Object Properties

You can view the properties of a Policy Server object.

Note: Your administrative privileges determine the objects you can access.

To view the properties of an object

1. Click <tab>, <Policy Server category>.

   Example: Click Policies, Domains.

2. Click <Policy Server object>, View <Policy Server object>.

   The View Object pane opens.

   Example:

   Click Domain, View Domain.

   The View Domain pane opens.

3. Specify search criteria, and click Search.

   A list of objects that match the search criteria opens.

4. Select an object from the list, and click Select.

   The View Object pane opens.

   Note: To view another Policy Server object, click Return to Search. To close the pane, click Close.

Modify an Existing Policy Server Object

The Administrative UI lets you modify the properties of existing Policy Server objects.

Note: Your administrative privileges determine the objects you can access.

To modify the properties of an existing object

1. Click <tab>, <Policy Server category>.

   Example: Click Policies, Domains.

2. Click <Policy Server object>, Modify <Policy Server object>.

   The Modify Object pane opens.

   Example:

   Click Realm, Modify Realm.

   The Modify Realm pane opens.

3. Specify search criteria, and click Search.

   A list of objects that match the search criteria opens.

4. Select an object from the list, and click Select.

   The Modify Object: *Name* pane opens.

5. Modify the object's properties, and click Submit.

   The Modify Object task is submitted for processing.

More information:

Start the Administrative UI (see page 42)

Delete a Policy Server Object

You can delete a Policy Server object that is no longer needed.

Note: Your administrative privileges determine the objects you can access.

To delete an object

1. Click <tab>, <Policy Server category>.

   Example: Click Infrastructure, Authentication.

2. Click <Policy Server object>, Delete <Policy Server object>.

   The Delete Object pane opens.

   Example:

   Click Authentication Scheme, Delete Authentication Scheme.

   The Delete Authentication Scheme pane opens.

3. Specify search criteria, and click Search.

   A list of objects that match the search criteria opens.

4. Select an object from the list, and click Select.

   A confirmation pane opens.

   Note: You can select more than one object at a time.

5. Click Yes.

   The Delete Object task is submitted for processing.

More information:

Start the Administrative UI (see page 42)

SiteMinder Administrators

A SOA Security Manager administrator is anyone who has access to Policy Server objects and tools. Depending on a person's role in an organization, SOA Security Manager administrators have access to different resources and features, and are responsible for different tasks.

The SOA Security Manager administrative model lets you implement fine-grained administrative privileges, so you can organize the management of Policy Server objects and SOA Security Manager tools across a few or many individuals in an organization.

Certain administrative responsibilities can overlap. For example, a Policy Manager and Sales Manager may both be able to make changes to the Sales policy domain and the objects in the policy domain.

When you install the Administrative UI, you specify a default administrator account during the Administrative UI registration process. This account has maximum privileges, and with it you can create additional administrator accounts to distribute administrative tasks.

The following administrators can be configured in the Administrative UI; they serve different functions:

Administrator

An administrator can do the following:

- manage the SOA Security Manager Administrative UI
- manage Policy Server tools, such as XPSImport, XPSExport, smobjimport, and smobjexport.

When you configure an administrator to manage the Administrative UI, the privileges granted to the administrator control what he sees in the Administrative UI. You can create administrators and assign privileges to each administrator to match the administrative roles that exist in your organization.

Legacy Administrator

A legacy administrator has the ability to manipulate the Policy Management API. If your environment includes a script or program that uses the Policy Management API, you need to create a legacy administrator that has authentication privileges to execute the functions via the Policy Management API.

In addition to the API function, the legacy administrator can be a Trusted Host Administrator. A Trusted Host administrator has the right to run the host registration process for a host where a SOA Security Manager Agent resides, enabling the Agent to communicate with the Policy Server.

Default Administrator Account for the Administrative UI

When you install the Administrative UI, you establish a Super User account. The account has the maximum system privileges. You can use this account to configure any type of Policy Server object, including other administrator accounts.

Create an Administrator

An administrator can manage the Administrative UI and the Policy Server tools.

Use the Administrative UI to establish an administrator and define the tasks and rights for this administrator.

Before you create an administrator, consider the following:

- Rights that each administrator needs to perform their job.
- Across how many administrators should the Administrative UI and tools tasks be distributed
- Will an administrator be responsible for application security policies

Important! An administrator can only create another administrator with the same or lesser privileges. For example, if an administrator with GUI and reports privileges, he cannot create an administrator with GUI, reports privileges and local API privileges.

To create an administrator

1. Click Administration, Administrators, Administrator, Create Administrator.

   The Create Administrator dialog opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

2. Specify a Name and optionally, a description for the administrator you are creating in the General group box.

3. Enter a value for the User Path in the Details group box.

   You can either type a value for the user path or click Lookup to search for users in the current user store configured for the Administrative UI.

4. (Optional) Select Super User to grant all rights to the administrator and then click Submit. Otherwise, go to the next step.

5. Specify how the administrator is permitted to interact with the Policy Server in the Access Methods group box. Check as many methods as required for the administrator to perform tasks.

   For example, if an administrator is going to use the XPSImport and XPSExport tools, check the Import Allowed and Export Allowed.

6. Click on Create in the Rights group box

   The Select category dialog opens.

7. Complete the following steps in the Select Category dialog:

   a. Select the security categories, that is, the tasks that you want the administrator to manage then click Next.

   b. If you selected Policy Administration or Application Administration as a security category, check all the appropriate boxes in the Select scope dialog to determine the domains that the administrator can control then click Next.

   c. Check all the relevant permissions (View, Manage, Protect, eXecute) in the Select permissions dialog for each administrative tasks.

   d. Click Finish.

   The administrator rights have been granted. You return to the Create Administrator dialog.

8. Click Submit.

You have defined an administrator.

Disable an Administrator Account

You can temporarily disable an administrator without deleting the administrator's account. This feature is helpful if you want to reinstate the administrator's privileges later on without recreating a new account.

To disable an administrator account:

1. Click Administration, Administrators, Administrator, Modify Administrator.

   The Search dialog opens.

2. Search for the administrator account you want to disable and then select the appropriate entry.

   The Modify Administrator dialog opens.

3. Click Disabled in the Details group box.
4. Click Submit.

The administrator is now disabled.

You can re-enable the administrator at any time by repeating this procedure and clearing the Disabled box and submitting the change.

Administrator Use Case

This use case for the Policy Server administrative model illustrates how administrators are created and how an existing administrator can create and grant privileges to a new administrator.

This scenario involves three administrators

- SuperAdmin
- ManagerAdmin
- JuniorAdmin

Using the three administrators, the following scenarios are described:

- SuperAdmin creating the ManagerAdmin (see page 50)
- ManagerAdmin creating the JuniorAdmin (see page 52)

The following terms are used throughout the use case:

Access Method

Specifies how an administrator interacts with the Policy Server.

Security Category

Specifies a functional area in which an administrator can execute tasks to manage Policy Server objects or tools.

Scope

Indicates whether the administrator's privileges extend to all domains and applications or to only specific domains and applications.

Permissions

Determines whether an administrator can read, manage, propagate, or execute a tasks related to a security category.

Rights

Defines the administrator's complete set of privileges based on the grouping of the security category, scope and permissions.

SuperAdmin Grants Privileges to ManagerAdmin

The SuperAdmin is an administrator created with the super user option (-su) set during the Administrative UI registration. This means that the SuperAdmin has the ability to assign all categories, rights and scope to any other administrator.

From the Administrative UI, the SuperAdmin creates a new administrator named ManagerAdmin. Initially, ManagerAdmin has no privileges until the SuperAdmin assigns them.

Initial Privileges for ManagerAdmin

The SuperAdmin initially assigns the following to ManagerAdmin:

Access Method

GUI Allowed

Rights

Security Category	Scope	Permissions*
Admin Administration	All	V, M
Agent Administration	All	V, M
Application Administration	All	V, M, P
Policy Administration	Domain 1	V, M, P

* Permissions: View, Manage, Propagate, eXecute (only for executing reports)

Important! The Propagate permission allows one manager to assign the category to another administrator.

At this stage, the SuperAdmin can change the permissions of the existing security categories.

Additional Privileges for ManagerAdmin

The SuperAdmin wants to assign an additional privilege to ManagerAdmin. Based on the categories already assigned to ManagerAdmin, the Security Category list from which the SuperAdmin can choose is slightly modified. All categories are displayed except the Agent and Admin Administration categories because they are already assigned to ManagerAdmin. Additionally, they cannot be assigned a scope so there is nothing that can be modified. The Admin Administration category is not displayed because the scope assigned is ALL so there is nothing to modify.

The only category still available from the original set of categories is Policy Administration because this category can be assigned a scope, which means that privileges can be applied to specific domains or applications. When SuperAdmin selects the Policy Administration category, the scope dialog displays a list that includes ALL as a selection as well as a complete list of domains, with the exception of Domain1, which ManagerAdmin has already been assigned.

Note: The Application Administration is a scoped category like Policy Administration; however, ALL has already been defined as the scope for this category so there is no need to redisplay this category as a choice.

SuperAdmin selects Domain2, extending ManagerAdmin's rights across a second domain.

ManagerAdmin's complete rights are now as follows:

Security Category	Scope	Permissions*
Admin Administration	All	V, M
Agent Administration	All	V, M
Application Administration	All	V, M, P
Policy Administration	Domain1	V, M, P
Policy Administration	Domain2	V, M, P

* Permissions: View, Manage, Protect, eXecute (only for executing reports)

ManagerAdmin Assigns Privileges to JuniorAdmin

The second part of this use case shows how privileges can be assigned from an Administrator with a specific set of privileges to another administrator.

ManagerAdmin creates a new manager named JuniorAdmin. When ManagerAdmin goes to select the Security Categories for JuniorAdmin, only the following two are available:

- Application Administration
- Policy Administration

ManagerAdmin can only assign those two categories to JuniorAdmin because ManagerAdmin only has the P (protect) permission for those two categories.

Important! The protect permission allows the one manager to assign the category to the another administrator.

ManagerAdmin proceeds to assign access methods and rights to JuniorAdmin as follows:

Access Method

GUI Allowed

Rights

- ManagerAdmin chooses the Application Administration category. ManagerAdmin has ALL for the Application Administration scope so when he chooses this category for JuniorAdmin he sees a list of all the applications available in the system. In this case, he assigns ALL as the scope for JuniorAdmin, but with only the permission to view the applications.
- ManagerAdmin chooses the Policy Administration category. He only has a choice of Domain1 and Domain2 in the scope dialog. He chooses Domain1 for JuniorAdmin and gives JuniorAdmin only view and manage permissions.

  Note: When choosing categories, ManagerAdmin only had the Policy Administration category available because he previously assigned the Application Administration category, which is the only other available choice.

  If ManagerAdmin wants to assign Domain2 later on, this will be the only domain available in the scope list because Domain1 is already assigned.

JuniorAdmin's final rights are as follows:

Security Category	Scope	Permissions*
Application Administration	All	V

Security Category	Scope	Permissions*
Policy Administration	Domain1	V, M
Policy Administration	Domain2	V, M

* Permissions: View, Manage, Propagate, eXecute (only for executing reports)

Create a Legacy Administrator

A legacy administrator has the ability to manipulate the Policy Management API. A legacy administrator also can use legacy command-line tools that require a username and password, such as smobjexport, smobjimport and smreg.

Use the Administrative UI to configure a legacy administrator and define the tasks that this administrator can perform via the Policy Management API.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure a legacy administrator

1. Click Administration, Administrators, Legacy Administrator, Create Legacy Administrator.

   The Create Legacy Administrator dialog opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

2. Enter a name and optionally, a description in the General group box.

3. Select one of the following radio buttons in the Administrator Lookup group box to specify the lookup method:

   SOA Security Manager Database

   Choose this option if the administrator authenticates against the SOA Security Manager database (policy store). Also, enter a value for the Password and Confirm Password fields.

   External Directory

   Choose this option if the administrator authenticates against an external directory. Also, select user directory and authentication scheme in the Authenticate group box.

4. Select the administrator privileges from the following options:

   System

   With System privileges, an administrator has access to all policy domains. When you select this radio button, a Task group box displays.

Domain

With Domain privileges, an administrator has access to specific subset of policy domains. When you select this radio button, a Tasks and Scope group box displays. The Tasks group box lists the administrative tasks that can be performed. The Scope group box lists all the available domains that can be managed.

5. Choose the tasks the administrator can perform.

   For a System administrator, select one or more of the following tasks:

   - Manage System and Domain Objects
   - Manage Users
   - Manage Keys and Password Policies
   - Register Trusted Hosts

   For a Domain administrator, select one or more of the following tasks:

   - Manage Domain Objects
   - Manage Users
   - Manage Password Policies

6. If you are creating a Domain administrator, select the domains in the Scope group box that the administrator can manage.
7. Click Submit.

You have defined a legacy administrator.

Note: An administrator cannot create a child administrator with more privileges than the parent administrator.

More information:

Legacy Administrator Privileges (see page 54)

Legacy Administrator Privileges

Administrator privileges are determined by the tasks that you enable for the administrator. These privileges allow administrators to use a set of Policy Server features.

The following tables describe the privileges associated with each combination of administrator task.

System Administrator Tasks

Tasks	Administrative Privilege
Manage System and	■ Create/edit/delete Agents, Agent Configuration

Tasks	Administrative Privilege
Domain Objects	Objects, Agent groups, Agent types, Host Configuration Objects, user directories, policy domains, authentication schemes, directory mappings, certificate mappings, registration schemes, and SQL query schemes. ■ All privileges for Manage Domain Objects. ■ Create/delete parent realms in all domains. ■ Create/edit/delete administrators. ■ Flush all caches, including cached resources. ■ Change global settings. ■ Delete Trusted Hosts **Note:** You cannot create or edit Trusted Host objects with this privilege, only delete them. To register Trusted Hosts, you must have Register Trusted Host privilege.
Manage Users	■ Flush all user session caches, or flush the user session cache of any individual user cache from any directory. ■ Enable/disable users in any directory. ■ Force password change on any user in any directory.
Manage Keys and Password Policies	■ Create/edit/delete password policies. ■ Manage keys.
Register Trusted Hosts	■ Register Trusted Hosts

Domain Administrator Tasks

Tasks	Administrative Privilege
Manage Domain Objects	■ In managed domains: create/edit/delete rules, rule groups, responses, response groups, policies. ■ Edit top-level realms in managed domains (not resource filters). ■ Create/edit/delete nested realms in managed domains. ■ Flush specific realms from the resource cache, and flush all resources (in privileged domains) from the cache.
Manage Users	■ Flush user session caches for individual users in

Tasks	Administrative Privilege
	directories attached to managed domains. ■ Enable/disable users in directories attached to managed domains. ■ Force password change on users in directories attached to managed domains.
Manage Password Policies	■ Create/edit/delete password policies for users in directories attached to managed domains.

Recreate a Deleted UI Administrator

If you are deleting an administrator, be careful not to delete all administrators, especially the Super User administrator.

Important! Proceed with caution when deleting or editing an administrator.

If you accidently delete all administrators, use the registration tool xpsregclient to restore a default Super User administrator. The registration tool is installed with the Administrative UI.

Note: You can find detailed information about the Registration Tool in the *Policy Server Installation Guide*.

Restoring the Super User administrator is a two-step process, involving:

1. Registering the Super User administrator with the host
2. Registering the Admin UI with a Policy Server.

To restore the Super User administrator:

1. Open a command prompt from the machine that is hosting the Policy Server, and enter the following command:
   xpsregclient *client_name* -adminui -su

   This command recreates the default Super User account.

client name

Specifies the name that identifies the Administrative UI that is to be registered.

Limit: This value must be unique. For example, if you have previously used smui1 to register an Administrative UI, enter smui2.

Note: Record this value. This is a required value to complete the registration process from the Administrative UI.

-adminui

Specifies that an Administrative UI is being registered.

-su

Specifies that the user that logs into the Administrative UI to complete the registration process becomes the default Super User.

If the Administrative UI that you are registering uses an administrative user store that contains a Super User account established by a previous Administrative UI registration, you do not have to supply this flag. You can use the same Super User account credentials to log into the Administrative UI you are registering to complete the process.

2. Register the Admin UI with a Policy Server.

   Registering the Admin UI is an administration task in the Administrative UI. More information about this process can be found in the *Policy Server Installation Guide*.

When editing an administrator's record, be careful not to modify the settings for the administrator that you are logged in as. If you change your administrative settings, you may lose access to SOA Security Manager features. If you lock yourself out of a feature, an administrator with full access to the Policy Server must restore your privileges.

How a SOA Agent and Policy Server Calculate Time

For each system that has a Policy Server or SOA Agent installed, you must set the system clock for the time zone appropriate to that system's geographical location. Policy Servers and Web Agents use the time zones to calculate time relative to Greenwich Mean Time (GMT).

The following example shows how the Policy Server executes a policy relative to time. A resource is stored on an Application Server in Massachusetts and is protected by a Policy Server in California. The policy allows access to the resource between 9:00 a.m. and 5:00 p.m. However, the user in Massachusetts can still access the resource at 6:00 p.m. because the policy is based on the Policy Server's time zone, Pacific Standard Time (PST), which is 3 hours behind the SOA Agent's time zone, Eastern Standard Time (EST).

For Windows systems, the time zone and the time of day that you set in the Date/Time control panel must agree. For example, to reset a system in the USA from Eastern Standard Time to Pacific Time, you must set the system's clock back three hours and change the time zone to Pacific Standard Time. If these two settings do not match, single sign-on across multiple domains and agent key management will not work properly.

Chapter 4: Agents and Agent Groups

This section contains the following topics:

How to Configure Policy Server Objects for SOA Agents (see page 59)
Trusted Hosts for SOA Agents (see page 59)
Host Configuration Objects for Trusted Hosts (see page 63)
SOA Agent Configuration Overview (see page 68)
How to Configure a SOA Agent (see page 70)
Agent Configuration Object Overview (see page 74)
Enable a SOA Agent (see page 79)
Agent Groups (see page 79)
Custom Agents (see page 81)

How to Configure Policy Server Objects for SOA Agents

You must configure agent-related objects on the Policy Server before you install a SOA Agents or SOA Security Gateway:

- *Host Configuration Objects* contain trusted host configuration parameters. You must define a Host Configuration Object for each system in your environment on which a SOA Agent or SOA Security Gateway is installed.
- *Agent Objects* name an agent, establishing an agent identity that can be mapped to a specific server. You must define an Agent object for each SOA Agent or SOA Security Gateway in your environment.
- *Agent Configuration Objects* define configurable agent properties. You must define Agent Configuration Objects for all SOA Agents for Web Servers and SOA Agents for IBM WebSphere.

Trusted Hosts for SOA Agents

A trusted host is a client computer where one or more SOA Agents can be installed. The term trusted host refers to the physical system.

Register a Trusted Host with the Policy Server

You register a trusted host from the system where you install a Web Agent; the host registration process is part of the Agent installation and configuration process. After registration is complete, the registration tool creates the SmHost.conf file. After this file is created successfully, the client computer becomes a trusted host. A trusted host must be registered to communicate with the Policy Server.

Note: You cannot create a trusted host using the Administrative UI; you can only view a trusted host once it is registered or delete a trusted host.

Upon initialization, the Web Agent uses the trusted host's configuration settings in the Host Configuration File (SmHost.conf). The Web Agent attempts to connect to the first Policy Server listed under the PolicyServer parameter of the Host Configuration File. If the trusted host fails to connect to the first Policy Server, the trusted host attempts to connect to the next Policy Server listed, if any.

Once the Web Agent connects to its bootstrap Policy Server, the trusted host looks for the Host Configuration Object, named in the hostconfigobject parameter of the Smhost.conf file. You specify this value when you register the trusted host. The trusted host then retrieves its configuration.

Important! You only register the host once, not each time you install and configure a Web Agent.

Trusted Host Configuration Settings

Most of the trusted host configuration settings are set in a Host Configuration Object. The only exceptions are the Policy Server and RequestTimeout parameters, which are set in the Host Configuration file, SmHost.conf.

Settings in the Host Configuration File apply only when the trusted host initializes. Once the trusted host initializes, the settings in the Host Configuration Object take effect.

Request Timeout

Use the RequestTimeout parameter to specify the number of seconds that the trusted host should wait before deciding that a Policy Server is unavailable. This setting allows you to optimize the response time of the Web server.

The default value is 60 seconds.

Note: If the Policy Server is busy due to heavy traffic or a slow network connection, you may want to increase the RequestTimeout value.

Operation Mode

The operation mode determines how the trusted host works with multiple Policy Servers. There are two operation modes: failover and round robin.

Failover

Failover is a redundancy mode. If the primary Policy Server fails, there is a backup Policy Server to take over policy operations. Failover is the default operation mode. When the trusted host initializes, it operates in Failover mode.

In this mode, *every* trusted host request is delivered to the first Policy Server in the list. If that Policy Server does not respond, the trusted host marks it *unavailable* and redirects the request to the next Policy Server in the list. If a previously failed Policy Server recovers, it is returned to its original place in the list.

Round Robin

Round robin mode lets the trusted host distribute requests across multiple Policy Servers, which provides faster access to Policy Servers and therefore, more efficient user authentication and authorization. It also prevents a single Policy Server from becoming overloaded with requests.

In this mode, the trusted host delivers a request to the first Policy Server in the list. The next request is delivered to the second Policy Server in the list, and so on, until the trusted host has sent requests to all the available Policy Servers. After sending requests to all of the Policy Servers, the next request returns to the first Policy Server in the list and the cycle begins again.

If a Policy Server fails, the request is redirected to the next Server in the list. The trusted host marks the failed Server as *unavailable* and redirects all of the requests to other servers. After the failed server recovers, it is automatically restored to its original place in the list.

Implement an Operation Mode

The operation mode determines how the trusted host works with multiple Policy Servers. There are two operation modes: failover and round robin.

To implement an operation mode

1. Configure more than one Policy Server**.**
2. Configure all Policy Servers to use a common policy store.
3. Set the EnableFailover parameter.
   - To enable failover, set the EnableFailover parameter to yes.
   - To enable load balancing, set the EnableFailover parameter to NO.

   The value for the EnableFailover parameter applies to all Policy Servers specified in the Host Configuration Object.

TCP/IP Connections

The trusted host and Policy Server communicate across TCP/IP connections. The number of available TCP/IP connections between the trusted host and Policy Server is determined by the available sockets for the Policy Server's authorization, authentication, and accounting ports.

The number of sockets per port controls the number of simultaneous threads accessing the Policy Server from the Web server. Each user access request is handled by a separate Web server thread, which requires its own socket. The Web server maintains a pool of threads for requests and only creates a new one when there are no more available threads. As traffic increases, the number of sockets per port needs to increase.

There are several settings that affect the TCP/IP connections between the trusted host and the Policy Server.

Maximum Sockets Per Port

Defines the maximum number of TCP/IP connections used by the trusted host to communicate with the Policy Server. By default, this value is set to 20, which is generally sufficient for low- and medium-traffic Web sites. If you are managing a high-traffic Web site or if you have defined agent identities for virtual servers, you may want to increase this number.

Minimum Sockets Per Port

Determines the number of TCP/IP connections open for the Policy Server at start up. The default value is 2. If you are managing a high-traffic Web site, you may want to increase this number.

New Socket Step

Specifies the number of TCP/IP connections that the Agent opens when new connections are required. The default value is 2. Modify the number of sockets that should be added at each required increment if you require more sockets.

Note: More information about these values and how you may have to adjust the values as your SOA Security Manager environment grows exists in the *Policy Server Administration Guide.*

Delete Trusted Host Objects

You delete a trusted host when you are re-registering it. You re-register a host using the Registration Tool: smreghost. This tool is installed with the Web Agent.

Note: You can run the Web Agent Configuration Wizard to re-register a trusted host, but you must delete or rename the SmHost.conf file or the Wizard does not prompt you to register a trusted host. We recommend you use the smreghost tool. More information on registering and re-registering trusted hosts exists in the *Web Agent Installation Guide*.

To delete a trusted host

1. Click Infrastructure, Hosts, Trusted Hosts.
2. Click Delete Trusted Host.

   The Search dialog opens.
3. Specify search criteria, and click Search.

   A list of trusted hosts that match the search criteria opens.
4. Select a trusted host from the list and click Select.

   Note: You can select more than one trusted host at a time.

   You are prompted to confirm the deletion of the host.
5. Click Yes.

   The Trusted Host is deleted.

Host Configuration Objects for Trusted Hosts

Host Configuration Objects hold configuration settings for trusted hosts. After a trusted host connects to a Policy Server, it uses the settings in the Host Configuration Object.

On the Web Agent side, the Host Configuration Object being used by the trusted host is identified in the hostconfigobject parameter of the SmHost.conf file. The settings in the SmHost.conf file are used by the Web Agent during the initialization phase of each web server child process. This means that the SmHost.conf file is not only used at start-up, but may also be used at run time by web servers that start new child processes to add capacity or to replace processes that stop unexpectedly. After initialization, the settings in the Host Configuration Object are used.

You need to create a Host Configuration Object before you can create trusted host objects.

Copy a Host Configuration Object

The recommended way to create a Host Configuration object is to copy an existing Host Configuration object and modify its properties. You can copy the DefaultHostSettings object and use its properties as a template for the new object.

Important! The name of a Host Configuration Object must be unique; do not use the same name as an existing Agent object. If you use a name assigned to another Web Agent, a message displays stating that the trusted host already exists.

To copy a host configuration object

1. Click Infrastructure, Hosts.
2. Click Host Configuration, Create Host Configuration.

   The Create Host Configuration pane opens.
3. Select the Create a copy radio button and do one of the following:
   - Select one of the default Host Configuration Objects listed.
   - Click Search and choose from the list of Host Configuration Objects that is displayed.
4. Click OK.

   The Create Host Configuration: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Type a new name and description in the fields on the General group box.
6. Modify the properties that are different for the new object, and click Submit.

   The Create Host Configuration task is submitted for processing.

Create a Host Configuration Object

You can create a new Host Configuration object or duplicate an existing object.

To create a host configuration object

1. Click Infrastructure, Hosts.
2. Click Host Configuration, Create Host Configuration.

   The Create Host Configuration pane opens.

3. Do one of the following:
   - (Recommended) Create a copy of an existing Host Configuration object and modify its properties. You can copy the DefaultHostSettings object and use its settings as a template for the new object. The DefaultHostSettings object is installed by the Policy Server installation program.

     Important! Do not directly modify and use the DefaultHostSettings object. Always copy this object and then modify it.
   - Create a new object.
4. Click OK.

   The Create Host Configuration: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Type the name and a description in the fields on the General group box.
6. Specify the Host Configuration settings in the Configuration Values group box.
7. Click Submit.

   The Create Host Configuration task is submitted for processing.

Add Multiple Policy Servers to the Host Configuration Object

A trusted host can access multiple Policy Servers. To set up trusted host connections and failover or round robin operation, you must add the Policy Servers to a Host Configuration object.

To add a Policy Server to a Host Configuration object

1. Click Infrastructure, Hosts
2. Click Host Configuration, Modify Host Configuration.

   The Modify Host Configuration pane opens.
3. Specify search criteria, and click Search.

   A list of Host Configuration objects that match the search criteria opens.
4. Select a Host Configuration object, and click Select.

   The Modify Host Configuration: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Click Add on the Clusters group box.

   The Add Cluster group box opens.

6. Type the IP address and port number of the Policy Server that you are adding to the cluster in the Host and Port fields, respectively, on the Add Cluster group box.

   Note: To add another Policy Server to the cluster, click Add to Cluster. To delete a Policy Server from the cluster, click the minus sign to its right. To change the sequence of Policy Servers in the cluster, click the up and down arrows.

7. Click OK.

   The cluster is added to the Clusters group box.

   Note: To modify a cluster, click the right-facing arrow to its left. To delete a cluster, click the minus sign to its right. To add another cluster, click Add on the Clusters group box, and repeat steps 6 and 7.

8. Enter a failover threshold in the Failover Threshold Percent field.

   Note: If the percentage of active servers in a cluster falls below the specified percentage, the cluster fails over to the next available cluster in the cluster list.

9. Click Submit.

   The Modify Host Configuration task is submitted for processing.

More information:

Delete Trusted Host Objects (see page 63)
Operation Mode (see page 61)
Host Configuration Objects for Trusted Hosts (see page 63)

Configure Policy Server Clusters for a Host Configuration Object

You can configure multiple Policy Servers in a cluster for failover operation. Clustering servers enables failover from one group of servers to another group.

Policy Server clusters are defined as part of a Host Configuration Object. When a SOA Security Manager Web Agent initializes, the settings from the Host Configuration Object are used to setup communication with Policy Servers.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure a cluster

1. Click Infrastructure, Hosts.
2. Click Host Configuration, Create Host Configuration.

   The Create Host Configuration pane opens.

3. Verify that Create a new object is selected, and click OK.

   The Create Host Configuration: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

4. Type the name and a description of the Host Configuration object in the fields on the General group box.

5. Specify the Host Configuration settings in the fields on the Configuration Values group box.

6. Click Add on the Clusters group box.

   The Add Cluster group box opens.

7. Type the IP address and port number of the Policy Server that you are adding to the cluster in the Host and Port fields, respectively, on the Add Cluster group box.

   Note: To add another Policy Server to the cluster, click Add to Cluster. To delete a Policy Server from the cluster, click the minus sign to its right. To change the sequence of Policy Servers in the cluster, click the up and down arrows.

8. Click OK.

   The cluster is added to the Clusters group box.

   Note: To modify a cluster, click the right-facing arrow to its left. To delete a cluster, click the minus sign to its right. To add another cluster, click Add on the Clusters group box, and repeat steps 7 and 8.

9. Type a percentage in the Failover Threshold Percent field on the Clusters group box.

   Note: If the percentage of active servers in a cluster falls below the specified percentage, the cluster fails over to the next available cluster in the cluster list.

10. Click Submit.

    The Create Host Configuration task is submitted for processing.

    Important! The Configuration Values group box specifies a single Policy Server and a simple failover operation that are only used when no clusters are specified in the Clusters group box. If you decide to delete all clusters in favor of a simple failover operation, be sure to delete all Policy Server information from the Clusters group box.

SOA Agent Configuration Overview

The two options for configuring a SOA Agent are:

Central configuration

Indicates that the SOA Agent is configured from the Policy Server. The policy store holds the set of configuration parameters to be used by a group of SOA Agents. Parameters are configured using the Administrative UI.

The Agent configuration is specified in an Agent Configuration Object.

Local configuration

Indicates that the SOA Agent is configured from a local configuration file on each web server where the Agent is installed.

You can store some parameters centrally and others locally.

Note: You can only enable and disable the SOA Agent from the local Agent configuration file, not from the Policy Server. This is true whether you are configuring centrally or locally.

Advantages of Centrally Configuring SOA Agents

When you centrally configure SOA Agents, the settings are stored in the policy store, not on a local configuration file on the protected server.

Compared with local configuration, central configuration provides:

Improved Usability When Using Central Agent Configuration

- Updating the configuration settings of multiple SOA Agents is easier and faster. You can change the setting in one place and have it propagate to multiple SOA Agents.
- Installing the SOA Agent no longer requires the administrator to specify a shared secret. The Policy Server generates the shared secret.

Added Security with Central Agent Configuration

- It is easier to control access to the SOA Agent configuration. For example, you can prevent the Web Server administrator from changing the configuration settings.
- You can store the configuration settings behind an internal firewall, instead of on the protected server.
- You can use a hardware-bound key to generate the shared secret, which is used by the client side to establish a secure connection to the Policy Server. The trusted host handles the connection to the Policy Server.

SOA Agent Components

On the Agent-side of a SOA Security Manager network, there are several main components involved in SOA Agent operation:

SOA Agent

Virtual interface to a web server, application server, or EJB; triggers rules and enforces policies

Trusted Host

A client computer where one or more SOA Agents is installed. It handles the connection to the Policy Server. The term trusted host refers to the physical system. You can have more than one trusted host on a physical server, but each must be identified by a unique name.

The trusted host is "trusted," because it is registered with the Policy Server. You must register a trusted host so the SOA Agents installed on that host can communicate with the Policy Server.

A trusted host is identified by the following data:

- Host name
- Host IP address
- Host description
- Shared secret
- Host Object Identifier (OID)

Agent Configuration File (WebAgent.conf or JavaAgent.conf)

Stored on the server where the Agent resides, this file is used for local configuration. It holds the Agent configuration parameters for each SOA Agent.

Host Configuration File (SmHost.conf)

Stored on the server where the SOA Agent resides, this file holds initialization parameters for the trusted host. Once the trusted host connects to a Policy Server, the trusted host uses the settings in the Host Configuration Object stored at the Policy Server. The Host Configuration Object is named in the hostconfigobject parameter of this file.

Policy Server Objects Related to SOA Agents

On the Policy Server-side there are three policy objects related to SOA Agent configuration:

Agent object

Names the Agent, establishing an Agent identity that can be mapped to a specific server.

Agent Configuration Object

Contains the SOA Agent configuration parameters. Use an Agent Configuration Object to centrally manage a group of SOA Agents. Though this object is primarily for central Agent configuration, it also contains the parameter that tells the Policy Server to use local configuration.

Host Configuration Object

Contains the trusted host configuration parameters. Except for initialization parameters, trusted host parameters are always maintained in a Host Configuration Object.

Configuration objects are stored in the policy store. Use the Administrative UI to create, modify, and view configuration objects.

How to Configure a SOA Agent

There are a number of tasks that must be completed in order to fully configure a SOA Agent. These tasks apply to local and central configuration of a SOA Agent.

Note: You must set up the Policy Server for SOA Agent communication before you install a SOA Agent and register a trusted host.

To configure a SOA Agent

1. Install a Policy Server.
2. Create a Host Configuration Object.
3. Grant the Register Trusted Hosts privilege to a Policy Server Administrator. An administrator must have the privilege to register trusted hosts.

   Note: If you create an administrator with *only* the Register Trusted Hosts privilege, that administrator will not be able to use the Administrative UI.
4. Create an Agent object to name the Agent. Do not confuse this object with an Agent Configuration Object.
5. Create an Agent Configuration Object.

   If you plan to configure an Agent locally, you still need this object to enable the local configuration parameter, AllowLocalConfig.
6. At the client site, install the SOA Agent.

7. Register the trusted host. Part of this process is to provide the name of the Host Configuration Object that you already created at the Policy Server.
8. When the Agent-related policy objects are configured, enable the SOA Agent. This setting is in the local Agent configuration file.

Note: The *SOA Security Manager Agent Configuration Guide* contains all the parameter descriptions, the default values, and instructions on setting the parameters. Whether you are configuring a SOA Agent centrally or locally, see this guide for parameter descriptions. Additionally, information about SOA Agents and the trusted host registration process exists in the *SOA Security Manager Implementation Guide.*

Configure SOA Agents Centrally

To centrally configure SOA Agents, perform the steps outlined in How to Configure a SOA Agent. These tasks apply to local and central configuration of a SOA Agent.

If you specify any configuration parameters locally, the parameter values in the local Agent configuration file override the values in the corresponding Agent Configuration Object, merging the input from both configuration sources.

To use a local configuration exclusively, without combining input from an Agent Configuration Object and an Agent configuration file, configure the Agent Configuration Object with only the AllowLocalConfig parameter and set it to yes. This ensures that the Web Agent will only have configuration data from the local configuration file.

To better understand how central and local configuration work together, read Combined Central and Local Configuration.

More information:

How to Configure a SOA Agent (see page 70)
Combined Central and Local Configuration (see page 72)

Configure SOA Agents Locally

The SOA Agent reads both the Agent Configuration Object and the local Agent configuration file, overriding values in the Agent Configuration Object with the values in the local Agent configuration file. The SOA Agent merges them together into one configuration source. This enables you to modify only a small subset of Agent parameters locally, then rely on the central Agent Configuration Object for the rest of an Agent's configuration.

To better understand how central and local configuration work together, read Combined Central and Local Configuration.

To configure parameters locally

1. Obtain permission to perform local configuration from a Policy Server administrator.
2. Complete the steps in How to Configure a SOA Agent. These tasks apply to local and central configuration of a Web Agent.
3. In the Agent Configuration Object, set the AllowLocalConfig parameter to yes.
4. Edit the Agent configuration file (WebAgent.conf or JavaAgent.conf depending on the agent type).

   Be sure to modify a copy of the Agent configuration file and maintain a backup copy.

   More information about local configuration and parameter descriptions exists in the *SOA Security Manager Agent Configuration Guide*.

More information:

Combined Central and Local Configuration (see page 72)

Combined Central and Local Configuration

When a SOA Agent is enabled, it searches the Agent Configuration Object for configuration information, and notes the value of the AllowLocalConfig parameter. If this parameter is set to yes, the SOA Agent searches the corresponding Agent's local configuration file for modified or additional parameters, overriding any Agent Configuration Object parameters with the value from its configuration file.

Using the central and local configuration sources, the Agent creates a unified local copy of an Agent Configuration Object that it uses for configuration. The local copy does not alter the Agent Configuration Object that resides at the Policy Server.

Create an Agent Object to Establish a SOA Agent Identity

To create a SOA Agent identity, you must create an Agent object in the Administrative UI. The object name must match the Agent name in the AgentName or DefaultAgentName parameter in the Agent configuration file or Agent Configuration Object. The Policy Server uses the Agent identity to map the Agent name to the IP address of the server hosting the SOA Agent and to associate policies with SOA Agents correctly. Creating a SOA Agent object and identity lets you associate the SOA Agent with a realm.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To create a SOA Agent object and identity

1. Click Infrastructure, Agent, Create Agent.

   The Create Agent pane opens.

2. Verify that Create a new object is selected, and click OK.

   The Create Agent: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Type the name and a description of the Agent in the fields on the General group box.

   Limits:

   - Agent names must contain 7-bit ASCII characters in the range of 32-127, including one or more printable characters.
   - Agent names must not contain the ampersand (&) and asterisk (*) characters.
   - Agent names are not case-sensitive. For example, you cannot create one Agent named MyAgent and another Agent named myagent.

4. Select SOA Security Manager as the Agent Style and Web Agent as the Agent Type on the Attributes group box.
5. Click Submit.

   The Create Agent Task is submitted for processing.

Set the Configuration Parameters in the Agent Configuration Object

The following procedure contains the two general sub-procedures required to set the configuration parameters of an agent configuration object.

To define the SOA Agent's configuration

1. Create an Agent Configuration Object.
2. Modify the configuration parameters in this object.

Note: When configuring centrally or locally configuring a SOA Agent, refer to the *SOA Security Manager Agent Configuration Guide* for parameter descriptions, the default values, and instructions on setting the parameters.

Agent Configuration Object Overview

An Agent Configuration Object holds the parameters that define the SOA Agent configuration. They are the Policy Server's counterpart to Web Agent configuration file.

When you configure a SOA Agent you are prompted for the Agent Configuration Object to associate an this configuration object with a specific SOA Agent.

For more information about the initial configuration of a SOA Agent and the required SOA Agent parameters, see the *SOA Security Manager Implementation Guide*.

Copy an Agent Configuration Object

You can create a new Agent configuration object by copying an existing Agent configuration object. Sample Agent configuration objects are provided for the SOA Agents that SOA Security Manager supports. Once the new Agent configuration object is created, you can modify its list of parameters and their values. For more information about parameters and default settings, see the *SOA Security Manager Agent Configuration Guide*.

To copy an Agent Configuration Object

1. Click Infrastructure, Agent Configuration, Create Agent Configuration.

   The Create Agent Configuration: Search pane opens.

2. Select the Create a copy radio button, and do one of the following:
   - Select one of the default Agent Configuration Objects from those listed.
   - Click Search and choose from the list of Agent Configuration Objects that is displayed.

3. Click OK.

   The Create Agent Configuration: *Name* pane opens.

4. Type a new name and description in the fields on the General group box, and click Submit.

   The Create Agent Configuration Task is submitted for processing.

Create an Agent Configuration Object

The Policy Server installer creates several Agent Configuration Objects that contain default settings. You can duplicate these default objects and use them as templates for your Agent configuration.

To create an Agent Configuration object

- (Recommended) Copy an existing Agent Configuration Object and modify the parameter values.

  Important! Do not directly modify and use a default object. Always copy the object and then modify it.

- Create a new Agent Configuration object.

Note: We recommend creating a new Agent Configuration Object by copying an existing object and modifying its list of parameters and their values. If you create a new Agent Configuration object without copying an existing one, you must enter all of your parameters and their values manually.

To create an Agent Configuration Object

1. Click Infrastructure, Agent Configuration, Create Agent Configuration.

   The Create Agent Configuration: Search pane opens.

2. Verify that Create a new object is selected, and then click OK.

   The Create Agent Configuration: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Type the name and a description of the Agent in the fields on the General group box.

4. Click Add.

   The Create Parameter group box opens.

5. Type the parameter's name and value in the respective fields of the Create Parameter group box.

6. (Optional) Do *one* of the following:
   - To encrypt the value of the parameter, select the Encrypted check box.
   - To create multiple values for one parameter, do the following:
     a. Click the Multi-value button.
     b. Type a value for the parameter in the Values field.
     c. Click Add.
     d. Repeat Steps b and c until you have added all the values you want.
     e. (Optional) To change the sequence of multiple values, click the up and down arrows. Multiple values are separated by the square symbol (☐) when displayed on the Parameters group box. To delete a value, click the minus sign.

     Note: You cannot enter multiple values for encrypted parameters. A single encrypted value is displayed as a string of symbols on the Parameters group box.
7. Click OK.

   The parameter is added to the list on the Parameters group box.

   Note: Parameters are listed in alphabetical order by name. To edit a parameter on the list, click the right arrow. To delete a parameter from the list, click the minus sign. To add more parameters to the Agent configuration object, repeat Steps 4 through 7.
8. Click Submit.

   The Create Agent Configuration Task is submitted for processing.

Required Agent Configuration Object Parameters

When you configure an Agent Configuration Object, the following parameters must be configured:

- All Agents require DefaultAgentName
- SOA Agents for (IIS) Web Servers require DefaultUserName and DefaultPassword
- SOA Agents for (Domino) Web Servers require DominoDefaultUser and DominoSuperUser

Note: More information about these parameters exists in the *SOA Security Manager Agent Configuration Guide*.

Specify the IIS Proxy User

Configuring a SOA Agent for Web Servers to protect IIS requires that you configure the following settings in the Agent Configuration Object:

Note: If you plan to use the NTLM authentication scheme, or enable the Windows User Security Context feature, do not specify values for these IIS parameters.

DefaultUserName

Specifies the username of the IIS Proxy user.

DefaultPassword

Specifies the password of the IIS Proxy user.

The DefaultUserName and DefaultPassword identify an existing NT user account that has sufficient privileges to access resources on an IIS web server protected by SOA Security Manager. When users want to access resources on an IIS Web server protected by SOA Security Manager, they may not have the necessary server access privileges. The SOA Agent must use this NT user account, which is assigned by an NT administrator, to act as a proxy user account for users granted access by SOA Security Manager.

Specify the Domino Users

Configuring a SOA Agent for Web Servers on a Domino server requires that you edit the following settings to match your system:

DominoDefaultUser

If the user is not in the Domino Directory, and they have been authenticated by SOA Security Manager against another user directory, this is the name by which the Domino SOA Agent identifies that user to the Domino server. This value can be encrypted.

DominoSuperUser

Ensures that all users successfully logged into SOA Security Manager will be logged into Domino as the Domino SuperUser. This value can be encrypted.

Specify the Agent Name

All SOA Agents require that you specify a value for the DefaultAgentName.

DefaultAgentName

Identifies the Agent identity that the SOA Agent uses when it detects an IP address on its Web server that does not have an Agent identity assigned to it.

Default: the default Agent name is the name of the installed SOA Agent.

Modify Agent Configuration Parameters

You can edit the names and values of the parameters in an Agent configuration object. Parameter names must be unique. Parameter values can be plain, encrypted, or multi-value. To make an inactive parameter active, remove the pound sign (#).

Note: More information on parameters and default settings exists in the *SOA Security Manager Agent Configuration Guide*.

To modify an Agent configuration object

1. Click Infrastructure, Agent Configuration, Modify Agent Configuration.

   The Modify Agent Configuration: Search pane opens.

2. Specify search criteria, and click Search.

   A list of Agent configuration objects that match the search criteria opens.

3. Select an Agent Configuration object, and click Select.

   The Modify Agent Configuration: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

4. Click the right arrow to the left of the parameter's name on the Parameters group box.

   The Edit Parameter group box opens.

5. Edit the parameter's value.

   Note: To delete a value, click the minus sign. To change the sequence of multiple values, click the up and down arrows. To specify multiple values for one parameter, click Add. Multiple values are separated by the square symbol (☐) when displayed on the Parameters group box.

6. (Optional) Select Encrypted on the Edit Parameter group box.

   Note: When Encrypted is selected, the option of specifying multiple values for one parameter is disabled. A single encrypted value is displayed as a string of symbols on the Parameters group box.

7. Click OK.

   The parameter's valued is updated on the Parameters group box.

   Note: Parameters are listed in alphabetical order by name. To edit a parameter on the list, click the right arrow. To delete a parameter from the list, click the minus sign. To edit multiple parameters for the Agent configuration object, repeat steps 4 through 7.

8. Click Submit.

   The Modify Agent Configuration Task is submitted for processing.

Enable a SOA Agent

Regardless of whether you configure a SOA Agent centrally or locally, you can only enable and disable a SOA Agent from the Agent configuration file.

The EnableWebAgent parameter value determines whether an Agent is enabled or disabled. Set it to yes to enable the Agent. It is set to no by default.

After you have set up all the necessary objects for the Policy Server and Agent to communicate, and you have installed and configured the SOA Agent, then you can enable it.

Note: More information about the Agent configuration file and enabling a SOA Agent exists in the *SOA Security Manager Agent Configuration Guide*.

Agent Groups

An Agent group is a collection of Agents of the same type grouped together for common resource protection. The advantage of Agent Groups is that you can provide access to the same resource to a larger user base because the resource is duplicated on many servers/SOA Agents. It also saves time because you define only one policy for all of the SOA Agents. For example, you can use an Agent group to protect a group of servers that use round robin processing to supply access to the same web service resources.

You can create Agent groups even if the Agents you want to include in a group have not yet been created. Once you add a new Agent in SOA Security Manager, you can edit an Agent group and add the new Agent to the group.

Note: If you configure Agent groups, you cannot directly assign a set of parameter values to an Agent group. You can assign the same Agent Configuration Object to multiple agents and edit the object.

Configure an Agent Group

You can manage Agents that protect a common resource more efficiently by creating an Agent group. All Agents in a group must be of the same type.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure an Agent group

1. Click Infrastructure, Agent Group, Create Agent Group.

   The Create Agent Group pane opens.

2. Verify that Create a new object is selected, and click OK.

   The Create Agent Group: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Type the name and a description of the Agent group in the fields on the General group box.

4. Select SiteMinder as the Agent Style and Web Agent as the Agent Type in the Agent Type group box

5. Click Add/Remove on the Group Members group box.

   The Choose agents pane opens.

6. Select one or more Agents from the list of Available Members, and click the right-facing arrows.

   The Agents are removed from the list of Available Members and added to the list of Selected Members.

   Note: To select more than one member at a time, hold down the Ctrl key while you click on the additional members. To select a block of members, click on the first member and then hold down the Shift key while you click on the last member in the block.

7. Click OK.

   The selected Agents are added to the Agent group.

8. Click Submit.

   The Create Agent Group Task is submitted for processing.

Add Agents to an Agent Group

You can add existing Agents to an Agent group.

To add Agents to an Agent group

1. Click Infrastructure, Agent Group, Modify Agent Group.

   The Modify Agent Group pane opens.

2. Specify search criteria, and click Search.

   A list of Agent groups that match the search criteria opens.

3. Select an Agent group, and click Select.

   The Modify Agent Group: *Name* pane opens.

4. Click Add/Remove on the Group Members group box.

   The Choose agents pane opens.

5. Select one or more Agents from the list of Available Members, and click the right-facing arrows.

   The Agents are removed from the list of Available Members and added to the list of Selected Members.

   Note: To select more than one member at a time, hold down the Ctrl key while you click on the additional members. To select a block of members, click on the first member and then hold down the Shift key while you click on the last member in the block.

6. Click OK.

   The selected Agents are added to the Agent group.

7. Click Submit.

   The Modify Agent Group Task is submitted for processing.

Custom Agents

You create a custom XML-enabled agent using either the C or Java version of the XML Agent Content Helper API included in the SOA Security Manager Software Development Kit.

Note: Custom Agents do not support central agent configuration. More information about using an API to create a custom Agent exists in the SiteMinder *Programming Guide for C* or the SiteMinder *Programming Guide for Java*.

After you develop your own Agent, you must configure a new Agent type. The Agent type defines the behavior of an agent and lets you use the custom Agent to protect resources.

Configure a Custom Agent Type

You configure a custom Agent type to define the behavior of a custom agent and to identify the custom agent when creating the agent object. You create the Agent type using a SiteMinder API.

Note: More information on creating an Agent type using the C version of the SiteMinder Agent API exists in the SiteMinder *Programming Guide for C*.

Create a Custom Agent Object for the Agent Identity

To create a custom Agent identity, you must create an Agent object in the Administrative UI. Creating an Agent object and identity lets you associate the Agent with a realm.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To create an Agent object

1. Click Infrastructure, Agent, Create Agent.

   The Create Agent pane opens.

2. Verify that Create a new object is selected, and click OK.

   The Create Agent: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Type the name and a description of the Agent in the fields on the General group box.

   Limits:

   - Agent names must contain 7-bit ASCII characters in the range of 32-127, including one or more printable characters.
   - Agent names must not contain the ampersand (&) and asterisk (*) characters.
   - Agent names are not case-sensitive. You cannot create one agent named MyAgent and another Agent named myagent.

4. Select SiteMinder as the Agent Style, Web Agent as the Agent Type, and the Supports 4.x agents check box on the Agent Type group box.

   The IP Address and Shared Secret group boxes open.

5. Type the IP Address of the server on which the Agent resides in the field on the IP Address group box.

6. Type and confirm a shared secret in the fields on the Shared Secret group box.

   Limits:

   - The secret that you supply must match the secret that was assigned when the Agent was installed on the Web Server.
   - The secret must contain at least 1 character and not more than 255 characters.
   - The secret must only contain alphanumeric characters.
   - The secret must not contain embedded spaces.

7. Click Submit.

   The Create Agent Task is submitted for processing.

Chapter 5: User Directories

This section contains the following topics:

User Directory Connections Overview (see page 85)
Directory Attributes Overview (see page 99)
How to Configure a CA Directory User Directory Connection (see page 101)
How to Configure a Sun Java System User Directory Connection (see page 105)
How to Configure a IBM Directory Server User Directory Connection (see page 106)
How to Configure a Domino User Directory as a User Store (see page 108)
How to Configure a Novell eDirectory LDAP Directory Connection (see page 110)
How to Configure an ADAM User Directory Connection (see page 115)
How to Configure an Active Directory Directory Connection (see page 118)
How to Configure an Active Directory Global Catalog User Directory Connection (see page 125)
How to Configure an Oracle Internet Directory User Directory Connection (see page 127)
How to Configure an ODBC User Directory Connection (see page 129)
How to Configure a Windows Directory Connection (see page 138)
How to Configure a Custom User Directory Connection (see page 143)
How to Configure an LDAP User Directory Connection over SSL (see page 144)
LDAP Load Balancing and Failover (see page 153)
Configure ODBC Data Source Failover (see page 157)
SQL Query Schemes (see page 158)
Define the Same User Directory Connection in Multiple Policy Stores (see page 166)
View User Directory Contents (see page 167)
Search User Directories (see page 167)
Universal IDs (see page 168)
Named Expressions (see page 169)
User Attribute Mapping (see page 183)

User Directory Connections Overview

User directories and user databases store user data, including organizational information, user and group attributes, and credentials such as passwords. The Administrative UI lets you configure connections to existing user directories and databases.

Directory connections resolve how the Policy Server establishes a context for user identities. The Policy Server uses these connections to verify the user data and credentials obtained from the SOAP envelope or body of a web service request and retrieve user attributes.

You configure the Policy Server to connect to any number of supported user directories, including:

- LDAP
- ODBC
- Oracle
- Windows NT
- Custom

A list of supported directories types exists in the SOA Security Manager platform support matrix.

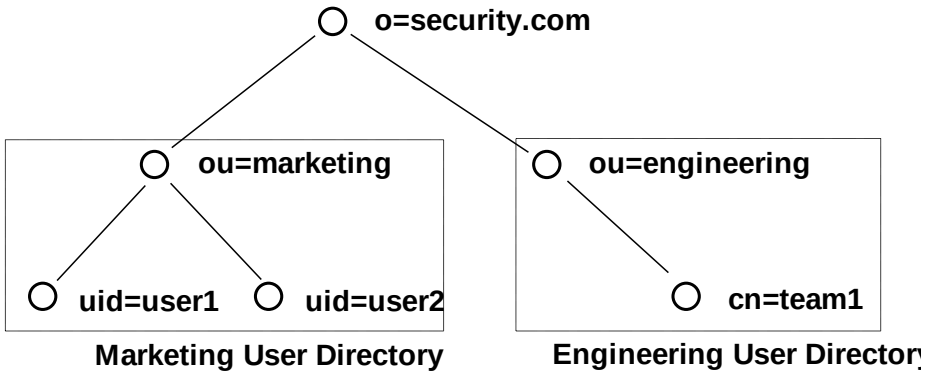
Note: If you are trying to configure or upgrade a SOA Security Manager store listed in the SOA Security Manager Platform Support Matrix and cannot find the procedures in this guide, see the *Directory Configuration Guide*.

LDAP Overview

The Policy Server can communicate with user directories that use the Lightweight Data Access Protocol (LDAP).

General Information About LDAP

LDAP user directories are created with an inverted tree structure. Due to this hierarchical structure, LDAP-enabled directories can contain multiple user namespaces. A namespace is a grouping of entities under a node in the LDAP Directory Information Tree (DIT). Any branch of an LDAP DIT can be defined in a user directory connection as a separate namespace. Typically, user directory connections are configured for DIT branches that represent an organization (o) or an organizational unit (ou).Users and user groups are located under an o= or ou= node in the directory structure.



Any node in an LDAP tree is identified by its distinguished name (DN), which is made up of a comma separated list of its own name and the names of the nodes above it in the directory tree. This method of naming allows each point in the user directory to have a unique DN.

For example, in the diagram above, one of the users in the Marketing department is identified by the following DN:
uid=user1,ou=marketing,o=security.com

The user group Engineering is identified as the following DN:
ou=engineering,o=security.com.

User Disambiguation in an LDAP Directory

User disambiguation is the process of locating a unique user in a user directory. There are two methods of locating users in a user directory. You can locate users by

- DN
- Search expression

The Policy Server uses information you supply in the User Lookup group box of the User Directory pane, and a user-supplied value, such as login name, to locate a user.

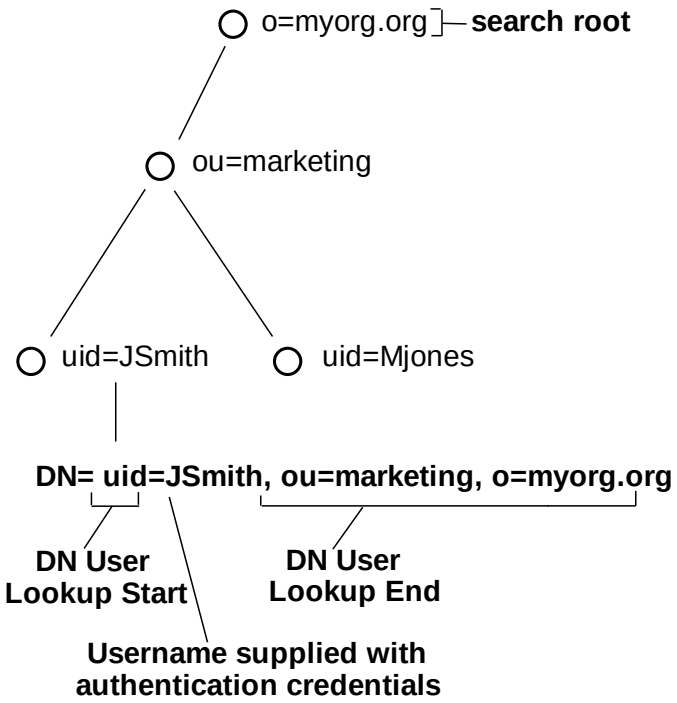
User Lookup by DN

You construct a user lookup by DN from the User Directory pane in the User Lookup group box of the LDAP Settings area. You concatenate the value specified in the User Lookup Start field, the username as specified by the user during login, and the value specified in the User Lookup End field.

The resulting DN has the following format:

<value in the User Lookup Start field>, <username>, <value in the User Lookup End field>

The following illustrates an LDAP Directory Information Tree (DIT) example:



In the previous diagram, the LDAP DIT design requires a DN to be of the form uid=JSmith,ou=marketing,o=myorg.org.

- The User Lookup Start property is uid=
- The User Lookup End property is,ou=marketing,o=myorg.org

Only the unique part, JSmith, must be specified in the credentials when the user logs in.

User Lookup via a Search Expression

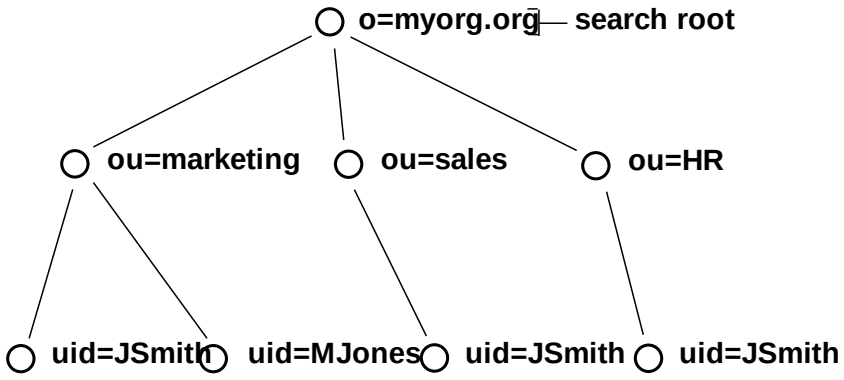
An LDAP directory server may contain numerous users in complicated DITs, and it may not be practical to create a large number of user directory connections. Your organization may have hundreds of organizational units and you may want to avoid having end users log in with detailed string representations. Instead, one user directory connection pointing to a common root can be created with the User DN Lookup Start and User DN Lookup End properties defining an LDAP search expression. The result of the search expression is a list of user DNs for the Policy Server to try during authentication.

Example: Search expressions for user DN lookups

To locate a user across many organizational units, define the User Lookup Start property as (&(objectclass=inetOrgPerson)(uid= and define the User Lookup End property as)). Only the unique part, the uid value, must be specified in the credentials. In the section of the User Directory Dialog shown above, these values replace the values contained in the LDAP User DN Lookup group box.

Note: An InetOrgPerson is a common object class used in LDAP directory deployments.

See the following figure for the type of LDAP DIT where invoking a search expression is useful:



User Name	Distinguished Name
JSmith	uid=JSmith,ou=marketing,o=myorg.org
JSmith	uid=JSmith,ou=sales,o=myorg.org
JSmith	uid=JSmith,ou=HR,o=myorg.org

In this case, if JSmith from ou=sales wants to access a resource, JSmith can authenticate using only his or her name for credentials (as opposed to an entire DN string). By placing the uid= attribute between the User DN Lookup Start and User DN Lookup End fields with the search expressions in the corresponding fields, the Policy Server will find all DNs that match the LDAP query (&(objectclass=inetOrgPerson)(uid=JSmith)).

The Policy Server then has a list of DNs to choose from in giving access to the protected resource. Assuming the resource can only be accessed by the JSmith of ou=sales, the username/password for the DN uid=JSmith,ou=sales,o=myorg.org will be the one that is authenticated.

LDAP Search Filters

As you work with LDAP directory connections in the Policy Server, you may need to specify filters for LDAP search expressions. The following table provides a brief description of some common LDAP search filters.

Search Filter	Format	Description
Equality	attribute=value For example, to find a user whose user ID is jsmith, the search filter is uid=jsmith.	This filter finds a specific value for an attribute in an LDAP directory.
String Matching	attribute=*value, OR attribute=value*, OR attribute=val*ue, OR attribute=*value* For example, uid=*smith matches all values that end in smith, such as jsmith, msmith, etc. A value of uid=*smith* matches jsmith, msmith, and bsmithe.	LDAP search filters support wild cards, which allow you to search for an attribute value based on a partial string. To find all of the values that match a partial string, use the wildcard character (*).
Greater than or equal to	attribute>=value For example, to find all of the users in a directory who are age 21 or over, part of the search filter would be age>=21.	This filter finds values that are greater than or equal to the specified value.
Less than or equal to	attribute<=value For example, to find all of the users in a directory who are age 21 or younger, part of the search filter would be age<=21.	This filter finds values that are less than or equal to the specified value.

Search Filter	Format	Description
Greater than	(!(attribute<=value)) For example, to find all of the users in a directory who are older than 21, part of the search filter would be (!(age<=21)).	LDAP does not support greater than expressions. In order to filter LDAP attribute values by greater than, you must use the Negation operator (!) in conjunction with a greater than or equal to expression.
Less than	(!(attribute>=value)) For example, to find all of the users in a directory whose age is less than 21, part of the search filter would be (!(age>=21)).	LDAP does not support less than expressions. In order to filter LDAP attribute values by less than, you must use the Negation operator (!) in conjunction with less than or equal to expression.
Approximate	attribute~=value For example, the filter uid~=smith may return the values smithe and smitt.	This filter returns values that are similar to the value specified in the filter.
Presence	attribute=* For example, email=* returns all users who have an email address.	This filter determines if an attribute is present.
Complex filters: And (&) Or (|) Not (!)	Intersection of Filter1 and Filter2: (&(filter1)(filter2)) Union of Filter1 and Filter2: (|(filter1)(filter2)) Satisfies Filter1, but not Filter2: (&(filter1)(!(filter2)) Note You must use parentheses to enclose the complex filter and each filter in the complex filter. For example, if you want to find all users whose User ID begins with the letter s and who are over 21 years old, you could use a filter of (&(uid=s*)(!(age<=21))).	Creates complex search filters.

Objectclass Searches

Each entry in an LDAP table has at least one objectclass attribute. You can use a presence filter in conjunction with the objectclass attribute to build filters for searching your LDAP user directories. In SOA Security Manager environments, the objectclass attribute is most useful in the following cases:

- List all entries directly below an entry

  To retrieve all entries one level below a directory entry, specify a search scope of One Level, and use a search filter of (objectclass=*). Since all LDAP directory entries have at least one objectclass attribute, the search filter returns a complete list of the entries below the root.

- List all entries in a subtree

  To retrieve all entries in the branches below a directory entry, specify a search scope of Subtree, and use a search filter of (objectclass=*). The search filter returns a complete list of the entries in the entire subtree.

Filtered Characters in User IDs

SOA Security Manager provides LDAP search filter checking functionality that parses LDAP search filters to ensure that they comply with the LDAP standard (RFC).

All user login IDs are filtered for "(", ")", "\" characters by default before being checked against an LDAP user store. To disable this check, set the following EnableSearchFilterCheck registry value to 0:

HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\Siteminder\Ds\LDAPProvider\EnableSearchFilterCheck

Important! By disabling this check, you may expose your system to attack, and should not allow user IDs using these characters.

LDAP Referrals

The Policy Server supports LDAP referrals for LDAP user directory connections. An LDAP referral in one directory points to a location in another LDAP directory. There are two types of LDAP referrals: write referrals and read referrals. In addition, the Policy Server supports enhanced referral processing.

Note: In order for LDAP referrals to work correctly in a multiple policy store environment, the SOA Security Manager LDAP policy store schema must be applied to all replicas. For more information see the section describing LDAP policy store installation in the *Policy Server Installation Guide*.

Write Referrals

In a directory deployment that includes master and slave LDAP directories, LDAP write referrals allow updates to a master directory that can then be replicated to slave directories. In a SOA Security Manager deployment, you can specify a connection to a slave LDAP directory. If you use any of SOA Security Manager's features that require data to be written to the LDAP directory, SOA Security Manager automatically detects referrals that point to a master LDAP directory. The information that SOA Security Manager writes to the LDAP directory will be stored in the master LDAP directory and replicated to the slave LDAP directory according to the replication scheme of your network resources.

Read Referrals

In a large LDAP directory deployment, information may be divided among several LDAP directories. For example, one directory may contain enough user information to authenticate a user, while another directory may contain other important user attributes. The authentication directory can be configured to point to the directory containing user attributes. This process is called a read referral. If a directory connection exists for an LDAP directory that contains read referrals, SOA Security Manager is able to use the read referrals to retrieve information from the associated directories.

More information:

Enhanced LDAP Referral Handling (see page 95)

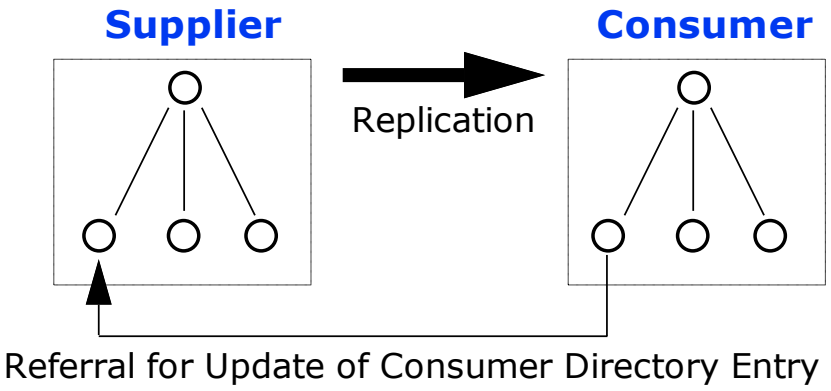
Directory Topology and LDAP Referrals

An LDAP directory's topology describes the division of a directory tree among physical servers. The logical sections of a directory tree are called partitions. LDAP directory topology varies widely between LDAP deployments, but regardless of the topology, the use of referrals between partitions allows the directory to function as a single service.

Three types of LDAP referrals can be employed in a directory topology. These types can be used in conjunction to create very complex directory structures. The types are:

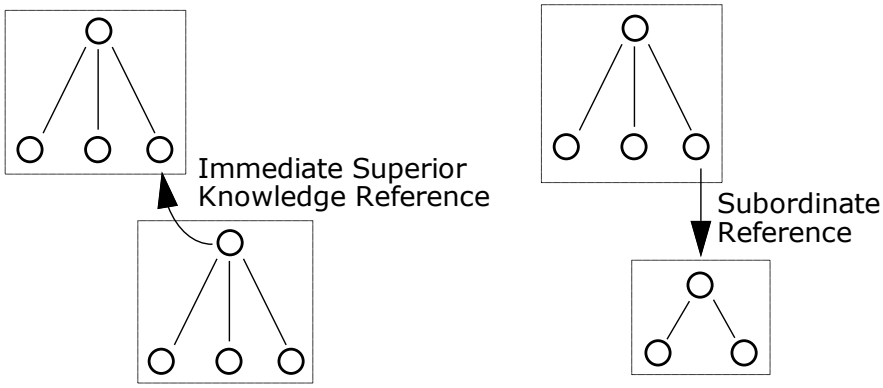
Replication Agreements

When a directory topology includes a replication agreement, all changes in a supplier directory are replicated (duplicated) in a second consumer directory. The consumer and supplier directories may be used to load balance requests, or may have a failover relationship. When an update request is received by the consumer directory, the consumer directory refers the request to the supplier directory where the update is completes. This is a very common type of LDAP referral.



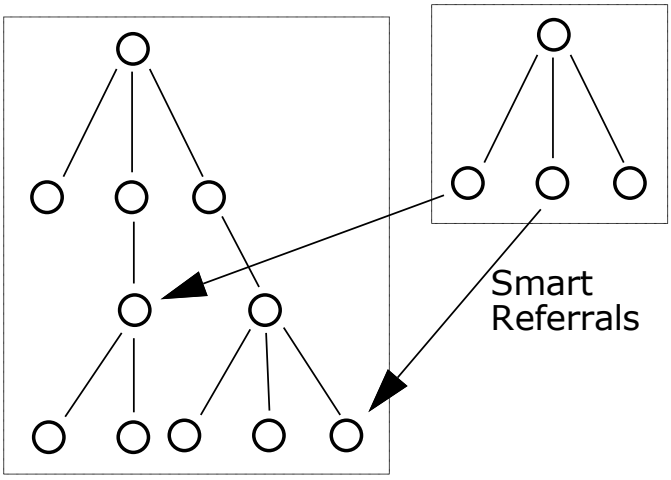
Knowledge References

Pointers from one directory partition to another are called knowledge references. Knowledge references that point to the node immediately upward toward the root in the DIT are considered *immediate superior knowledge references*. Knowledge references that point downward in the DIT to other partitions are considered *subordinate references*.



Smart Referrals

A pointer to a location in a portion of the directory that is not immediately above or below the original partition s called a *smart referral*. A smart referral contains enough information to see a node anywhere in the directory topology.



Enhanced LDAP Referral Handling

Enhancements have been made to the Policy Server's LDAP referral handling to improve performance and redundancy. Previous versions of the Policy Server supported automatic LDAP referral handling through the LDAP SDK layer. When an LDAP referral occurred, the LDAP SDK layer handled the execution of the request on the referred server without any interaction with the Policy Server.

SOA Security Manager includes support for non-automatic (enhanced) LDAP referral handling. With non-automatic referral handling, an LDAP referral is returned to the Policy Server rather than the LDAP SDK layer. The referral contains all of the information necessary to process the referral. The Policy Server can detect whether the LDAP directory specified in the referral is operational, and can terminate a request if the appropriate LDAP directory is not functioning. This feature addresses performance issues that arise when an LDAP referral to an offline system causes a constant increase in request latency. Such an increase can cause SOA Security Manager to become saturated with requests.

For example, a SOA Security Manager deployment includes two LDAP directories that are deployed in a supplier/consumer replication scheme with failover. All requests are made to the consumer directory. If the consumer directory is unavailable, the Policy Server uses the failover configuration to query the supplier directory.

If Password Services is enabled, an LDAP referral in the consumer directory takes place to ensure that password changes are written in the supplier directory. In previous versions of the Policy Server which only supported automatic LDAP referrals, if the supplier directory was unavailable, the Policy Server was unaware that the referral from the consumer directory required to write new password information into the supplier directory was not functioning, and would wait for a response from the supplier. This delay could cause the system to become saturated by pending requests.

If you configure your Policy Server with enhanced LDAP referral handling, the Policy Server is aware of the unavailable supplier LDAP directory and terminates requests that require password changes automatically, until the supplier directory is available to record password changes.

For information about configuring enhanced LDAP referral handling, see the *Policy Server Management* guide.

How the Policy Server Binds to an LDAP User Store

The Policy Server opens three connections when connecting to an LDAP user store:

- The first connection verifies that the user store is up and running. By default, the Policy Server pings the user store every 30 seconds on this connection.
- The second connection is used for searches and updates. For example, the Policy Server uses this connection for user lookup and setting attributes on bind failures.
- The third connection is used for testing credentials. The Policy Server attempts to bind to the user store using the user's credentials. The result of the bind attempt identifies if the user's credentials are accepted or rejected.

ODBC Database Overview

SOA Security Manager can use a proprietary schema in an ODBC-compatible database as a user directory for authentication and authorization purposes. This option is useful at sites where user information, such as a user name, password, and group membership is stored in an ODBC database. At such sites, this feature allows the Policy Server to view a proprietary database schema as a user directory.

The Policy Server supports connections to the following types of ODBC-compatible databases:

- Microsoft SQL Server—Windows Policy Servers only
- Oracle RDBMS

Note: If your user directory data is stored in an Oracle database, we recommend that you use OCI, instead of ODBC, to connect to that user directory.

For the most current information about supported database versions, check the CA Support Site.

To configure the Policy Server to use a database as a user directory you must:

- Type the SQL query information in the fields on the SOA Security Manager SQL Query Scheme pane. This pane is accessible from the SOA Security Manager User Directory pane. It contains fields for mapping information required by the Policy Server to fields in the ODBC-compatible database.

  Note: Don't use the same data source with different query schemes. Create a unique data source for each query scheme.

- Define an ODBC data source that points to the database containing user information.

  Note: Configure the ODBC data source as a System DSN. The data source must point to a Microsoft SQL Server database or an Oracle database. For information on configuring an ODBC data source, see your Windows operating system documentation.

More information:

Configure a SQL Query Scheme (see page 158)

Windows Directory Overview

User directory connections can be set up with any of the following WinNT implementations:

- WinNT domain
- Individual WinNT computer in a domain
- Stand-alone WinNT computer

In order for the Policy Server to connect to your WinNT domain, it must meet the following requirements:

- A one way trust relationship between the domain containing the Policy Server, and the domain containing users, must exist.
- Every user in the domain that will be authenticated by SOA Security Manager needs the Access the computer from network user right for the machine where the policy server is running.
- The account that SOA Security Manager uses to access the User Directory object needs to have access to the IPC$ share on the domain controller machine where the WinNT domain users are located.

  Note: These requirements may be met by default if you use the default configuration for your WinNT domain. Your WinNT domain administrator should verify that the domain meets the above requirements.

The Policy Server authenticates against WinNT and can authorize users based on their individual identities and group membership.

When authenticating against a WinNT namespace, the Policy Server passes user credentials to WinNT for authentication. The credentials are the user's WinNT user name and password. In a SOA Security Manager environment, where multiple WinNT namespaces are defined, user authentication is faster if the user name supplied to SOA Security Manager includes the domain name (i.e. *domain\username)*. In that case, SOA Security Manager skips all WinNT namespaces that do not match the specified domain name.

WinNT user names and passwords can be used as credentials.

Note: To authenticate users against a WinNT domain, the Policy Server must run on WinNT.

Active Directory Overview

SOA Security Manager supports user directories on the Microsoft Active Directory platform. Although the configuration for Active Directory (AD) and LDAP namespaces in the Administrative UI is very similar, there are several functional differences.

The advantages of using the AD namespace when configuring an Active Directory user store include:

- SSL connectivity using a native Windows certificate database.

  Note: Both the Policy Server and the systems hosting Active Directory user stores must have an established trust. For information about configuring Windows systems and Active Directory for SSL, see your Windows documentation.

- Support for native Windows SASL which allows for secure LDAP bind operations.

The disadvantages include:

- No support for enhanced LDAP referrals.
- No support for LDAP paging and sorting operations.

Note: For information about using the LDAP namespace for an Active Directory user store, see Configure Active Directory Connections (see page 123).

Custom Directory Overview

The Administrative UI allows you to create custom user directory connections by creating shared libraries using the SOA Security Manager Directory API (available separately with the Software Development Kit; if installed, see the *API Reference Guide for C* for more information). Custom connections allow the Policy Server to interact with legacy directories. Once you create a directory connection using the SOA Security Manager APIs, you can configure a custom namespace on the User Directory pane.

Directory Attributes Overview

Some SOA Security Manager features require read or read/write access to seven SOA Security Manager attributes whose values are stored in the user directories connected to the Policy Server. When you configure a connection from the Policy Server to a user directory, you must specify the names of the user attributes in that user directory that correspond to the seven SOA Security Manager attributes. This is done in the fields on the User Attributes group box.

For example, the name of the Universal ID might be Student ID in one user directory, while in another directory, the name of the Universal ID might be Account Number. Once the directory connections are configured, SOA Security Manager can access the correct user attribute in the selected user directory each time that it encounters the Universal ID.

You can extend this capability of SOA Security Manager through user attribute mapping. User attribute mapping allows you to define your own common names, mapping each one to user attribute names in multiple user directories with different underlying schema.

Each SOA Security Manager attribute is associated with a data type and one or more directory types and is described in the following table. (R) indicates that the attribute requires read access. (RW) indicates that the attribute requires read/write access.

Attribute Name	Data Type	Directory Types	Description
Universal ID (R)	string	LDAP Database WinNT	Specifies the universal ID or user identifier that SOA Security Manager passes to protected applications to maintain a user's identity. This feature is a bridge between SOA Security Manager and legacy applications, which often use attributes to identify a user. The universal ID is also used in configuring Directory mapping.
Disabled Flag (RW)	string	LDAP Database	Specifies the user's account status. More information exists in the *Policy Server Administration Guide*.
Password Attribute (RW)	binary	LDAP Database	Specifies the user's password.
Password Data (RW)	binary	LDAP Database	Is used to track password policy information.
Anonymous ID (RW)	string	LDAP Database	Stores the DN of users who are authenticated using an anonymous authentication scheme.
Email (R)	string	LDAP Database	This attribute is not currently used by a SOA Security Manager feature.
Challenge/Response (RW)	string	LDAP	Specifies the question and answer pair that is used by the Forgotten Password feature in Password Services and DMS. The Challenge string is the password hint that is passed to the

Attribute Name	Data Type	Directory Types	Description
			user.

Note: When configuring a user directory connection, you can specify the administrator credentials that the Policy Server uses to access the directory. These credentials must have the same read/write access as the corresponding user attributes in the table.

More information:

Universal IDs (see page 168)
User Attribute Mapping (see page 183)

How to Configure a CA Directory User Directory Connection

You can use a CA Directory user directory as a user store. The following process lists the steps for creating the user store connection to the Policy Server:

1. Ping the User Store System
2. Configure a CA Directory User Directory Connection
3. (Optional) Enable Caching for a CA Directory User Store
4. (Optional) Verify the CA Directory Cache Configuration

Ping the User Store System

Pinging the user store system verifies that a network connection exists between the Policy Server and the user directory or database.

Note: Some user store systems may require the Policy Server to present credentials.

Configure CA Directory User Directory Connections

You can configure a user directory connection that lets the Policy Server communicate with a CA Directory user store.

To configure the user directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Select LDAP from the Namespace list.

   LDAP settings open.
4. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.
5. Type the LDAP Search and LDAP User DN Lookup settings in the fields on the LDAP Settings group box.
6. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.
7. (Optional) Specify the user directory profile attributes that are reserved for SOA Security Manager's use in the fields on the User Attributes group box.
8. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
9. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 153)
Define an Attribute Mapping (see page 186)

Enable User Store DSA Parameters

SOA Security Manager uses the Sun Java System LDAP SDK, which lets clients open one managed connection to the directory server and perform user binds under that connection. If you are using CA directory as a user store, the Policy Server connects to CA Directory by performing a bind request for each authentication request. Configure CA Directory to handle these requests, or CA Directory runs out of connections and authentication fails.

To enable user store DSA parameters

1. Open the .dxc file for the user store DSA.
2. Define the following at the bottom of the file:

```
#SiteMinder
set mimic-netscape-for-siteminder = true;
set concurrent-bind-user = DN;
set hold-ldap-connections = true;
```

3. Save and close the .dxc file.

   The user store DSA parameters are enabled.

   Note: The DN is in x500 format.

   Example: <o acme><cn smadmin>

Enable Caching for a CA Directory User Store

You can improve SOA Security Manager authentication and authorization performance for large user stores by enabling the CA Directory DXcache feature. A 5 MB user store is considered large.

To enable caching

1. As the dsa user, edit the user store DSA's DXI file (for example, *<dxserver_install>*\config\servers\eTrustDsa.dxi) and add the following lines to the end of the file:

```
# cache configuration
set max-cache-size = 100;
set cache-index = commonName, surname, objectClass;
set cache-attrs = all-attributes;
set cache-load-all = true;
set lookup-cache = true;
```

   Note: The max-cache-size entry is the total cache size in MB. Adjust this value based on the total memory available on the CA Directory server and overall size of the user store. In addition, set the cache-index fields to those fields used by SOA Security Manager to perform a user search in the user store. For example, if users are authenticated and authorized based on their common name (cn=*), make sure that the commonName is set in the cache-index.

2. As the dsa user, stop and restart the user DSA to allow the DXcache configuration changes to take effect:

```
dxserver stop eTrustDsa
dxserver start eTrustDsa
```

Verify the CA Directory Cache Configuration

After configuring the CA DXcache feature for the user store, you can verify that the cache is enabled using the DXmanager user interface.

To verify the cache

1. Using a Web browser, connect to the CA DXmanager Web interface.

   For example:

   http://*<CA_host>*:8080/dxmanager/ManagerServlet?hostgroup=All

2. Navigate to the DSA configuration page and verify that the DXcache Status field is set to Enabled for your policy store DSA.

How to Configure a Sun Java System User Directory Connection

You can use a Sun Java System user directory as a user store. The following process lists the steps for creating the user store connection to the Policy Server:

1. Ping the User Store System
2. Configure the Sun Java User Directory Connection

Ping the User Store System

Pinging the user store system verifies that a network connection exists between the Policy Server and the user directory or database.

Note: Some user store systems may require the Policy Server to present credentials.

Configure Sun Java System User Directory Connections

You can configure a user directory connection that lets the Policy Server communicate with a Sun Java System user store.

Sun Microsystems recommends using an administrator account other than cn=Directory Manager. Using cn=Directory Manager may cause performance issues due to security policies applied to this account. Instead, create a new user with sufficient privileges to manage the directory and specify that user in the Username field.

To configure the user directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Select LDAP from the Namespace list.

   LDAP settings open.

4. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.

5. Type the LDAP Search and LDAP User DN Lookup settings in the fields on the LDAP Settings group box.
6. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.
7. (Optional) Specify the user directory profile attributes that are reserved for SOA Security Manager's use in the fields on the User Attributes group box.
8. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.

9. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 153)
Define an Attribute Mapping (see page 186)
How to Configure an LDAP User Directory Connection over SSL (see page 144)

How to Configure a IBM Directory Server User Directory Connection

You can use an IBM Directory Server as a user store. The following process lists the steps for creating the user store connection to the Policy Server:

1. Ping the User Store System
2. Configure an IBM Directory Server User Directory Connection

Ping the User Store System

Pinging the user store system verifies that a network connection exists between the Policy Server and the user directory or database.

Note: Some user store systems may require the Policy Server to present credentials.

Configure IBM Directory Server User Directory Connections

You can configure a user directory connection that lets the Policy Server communicate with an IBM Directory Server user store.

To configure the user directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Select LDAP from the Namespace list.

   LDAP settings open.
4. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.
5. Type the LDAP Search and LDAP User DN Lookup settings in the fields on the LDAP Settings group box.
6. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.
7. (Optional) Specify the user directory profile attributes that are reserved for SOA Security Manager's use in the fields on the User Attributes group box.
8. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
9. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 153)
Define an Attribute Mapping (see page 186)
How to Configure an LDAP User Directory Connection over SSL (see page 144)

How to Configure a Domino User Directory as a User Store

Configuring a Domino user directory as a user store is a three-step process:

1. Verify that a Domino User Directory Meets Policy Server Requirements
2. Ping the User Store System
3. Configure a Connection from the Policy Server to a Domino User Store

Verify that a Domino User Directory Meets Policy Server Requirements

A Domino user directory is an LDAP directory. Before configuring a Domino user directory as a user store, verify that it meets the following Policy Server requirements:

- The Domino user directory must be version 5.02 or greater.
- The Domino user groups must share the root DN that you specify when configuring the connection from the Policy Server to the Domino user store.

  Example: When adding the group *marketing/myorg.org* to the address book (names.nsf) in Lotus Notes, you must type *o=myorg.org* in the Root field on the User Directory pane.
- Each new user must have both a user name entry and an Internet password entry in the Domino user directory.

  Note: We recommend that you register users when you add them to a Domino user directory. This additional step prevents multiple user name entries in the Domino user directory. When there are multiple entries in the directory, the Policy Server uses the first one. Attempts to log in with other user names fail.

Ping the User Store System

Pinging the user store system verifies that a network connection exists between the Policy Server and the user directory or database.

Note: Some user store systems may require the Policy Server to present credentials.

Configure Domino Directory Connections

You can configure a user directory connection that lets the Policy Server communicate with a Domino user store.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure the user directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Select LDAP from the Namespace list.

   LDAP settings open.
4. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.
5. Type the LDAP Search and LDAP User DN Lookup settings in the fields on the LDAP Settings group box.

   Note: The value that you specify in the Root field must match the organization name that you assigned in Lotus Notes. The Root must also include a country, if you specified a country in Lotus Notes.

   Example: You have an organization called "myorg", which is located in the United States. The Search Root is specified as o=myorg,c=us.

   Note: The search strings that you specify in the User DN Lookup Start and End fields must adhere to proper LDAP notation, not the Lotus Notes shorthand notation. More information about search strings exists in LDAP Search Filters.
6. (Optional) Click Configure to configure load balancing and failover.

   Note: More information on load balancing and failover exists in LDAP Load Balancing and Failover.
7. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.

8. (Optional) Specify the user directory profile attributes that are reserved for SOA Security Manager's use in the fields on the User Attributes group box.
9. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
10. Click Submit.

    The Create User Directory task is submitted for processing.

More information:

User Disambiguation in an LDAP Directory (see page 87)
LDAP Load Balancing and Failover (see page 153)
Directory Attributes Overview (see page 99)
Define an Attribute Mapping (see page 186)
How to Configure an LDAP User Directory Connection over SSL (see page 144)

How to Configure a Novell eDirectory LDAP Directory Connection

You can use a Novell eDirectory LDAP user directory as a user store. The following process lists the steps for creating the user store connection to the Policy Server:

1. Configure NetWare
2. Configure Anonymous LDAP Access on Novell eDirectory

   or

   Create access for a specific SOA Security Manager Administrator:

   - Special Access for the SOA Security Manager Administrator
   - Create a Novell eDirectory User Account for SOA Security Manager Administration
3. Ping the User Store System
4. Configure a Novell eDirectory LDAP Directory Connection

Configure NetWare

The goal of the configuration described in this section is to allow the Policy Server to log into the Novell eDirectory, view the contents of the directory, and retrieve directory attributes. For some advanced features of SOA Security Manager, you may also need to configure the Novell eDirectory to allow the Policy Server write-access to the directory.

If you installed LDAP as part of your Novell eDirectory installation, you should have a server in Novell eDirectory called LDAP Server and an LDAP group named LDAP Group. LDAP Server should be a member of the LDAP Group.

To create the LDAP Server and LDAP Group in Novell eDirectory

1. Create an LDAP Server in Novell eDirectory. (For this example, it is called LDAP Server.)
2. Create an LDAP Group in Novell eDirectory. (For this example, it is called LDAP Group.)
3. Assign LDAP Group to LDAP Server.
   a. In the NW Admin tool, right click on LDAP Server.

      Note: If you are using the Netware ConsoleOne tool instead of the NW Admin tool to modify your Novell eDirectory, you must complete the same tasks using the tools available in ConsoleOne. The interface for the two tools is similar. See your Novell documentation for more information.

   b. From the popup menu, select Details.
   c. Type LDAP Group in the LDAP Group field.
   d. Click OK.

More information:

Directory Attributes Overview (see page 99)

Configure Anonymous LDAP Access on Novell eDirectory

In order for the Policy Server to interact with an Novell eDirectory, you must create an account with enough administrative privileges to allow access to the directory.

The easiest configuration is to generate an anonymous user on the LDAP server and make this the proxy user. The user should be assigned enough power to perform all functions necessary for SOA Security Manager on the LDAP server.

The instructions below assign administrator privileges to an anonymous user, although you can configure the user with more limited privileges. The effect of this is that any anonymous access to the LDAP directory will gain the same privileges you give to SOA Security Manager.

To configure anonymous LDAP access

1. Create a user called LDAP_Anonymous.

   The following procedure is an example which may differ based on your version of Novell products.

   a. From the menu bar of the NW Admin tool, select Object, Create, User.

   b. Add the name LDAP_Anonymous.

   c. Do not assign a password.

   d. In the right frame, select Security Equal To and add the admin user (for example, Admin.transpolar).

   e. Click OK.

2. Set up a proxy account:

   The following procedure is an example which may differ based on your version of Novell products.

   a. In the NW Admin tool, select LDAP Group.

   b. From the popup menu, select Details.

   c. Click Continue.

   d. In Proxy Username field, enter LDAP_Anonymous.

   e. In right frame, select Access Control and click Add.

   f. In the LDAP ACL Name field, enter LDAP_Anonymous.

   g. Select the LDAP Distinguished Name check box and enter cn=LDAP_Anonymous.

   h. Select the All Attributes and Object Rights check box.

   i. Click OK.

   j. In right frame, select Access Control and click Add.

   k. In box labeled LDAP ACL Name, enter Everyone.

   l. Select the Everything check box.

   m. Select the All Attributes and Object Rights check box.

   n. Click OK.

   o. Click OK.

      To continue configuring your Novell eDirectory for use with the Policy Server, see Configure a Novell eDirectory LDAP Connection in Policy Server User Interface (see page 114).

Special Access for the SiteMinder Administrator

The alternate instructions below allow special access only to the Policy Servers and may be more appropriate in some environments.

1. Create an Novell eDirectory user to represent the SOA Security Manager administrator (for example called SOA Security Manager_admin).
2. Give this user a password generated by the SOA Security Manager administrator which will be entered in the Administrative UI.

Create a Novell eDirectory User Account for SiteMinder Administration

You can give the SOA Security Manager Administration a user account using the NW Admin tool.

To create a Novell eDirectory user account for SOA Security Manager administration

1. In the NW Admin tool, right click LDAP Group.
2. From the popup menu, select Details.
3. In the right panel, click Access Control.
4. Add an ACL.
5. Enter a name for the ACL.
6. In the Access By List screen, click Add.
7. In the Access By List panel, click LDAP Distinguished Name.
8. Enter cn=SOA Security Manager_admin.

By default, set the access level to Read, which is sufficient for SOA Security Manager's basic functions. Customers whose use active APIs or some of SOA Security Manager's advanced features (for example, Password Services, User Disablement, Registration Services) may require Write access.

Ping the User Store System

Pinging the user store system verifies that a network connection exists between the Policy Server and the user directory or database.

Note: Some user store systems may require the Policy Server to present credentials.

Configure Novell eDirectory LDAP Directory Connections

You can configure a user directory connection that lets the Policy Server communicate with a Novell eDirectory user store.

To configure the user directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Select LDAP from the Namespace list.

   LDAP settings open.
4. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.
5. Type the LDAP Search and LDAP User DN Lookup settings in the fields on the LDAP Settings group box.

   Note: If the user directory contains multiple organizations, you can leave the Root field blank. This lets the Policy Server search for users in multiple organizations.
6. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.
7. (Optional) Specify the user directory profile attributes that are reserved for SOA Security Manager's use in the fields on the User Attributes group box.
8. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
9. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 153)
Define an Attribute Mapping (see page 186)
How to Configure an LDAP User Directory Connection over SSL (see page 144)

How to Configure an ADAM User Directory Connection

You can use an ADAM user directory as a user store. The following process lists the steps for creating the user store connection to the Policy Server:

1. Create an ADAM Instance and User Store Connection
2. Create a SOA Security Manager User for Connecting to the ADAM User Store
3. Assign Administrator Privileges to the SOA Security Manager User
4. Load Users into the ADAM User Store.
5. Configure the ADAM User Store Directory Connection

Create an ADAM Instance and User Store Connection

You use the Connection Settings dialog of ADAM ADSI Edit to create an ADAM instance and user store connection.

To create an ADAM instance and user store connection

1. Install an ADAM instance on a free port number. This example uses port 390.
2. Open the ADAM ADSI utility by going to Start, Program Files, ADAM, ADAM ADSI Edit.
3. Right-click ADAM ADSI Edit and select Connect to, which opens the Connections Setting dialog.
4. In the Connections Setting dialog:
   a. Enter a name in the Connection name field. This example uses My Connection.
   b. Select localhost in the Server name field.
   c. Enter a free port number in the Port field. This example uses port 390.
   d. Select Distinguished name (DN) or naming context and enter O=userstore in the field.
   e. Click OK.

   The connection (MyConnection [localhost:390] root element) you configured appears under ADAM ADSI Edit.

Create a SiteMinder User For Connecting to the ADAM User Store

You can create a SOA Security Manager user in the Connection Settings dialog shown in the previous topic.

To create a SOA Security Manager user for connecting to the ADAM user store

1. Under o=userstore, right-click CN=Roles and select New, Object.
2. In the Create Object dialog:
   a. Select user and click Next.
   b. In the Value field, enter a user name that SOA Security Manager uses to connect to the ADAM user store and click Next. This example uses admin.
   c. Click Finish.
3. In the right column of ADAM ADSI Edit, right-click CN=admin and select Reset Password.
4. In the Reset Password dialog, enter and confirm the new password and click OK.

The SOA Security Manager user exists in ADAM and the next step is to give this user administrator privileges.

Assign Administrator Privileges to the SiteMinder User

You use ADAM ADSI Edit to assign administrator privileges to SOA Security Manager users.

To assign administrator privileges to the SOA Security Manager user

1. In the right column of ADAM ADSI Edit, right-click CN=Administrators and select Properties.
2. In the CN=Administrators Properties dialog, scroll down to the member attribute and click Edit.
3. Click Add ADAM Account.
4. In the Add ADAM Account dialog, add the SOA Security Manager user you created in Create a SOA Security Manager User For Connecting to the ADAM User Store.
5. Click OK.

   This example uses CN=admin,CN=Role,O=userstore.

Load Users into the ADAM User Store

You use the ADAM Tools Command Prompt to load users into the ADAM user store.

To load users into the ADAM user store

1. Start the ADAM Tools Command Prompt by going to Start, Programs, ADAM, ADAM Tools Command Prompt.
2. Load the .ldif file that has the users for the store by running the following command:

   ldifde -i -f *user_store*.ldif -s *IP_address* -t *port_number*

   user store

   Specifies the name of your user store.

   IP address

   Specifies the IP address of the machine that is hosting ADAM.

   port number

   Specifies the port number on which ADAM is listening.

Configure ADAM User Store Directory Connections

You can configure a user directory connection that lets the Policy Server communicate with an Active Directory Global Catalog user store.

To configure the user directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Select LDAP from the Namespace list.

   LDAP settings open.

4. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.

5. Type the LDAP Search and LDAP User DN Lookup settings in the fields on the LDAP Settings group box.

6. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.
7. (Optional) Specify the user directory profile attributes that are reserved for SOA Security Manager's use in the fields on the User Attributes group box.
8. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
9. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 153)
Define an Attribute Mapping (see page 186)
How to Configure an LDAP User Directory Connection over SSL (see page 144)

How to Configure an Active Directory Directory Connection

The following process lists the steps for creating the user store connection to the Policy Server:

1. Verify that an Active Directory User Directory Meets Policy Server Requirements
2. Specify an Active Directory or LDAP Namespace
3. Ping the User Store System
4. Configure a Connection from the Policy Server to an Active Directory User Store

Active Directory Considerations

Before you configure a connection to an Active Directory consider the following:

Windows deployments

SOA Security Manager establishes the Windows user context by passing the user's fully qualified Windows ID and password to IIS. SOA Security Manager obtains the fully qualified Windows ID from the user's DN entry by concatenating the first cn and dc values found in the DN. For example, if the user DN is:

cn=<username>,cn=<usergroup>,dc=<server>,dc=<domain>,
dc=<extension>

The resulting Windows ID is <server>\<username>. IIS requires the <username> to be the same as the Windows user ID and the <server> to be the log-on domain name.

Multi-byte Character Support

The AD namespace does not support multi-byte character sets. To use a multi-byte character set with Active Directory, configure your directory connection using the LDAP namespace.

Note: Regardless of the code page you are using, SOA Security Manager treats characters as they are defined in Unicode. Although your code page can reference a special character as single-byte, SOA Security Manager treats it as a multi-byte character if Unicode defines it as such.

Authentication against an AD namespace

The Policy Server binds to Active Directory using SASL. If a user's common name (CN) is different from the user's Windows logon name, the user can still authenticate even if the EnableSaslBind registry setting exists on the Policy Server machine.

The EnableSaslBind setting is a DWORD registry key that you can set to 0 or 1:

HKLM\Software\Netegrity\SiteMinder\CurrentVersion\Ds\LDAPProvider\Enable SaslBind

This setting disables or enables the SASL protocol while authenticating users. For example, if EnableSaslBind does not exist and you configure this setting to 1, the bind occurs with SASL. If EnableSaslBind exists and you configure this setting to 0, the bind occurs with Simple Authentication mechanism.

Administrator Credentials

When configuring a user directory in the Active Directory (AD) namespace, specify the fully qualified domain name (FQDN) of the administrator in the Username field on the Administrator Credentials group box. If you do not satisfy this requirement, user authentication can fail.

LDAP Search Root Configuration

In order for the Policy Server to identify the AD domain of an AD namespace, which is necessary to read account lock status, configure the LDAP search root of the user directory as the DN of the domain. If you set the LDAP search root to any other DN, the Policy Server is not able to identify the AD domain and is therefore unable to read the Windows lockout policy associated with the domain. This situation can lead users that are locked through the AD console to appear enabled when viewed in the Administrative UI User Management dialog.

For example, create five users through the AD console at DN ou=People,dc=clearcase,dc=com and lock two of these users. The SOA Security Manager User Management dialog shows locked users as disabled only if the LDAP search root is configured as the DN of the AD domain (that is, dc=clearcase,dc=com). If you configure the LDAP search root as ou=People,dc=clearcase,dc=com, the locked users are incorrectly shown as enabled.

Disable Password Services Redirect for Natively Disabled Unauthorized Users

By default, SOA Security Manager reprompts users for credentials when they are unauthorized due to being natively disabled in the directory server. This behavior does not occur for users stored in Active Directory. Rather, SOA Security Manager redirects natively disabled users to Password Services, even if Password Services is not enabled for the authentication scheme protecting the resource. Create and enable IgnoreDefaultRedirectOnADnativeDisabled to prevent this Active Directory behavior.

IgnoreDefaultRedirectOnADnativeDisabled

Location:
HKEY_LOCAL_MACHINE/SOFTWARE/Netegrity/Siteminder/CurrentVersion/Ds/LDAPProvider

Values: 0 (disabled) or 1 (enabled)

Default: 0. If the registry key is disabled, the default behavior is in effect.

Note: If a password policy is in effect that specifies a redirect to Password Services, SOA Security Manager redirects the natively disabled users to Password Services regardless of the registry key's setting.

Active Directory Namespace Does Not Support Paging

The Active Directory namespace does not support paging, causing searches of more than 1000 users to fail. To support searches of large numbers of users in the Active Directory namespace, enable the following registry key by setting it to one:

HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Ds\LDAPProvider\EnablePagingADNameSpace

Values: 0 (disabled) or 1 (enabled)

Default Value: 0

LDAP Namespace for an Active Directory User Directory Connection

When accessing an Active Directory user directory using an LDAP namespace, disable the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Ds\LDAPProvider\EnableADEnhancedReferrals

Values: 0 (disabled) or 1 (enabled)

Default Value: 1

This step prevents LDAP connection errors from occurring.

LDAP Namespace for an Active Directory Connection

SOA Security Manager supports user directories on the Microsoft Active Directory platform. Although the configuration for Active Directory (AD) and LDAP namespaces in the Administrative UI is very similar, there are several functional differences.

The advantages of using the LDAP namespace for an Active Directory user store include:

- Support for enhanced LDAP referrals.
- Support for LDAP paging and sorting.

The disadvantages include:

- No support for Windows SASL

  The LDAP namespace does not support native Windows SASL, which allows native secure LDAP bind operations to support native Windows authentication methods such as Kerberos and NTLM.

- Indexing the Objectclass Attribute

  Microsoft Active Directory uses a non-standard method for identifying object classes. Because of this, the objectclass attribute in Active Directory is not indexed by default. This can cause the Administrative UI to timeout when it searches through an Active Directory LDAP implementation that includes large numbers of users or groups.

  For SOA Security Manager to run efficiently with an Active Directory user directory, you must index the objectClass attribute in Active Directory. For more information, see your Active Directory documentation.

- Active Directory and Password Services

  Microsoft Active Directory requires an SSL connection to change stored user passwords. For Password Services to work with Active Directory user directories, you must configure an SSL connection to any Active Directory LDAP user directory to which password policies will be applied.

  Additionally you must define a specific Password Attribute: unicodePWD to enable Password Services to work with Active Directory user directories.

  Note: For complete information about configuring Microsoft Active Directory, see your Active Directory documentation.

- Using a Windows User Security Context

  A SOA Security Manager Web Agent can run in a Windows user security context for accessing Web resources on IIS Web servers. Before SOA Security Manager can provide the Windows user security context, you must configure a session store and enable persistent sessions on a per realm basis (see How SOA Security Manager Is Configured to Provide a Windows User Security Context). You must also enable this feature in the Credentials and Connection tab in the User Directory dialog.

More information:

LDAP Referrals (see page 92)

AD Namespace for an Active Directory Connection

SOA Security Manager supports user directories on the Microsoft Active Directory platform. Although the configuration for Active Directory (AD) and LDAP namespaces in the Administrative UI is very similar, there are several functional differences.

The advantages of using the AD namespace when configuring an Active Directory user store include:

- SSL connectivity using a native Windows certificate database.

  Note: Both the Policy Server and the systems hosting Active Directory user stores must have an established trust. For information about configuring Windows systems and Active Directory for SSL, see your Windows documentation.

- Support for native Windows SASL which allows for secure LDAP bind operations.

The disadvantages include:

- No support for enhanced LDAP referrals.
- No support for LDAP paging and sorting operations.

Ping the User Store System

Pinging the user store system verifies that a network connection exists between the Policy Server and the user directory or database.

Note: Some user store systems may require the Policy Server to present credentials.

Configure Active Directory Connections

You can configure a user directory connection that lets the Policy Server communicate with an Active Directory user store.

To configure the user directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Select either AD or LDAP from the Namespace list on the Directory Setup group box.

   LDAP settings open.

   Note: Because Microsoft Active Directory is an LDAP-compliant user directory, you can configure an Active Directory connection using the AD namespace or the LDAP namespace.

4. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: Consider the following:

   - For more information about an authenticated user's security context, see How a Windows User Security Context is Obtained.
   - If you plan to use an SSL connection from the Policy Server to an Active Directory namespace, you must specify the FQDN and port number in the Server field on the Directory Setup group box. When the FQDN is not specified, an error is logged that states the user directory cannot be contacted. A Windows Event is also logged that reports the certificate does not match the server name.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.

5. (Optional) Click Configure on the Directory Setup group box to configure load balancing and failover.

   Note: More information on load balancing and failover exists in LDAP Load Balancing and Failover.

6. Select Require Credentials on the Administrator Credentials group box, and type the username and password of the administrator's account in the fields on the group box.

   Note: When configuring a user directory in the Active Directory (AD) namespace, you must specify the fully qualified domain name (FQDN) of the administrator in the Username field. Otherwise, user authentication can fail.

7. Type the LDAP Search and LDAP User DN Lookup settings in the fields on the LDAP Settings group box.

8. (Optional) Specify the user directory profile attributes that are reserved for SOA Security Manager's use in the fields on the User Attributes group box.

9. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.

10. Click Submit.

    The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 153)
Directory Attributes Overview (see page 99)
Define an Attribute Mapping (see page 186)
How to Configure an LDAP User Directory Connection over SSL (see page 144)

How to Configure an Active Directory Global Catalog User Directory Connection

You can use an Active Directory Global Catalog as a user store. The following process lists the steps for creating the user store connection to the Policy Server:

1. Ping the User Store System
2. Configure the Active Directory Global Catalog Connection

Ping the User Store System

Pinging the user store system verifies that a network connection exists between the Policy Server and the user directory or database.

Note: Some user store systems may require the Policy Server to present credentials.

Configure Active Directory Global Catalog Directory Connections

You can configure a user directory connection that lets the Policy Server communicate with an Active Directory Global Catalog user store.

The Policy Server user store supports the Global Catalog Support feature in Active Directory. However, SOA Security Manager features that require writing to Active Directory, such as Password Services, are not supported, because Global Catalog does not support writes to Active Directory.

To configure the user directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Select LDAP from the Namespace list.

   LDAP settings open.

4. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.

5. Type the LDAP Search and LDAP User DN Lookup settings in the fields on the LDAP Settings group box.

6. (Optional) Click Configure to configure load balancing and failover.

   Note: More information on load balancing and failover exists in LDAP Load Balancing and Failover.

7. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.

8. (Optional) Specify the user directory profile attributes that are reserved for SOA Security Manager's use in the fields on the User Attributes group box.

9. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.

10. Click Submit.

    The Create User Directory task is submitted for processing.

More information:

Define an Attribute Mapping (see page 186)
How to Configure an LDAP User Directory Connection over SSL (see page 144)

Configure a Global Catalog User Store With an SSL Connection

You can configure the Global Catalog and the Policy Server to communicate over an encrypted SSL connection.

To configure the Policy Server to use an SSL connection with the Global Catalog

1. Install the Certificate Authority's (CA) root certificate into the Netscape cert7.db database on each machine that expects to use SSL to communicate with the Global Catalog user directory.

   Note: SOA Security Manager requires the certificate to be in a Netscape version file format (cert7.db), so do not use Microsoft Internet Explorer to install the certificate.

2. In the Netscape Certificate Database File field on the Data tab on the Policy Server Management Console, configure the Policy Server to use SSL by specifying the path to the cert7.db file.

How to Configure an Oracle Internet Directory User Directory Connection

You can use an Oracle Internet Directory (OID) user directory as a user store. The following process lists the steps for creating the user store connection to the Policy Server:

1. Ping the User Store System
2. Create the Organizational Unit in Oracle Internet Directory
3. Configure the Oracle Internet Directory Connection

LDAP Referral Limitation for Oracle Internet Directory User Directory

LDAP referrals do not work when Oracle Internet Directory Server 10g (9.0.4) is configured as a user store and enhanced referrals are enabled. This is a limitation with OID.

Ping the User Store System

Pinging the user store system verifies that a network connection exists between the Policy Server and the user directory or database.

Note: Some user store systems may require the Policy Server to present credentials.

Create an Organizational Unit for an OID Directory

You can create an organizational unit for adding users to an OID directory.

To create an organizational unit for an OID directory

1. Create an organizational unit under a domain using the ADD.

   Example: OracleSchemaVersion

2. Select the organizational unit, and enter a Distinguished Name.

   Example: ou=people,cn=OracleSchemaVersion

3. Right-click Entry Management, and select Create.
4. Click Add on the Distinguished Name dialog, and select inetOrgPerson.
5. Type the following on the Mandatory Properties tab:
   - cn=user1
   - sn=user1
   - uid=user1
   - userpassword=user1
6. Specify the dn as: cn=user1,ou=people,cn=OracleSchemaVersion.

Configure Oracle Internet Directory Connections

You can configure a user directory connection that lets the Policy Server communicate with an OID user store.

To configure the user directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Select LDAP from the Namespace list.

   LDAP settings open.

4. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.

5. Type the LDAP Search and LDAP User DN Lookup settings in the fields on the LDAP Settings group box.
6. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.
7. (Optional) Specify the user directory profile attributes that are reserved for SOA Security Manager's use in the fields on the User Attributes group box.
8. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
9. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 153)
Define an Attribute Mapping (see page 186)
How to Configure an LDAP User Directory Connection over SSL (see page 144)

How to Configure an ODBC User Directory Connection

You can use an ODBC user directory as a user store. The following process lists the steps for creating the user store connection to the Policy Server:

1. Ping the User Store System
2. Configure the ODBC Directory Connection

Ping the User Store System

Pinging the user store system verifies that a network connection exists between the Policy Server and the user directory or database.

Note: Some user store systems may require the Policy Server to present credentials.

Configure ODBC Directory Connections

You can configure a user directory connection that lets the Policy Server communicate with an ODBC user store.

If you are using a SQL database for audit logs and caching is turned on, under heavy load, SOA Security Manager performance may suffer as the Policy Server queues messages for logging. Turn on asynchronous auditing for the realms associated with the resources being accessed by a high volume of users to alleviate the problem.

To configure the user directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory

   The Create User Directory pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Select ODBC from the Namespace list.

   ODBC settings open.
4. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.
5. Select a SQL query scheme.
6. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator who has an account on the user directory in the fields on the group box.

   Note: The user name must match the user who owns the tables containing user directory data. For example, if you are using the SmSampleUsers schema, this user must be the owner of the SmUser, SmUserGroup, and SmGroup tables. The administrator's account must have read or read/write privileges for the user directory.
7. (Optional) Specify the user directory profile attributes that are reserved for SOA Security Manager's use in the fields on the User Attributes group box.
8. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
9. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

SQL Query Schemes (see page 158)
Configure ODBC Data Source Failover (see page 157)
Define an Attribute Mapping (see page 186)

SQL Server User Store Case Insensitivity and Extra Trailing Spaces Password Issues

A SQL Server user store can be case insensitive and also ignore extra trailing spaces in users' passwords. This causes a problem because users entering passwords that are case insensitive or with extra trailing spaces can gain access to protected resources. For example, if a password policy is configured so that a user's password must be the case sensitive "ABCD", but the user enters "ABcd", SQL Server allows entry. In another example, if the password policy is configured so that a user's password must be " A B C", but the user enters " A B C ", SQL Server ignores the extra trailing spaces in the password and allows entry.

The following are solutions to these two issues.

First Issue: SQL Server User Store is Case Insensitive

The SQL Server database is not performing proper collation and does not recognize case sensitivity in passwords.

Solution 1

To impose case sensitivity to a SQL Server user store, select the proper collation during table creation when installing the database.

For more information about the collation, see the following:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/instsql/in_collation_3oa6.asp

Solution 2

If you already installed SQL Server with the default collation and the database does not recognize case sensitivity in passwords, create the proper collation when creating tables for the SOA Security Manager user store.

To specify the collation, modify and then import one of the following scripts.

siteminder_install\db\SQL\smsampleusers_sqlserver.sql

siteminder_install\db\SQL\smsampleusers_sqlserver_upgrade.sql

Note: These two script files use the default US English localization.

smsampleusers_sqlserver.sql Script File

Change the following lines highlighted in bold in the smsampleusers_sqlserver.sql script file:

```
CREATE TABLE SmGroup (
    GroupID       int NOT NULL,
    Name nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL ,
    PRIMARY KEY (GroupID)
)
DROP TABLE SmUser
go
CREATE TABLE SmUser (
  UserID       int NOT NULL,
  Name  nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL,
  Password nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL,
  LastName nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL,
  FirstName nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL,
  EmailAddress nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL,
  TelephoneNumber nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL ,
   Disabled nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL ,
   PIN     nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL ,
   Mileage int NOT NULL,
   PasswordData varchar(2000) NOT NULL,
   PRIMARY KEY (UserID)
)
go
```

After modifying the script, you need to import it into the SQL Server database.

Important! Back up any existing data as running this script removes existing data and creates new tables.

smsampleusers_sqlserver_upgrade.sql Script File

The following lines highlighted in bold are changes you need to make in the smsampleusers_sqlserver_upgrade.sql script file:

```
/* Upgrade table SmGroup*/
ALTER TABLE  SmGroup ALTER COLUMN Name nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
/* Upgrade table SmUser*/
ALTER TABLE  SmUser ALTER COLUMN Name nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN Password nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN LastName nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN FirstName nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN EmailAddress nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN TelephoneNumber nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN Disabled nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN PIN nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
```

After modifying the script, you need to import it into the SQL Server database.

Important! Back up any existing data as running this script removes existing data and creates new tables.

Second Issue: SQL Server Ignores Trailing Spaces in the Password

SQL Server pads the strings that are being compared and makes their lengths equal.

Solution

To have SQL Server recognize trailing white spaces in passwords stored in the database, modify the Authenticate User query in the ODBC Query Scheme object using the Administrative UI. To have SQL Server compare strings without padding or trimming, incorporate the LIKE predicate instead of the **=** operator. When the right side of a LIKE predicate expression features a value with a trailing space, SQL Server does not pad the two values to the same length before the comparison occurs. An example authentication query is:

```
select Name from SmUser where Name = '%s' and Password LIKE '%s'
```

Important! Using the LIKE predicate expression in the password matching query can open a security hole. If a user enters a '%' in the password, SQL Server treats it as wild card character for the LIKE predicate and this user can authenticate using more than one password.

Note the following:

- If you are creating the ODBC query scheme object using the SDK, you can specify the authentication query using the pszQueryAuthenticateUser attribute of the Sm_PolicyApi_ODBCQueryScheme_t object.
- If you are creating the ODBC query scheme object using Perl Scripting Interface, you can specify the authentication query while calling the CreateODBCQueryScheme() API.

SQL Server User Store Case Insensitivity and Extra Trailing Spaces Password Issues

A SQL Server user store can be case insensitive and also ignore extra trailing spaces in users' passwords. This causes a problem because users entering passwords that are case insensitive or with extra trailing spaces can gain access to protected resources. For example, if a password policy is configured so that a user's password must be the case sensitive "ABCD", but the user enters "ABcd", SQL Server allows entry. In another example, if the password policy is configured so that a user's password must be " A B C", but the user enters " A B C ", SQL Server ignores the extra trailing spaces in the password and allows entry.

The following are solutions to these two issues.

First Issue: SQL Server User Store is Case Insensitive

The SQL Server database is not performing proper collation and does not recognize case sensitivity in passwords.

Solution 1

To impose case sensitivity to a SQL Server user store, select the proper collation during table creation when installing the database.

For more information about the collation, see the following:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/instsql/in_collation_3oa6.asp

Solution 2

If you already installed SQL Server with the default collation and the database does not recognize case sensitivity in passwords, create the proper collation when creating tables for the SOA Security Manager user store.

To specify the collation, modify and then import one of the following scripts.

SOA_HOME\db\SQL\smsampleusers_sqlserver.sql

SOA_HOME_install\db\SQL\smsampleusers_sqlserver_upgrade.sql

Note: These two script files use the default US English localization.

smsampleusers_sqlserver.sql Script File

Change the following lines highlighted in bold in the smsampleusers_sqlserver.sql script file:

```
CREATE TABLE SmGroup (
    GroupID       int NOT NULL,
    Name nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL ,
    PRIMARY KEY (GroupID)
)
DROP TABLE SmUser
go
CREATE TABLE SmUser (
  UserID       int NOT NULL,
  Name  nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL,
  Password nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL,
  LastName nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL,
  FirstName nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL,
  EmailAddress nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL,
  TelephoneNumber nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL ,
   Disabled nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL ,
   PIN     nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL ,
   Mileage int NOT NULL,
   PasswordData varchar(2000) NOT NULL,
   PRIMARY KEY (UserID)
)
go
```

After modifying the script, you need to import it into the SQL Server database.

Important! Back up any existing data as running this script removes existing data and creates new tables.

smsampleusers_sqlserver_upgrade.sql Script File

The following lines highlighted in bold are changes you need to make in the smsampleusers_sqlserver_upgrade.sql script file:

```
/* Upgrade table SmGroup*/
ALTER TABLE  SmGroup ALTER COLUMN Name nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
/* Upgrade table SmUser*/
ALTER TABLE  SmUser ALTER COLUMN Name nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN Password nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN LastName nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN FirstName nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN EmailAddress nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN TelephoneNumber nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN Disabled nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
ALTER TABLE  SmUser ALTER COLUMN PIN nvarchar(255) COLLATE Latin1_General_CS_AS NOT NULL
go
```

After modifying the script, you need to import it into the SQL Server database.

Important! Back up any existing data as running this script removes existing data and creates new tables.

Second Issue: SQL Server Ignores Trailing Spaces in the Password

SQL Server pads the strings that are being compared and makes their lengths equal.

Solution

To have SQL Server recognize trailing white spaces in passwords stored in the database, modify the Authenticate User query in the ODBC Query Scheme object using the Administrative UI. To have SQL Server compare strings without padding or trimming, incorporate the LIKE predicate instead of the **=** operator. When the right side of a LIKE predicate expression features a value with a trailing space, SQL Server does not pad the two values to the same length before the comparison occurs. An example authentication query is:

```
select Name from SmUser where Name = '%s' and Password LIKE '%s'
```

Important! Using the LIKE predicate expression in the password matching query can open a security hole. If a user enters a '%' in the password, SQL Server treats it as wild card character for the LIKE predicate and this user can authenticate using more than one password.

Note the following:

- If you are creating the ODBC query scheme object using the SDK, you can specify the authentication query using the pszQueryAuthenticateUser attribute of the Sm_PolicyApi_ODBCQueryScheme_t object.
- If you are creating the ODBC query scheme object using Perl Scripting Interface, you can specify the authentication query while calling the CreateODBCQueryScheme() API.

How to Configure a Windows Directory Connection

You can use a Windows Directory as a user store. User directory connections can be set up with any of the following WinNT implementations:

- WinNT domain
- Individual WinNT computer in a domain
- Stand-alone WinNT computer

The following process lists the steps for creating the user directory connection in the Policy Server.

1. Meet the WinNT Domain Connection Requirements
2. Ping the User Store System
3. Configure the Windows Directory Connection

WinNT Domain Connection Requirements

For the Policy Server to connect to your WinNT domain, it must meet the following requirements:

- A one way trust relationship between the domain containing the Policy Server and the domain containing users must exist.
- Every user in the domain that SOA Security Manager will authenticate needs the "Access the computer from network" user right for the machine where the Policy Server is running.
- The account that SOA Security Manager uses to access the User Directory object needs to have access to the IPC$ share on the domain controller machine where the WinNT domain users are located.

  Note: These requirements may be met by default if you use the default configuration for your WinNT domain. Your WinNT domain administrator should verify that the domain meets the above requirements.

Note: For Windows deployments, SOA Security Manager establishes the Windows user context by passing the user's fully qualified Windows ID and password to IIS. SOA Security Manager obtains the fully qualified Windows ID from the user's DN entry by concatenating the first cn and dc values found in the DN. For example, if the user DN is:

cn=<username>,cn=<usergroup>,dc=<server>,dc=<domain>,
dc=<extension>

The resulting Windows ID is <server>\<username>. IIS requires that <username> be the same as the Windows user ID, and that <server> be the logon domain name.

The Policy Server authenticates against WinNT and can authorize users based on their individual identities and group membership.

When authenticating against a WinNT namespace, the Policy Server passes user credentials to WinNT for authentication. The credentials are the user's WinNT user name and password. In a SOA Security Manager environment, where multiple WinNT namespaces are defined, user authentication is faster if the user name supplied to SOA Security Manager includes the domain name (i.e. *domain\username)*. In that case, SOA Security Manager skips all WinNT namespaces that do not match the specified domain name.

WinNT user names and passwords can be used as credentials.

Note: To authenticate users against a WinNT domain, the Policy Server must run on WinNT.

Ping the User Store System

Pinging the user store system verifies that a network connection exists between the Policy Server and the user directory or database.

Note: Some user store systems may require the Policy Server to present credentials.

Configure a Windows Directory Connection

You can configure a user directory connection that lets the Policy Server communicate with a WinNT user store.

To configure the directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Select WinNT from the Namespace list.

   WinNT settings open.
4. Complete the remaining required connection information on the General and Directory Setup group boxes for the WinNT domain, a computer within the WinNT domain, or a stand-alone WinNT computer.

   - WinNT domains

     To define a namespace that represents global users and groups in a WinNT domain, specify the name of the domain in the NT Domain Name field. SOA Security Manager supports domain authentication and trusted domain authentication for user accounts in domains for user accounts in domains.

     Note: The system on which the Policy Server is installed must have a computer account in the appropriate domain for domain authentication to work. If the system does not have a computer account in all domains in which users need to be authenticated, the appropriate trust relationships must be established between domains.

     You can change the NT account under which the Policy Server is running by opening the Services dialog from the Control Panel and changing the account under which the Policy Server is running to an account that has the necessary privileges to access the specified domain. This account must have the "Act As Part of Operating System" system privilege.

- Individual WinNT computer in a domain

  To define a Namespace that represents local users and groups in a computer that is a member of a domain, specify the name of the domain followed by the name of the computer in the NT Domain Name field. The domain name and computer name must be separated by a forward slash (/). If you do not specify the name of the computer, performance during searches may suffer.

  Example: SampleDomain/Comp1

- Stand-alone WinNT computer

  To define a namespace that represents local users and groups in a stand-alone computer, specify the name of the computer in the NT Domain Name field. The stand-alone computer must have a Policy Server installed on it in order for this namespace to be accessible. There may be an initial delay when the Policy Server accesses this namespace for the first time.

5. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator who has an account on the user directory in the fields on the group box.
6. (Optional) Specify the user directory profile attributes that are reserved for SOA Security Manager's use in the fields on the User Attributes group box.
7. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
8. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

Directory Attributes Overview (see page 99)
Define an Attribute Mapping (see page 186)

LanMan User Directories

A LanMan user directory connection is an alternative method for distributing directory requests in a WinNT environment. The LanMan user directory connection option allows you to specify a failover list of Backup Domain Controllers (BDCs) used for each user directory lookup in the WinNT Registry. This list of BDCs takes precedence over the Primary Domain Controller (PDC), which handles directory requests in most WinNT deployments. Using a LanMan directory connection, the Policy Server sends WinNT directory requests to the first active BDC in the Registry list, rather than forcing requests to pass through the PDC.

More information:

LanMan User Directories (see page 459)

Failover for WinNT User Directories

Windows NT LanManager automatically provides failover for WinNT user directories by defining a list of Primary Domain Controllers (PDCs) and Backup Domain Controllers (BDCs). SOA Security Manager does not contain additional features for WinNT directory failover. To configure failover, use Windows NT LanManager.

Note: More information about LanManager exists in the Windows NT documentation.

WinNT Attributes in Responses

For the purpose of delivering information back to a Web application via responses, SOA Security Manager has access to the following default user attributes:

- Description
- FullName
- AccountExpirationDate
- UserFlags
- LoginWorkstations
- BadPasswordAttempts
- MaxLogins
- MaxStorage
- PasswordExpired
- LastLogin
- LastLogoff
- HomeDirectory
- Profile
- LoginScriptMinPasswordLength
- MaxPasswordAge
- MinPasswordAge
- PasswordHistoryLength

These user attributes can be accessed through the User Attribute feature on a Response pane. A response can return the value of any of these attributes to a SOA Security Manager Agent. For Web implementations, SOA Security Manager returns these attributes as name/value pairs in HTTP header variables.

More information:

Responses and Response Groups (see page 351)

How to Configure a Custom User Directory Connection

You can use a Custom directory as a user store. The following process lists the steps for creating the user store connection to the Policy Server:

1. Ping the User Store System
2. Configure the Custom Directory Connection

Ping the User Store System

Pinging the user store system verifies that a network connection exists between the Policy Server and the user directory or database.

Note: Some user store systems may require the Policy Server to present credentials.

Configure Custom Directory Connections

You can configure a user directory connection that lets the Policy Server communicate with a custom user store.

To configure the directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create User Directory: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Select Custom from the Namespace list.

   Custom settings open.

5. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.

6. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.

   Note: The Policy Server uses the shared library to determine the user attributes that are available to the custom directory. Before you enter user attributes, you must create the user directory connection.

7. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.

8. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

Directory Attributes Overview (see page 99)
Define an Attribute Mapping (see page 186)

How to Configure an LDAP User Directory Connection over SSL

Configuring an LDAP user directory connection over SSL requires that you configure SOA Security Manager to use your certificate database files.

Complete the following steps to configure the connection over SSL:

1. Before you configure a connection over SSL.
2. Install the NSS utility.
3. Create the certificate database files.
4. Add the root Certificate Authority (CA) to the certificate database.
5. Add the server certificate to the certificate database.
6. List the certifications in the certificate database.
7. Configure the user directory connection for SSL.
8. Point the Policy Server to the certificate database.
9. Verify the SSL connection.

Before You Configure a Connection over SSL

Review the following before configuring an LDAP user directory connection over SSL:

- Ensure your directory server is SSL-enabled.

  Note: For more information on configuring your directory server to communicate over SSL, refer to the vendor-specific documentation.

- SOA Security Manager uses a Netscape LDAP SDK to communicate with LDAP directories. As a result, SOA Security Manager requires that the database files be in a Netscape version file format (cert7.db).

  Important! Do not use Microsoft Internet Explorer to install certificates into your cert7.db database file.

- A third-party certificate utility, which is compatible with Netscape, is required to manage your SSL certificates. We recommend the Mozilla® Network Security Services (NSS) utility, version 3.2.2.

  Note: Version 3.2.2 is required to support the cert7.db format. Do not use later versions.

- (Active Directory) Considering the following:

  - If the SOA Security Manager user directory connection was configured with the AD namespace, the following process does not apply. The AD namespace uses the native Windows certificate repository when establishing an SSL connection. When configuring the AD namespace to communicate over SSL:

    - Ensure that the SOA Security Manager user directory connection is configured for a secure connection. For more information, refer to Configure the User Directory Connection for SSL (see page 151).

    - On the machine hosting the Active Directory instance, ensure that the root CA certificate and the server certificate are added to the services' certificate store.

    Note: For more information on configuring Active Directory to communicate over SSL, refer to the Microsoft documentation.

  - If the SOA Security Manager user directory connection was configured with the LDAP namespace, complete the following process to configure the connection over SSL.

Install the NSS Utility

You install the NSS utility to manage your certificate database files.

Note: Install the utility on a system to which the Netscape Portable Runtime (NSPR) or the Policy Server is installed. Installing the utility to a system with either component ensures that the necessary DLLs or shared objects are available.

To install the NSS utility

1. Access the Mozilla NSS 3.2.2 FTP site.
2. Download the respective zip or tar for your operating system.

   Note: A zip is not available for Windows Server 2000 or 2003. Download the zip for Windows NT.
3. Extract the NSS utility to a temporary location on the system to which you are managing your certificate database files.

Create the Certificate Database Files

The Policy Server requires that the certificate database files be in the Netscape version file format (cert7.db). You may use the NSS utility to create the certificate database files.

Note: The following procedure details the specific options and arguments to complete the task. For a complete list of the NSS utility options and arguments, refer to the Mozilla documentation on the NSS project page.

To create the certificate database files

1. From a command prompt, navigate to the bin directory in the location to which you extracted the NSS utility.

   Example: C:\nss\bin

   Note: Windows has a native certutil utility. Ensure you are working from the bin directory of the NSS utility or you may inadvertently run the Windows certutil utility.
2. Enter the following command:

   certutil -N -d *certificate_database_directory*

   -N

   Creates the cert7.db, key3.db, and secmod.db certificate database files.

-d *certificate_database_directory*

Specifies the directory to which the NSS utility is to create the certificate database files.

Note: If the file path contains spaces, bracket the path in quotes.

The utility prompts for a password to encrypt the database key.

3. Enter and confirm the password.

   NSS creates the required certificate database files:

   - cert7.db
   - key3.db
   - secmod.db

Example: Create the Certificate Database Files

```
certutil -N -d C:\certdatabase
```

Add the Root Certificate Authority to the Certificate Database

You add the root Certificate Authority (CA) to make it available for communication over SSL.

Note: The following procedure details the specific options and arguments to complete the task. For a complete list of the NSS utility options and arguments, refer to the Mozilla documentation on the NSS project page.

Important! If you are running a SOA Security Manager utility or executable on Windows Server 2008, be sure to open the command–line window with Administrator permissions, even if you are logged into the system as an Administrator. For more information, see the release notes for your SOA Security Manager component.

To add the root CA certificate to the certificate database

1. From a command prompt, navigate to the bin directory in the location to which you extracted the NSS utility.

   Example: C:\nss\bin

   Note: Windows has a native certutil utility. Ensure you are working from the bin directory of the NSS utility or you may inadvertently run the Windows certutil utility.

2. Run the following command to add the root CA to the database file:

certutil -A -n *alias* -t *trust_arguments* -i root_*CA_path* -d *certificate_database_directory*

-A

Adds a certificate to the certificate database.

-n *alias*

Specifies an alias for the certificate.

Note: If the alias contains spaces, bracket the alias with quotes.

-t *trust_arguments*

Specify the trust attributes to apply to the certificate when adding it to the certificate database. There are three available trust categories for each certificate, which are expressed in this order: "SSL, email, object signing". Specify the appropriate trust arguments so that the root CA is trusted to issue SSL certificates. In each category position, you may use zero or more of the following attribute arguments.

p

Valid peer.

P

Trusted peer. This argument implies p.

c

Valid CA.

T

Trusted CA to issue client certificates. This argument implies c.

C

Trusted CA to issue server certificates (SSL only). This argument implies c.

Important! This is a required argument for the SSL trust category.

u

Certificate can be used for authentication or signing.

-i *root_CA_path*

Specifies the path to the root CA file. Consider the following:

- The path must include the certificate name.
- Valid extensions for a certificate include .cert, .cer, and .pem.

Note: If the file path contains spaces, bracket the path in quotes.

-d *certificate_database_directory*

Specifies the path to the directory that contains the certificate database.

Note: If the file path contains spaces, bracket the path in quotes.

NSS adds the root CA to the certificate database.

Example: Adding a Root CA to the Certificate Database

```
certutil -A -n "My Root CA"  -t "C,," -i C:\certificates\cacert.cer -d C:\certdatabase
```

Add the Server Certificate to the Certificate Database

You add the server certificate to the certificate database to make it available for communication over SSL.

Note: The following procedure details the specific options and arguments to complete the task. For a complete list of the NSS utility options and arguments, refer to the Mozilla documentation on the NSS project page.

Important! If you are running a SOA Security Manager utility or executable on Windows Server 2008, be sure to open the command–line window with Administrator permissions, even if you are logged into the system as an Administrator. For more information, see the release notes for your SOA Security Manager component.

To add the server certificate to the certificate database

1. From a command prompt, navigate to the bin directory in the location to which you extracted the NSS utility.

   Example: C:\nss\bin

   Note: Windows has a native certutil utility. Ensure you are working from the bin directory of the NSS utility or you may inadvertently run the Windows certutil utility.

2. Run the following command to add the root certificate to the database file:

   certutil -A -n *alias* -t *trust_arguments* -i server_*certificate_path* -d *certificate_database_directory*

   -A

   Adds a certificate to the certificate database.

   -n *alias*

   Specifies an alias for the certificate.

   Note: If the alias contains spaces, bracket the alias with quotes.

-t *trust_arguments*

Specify the trust attributes to apply to the certificate when adding it to the certificate database. There are three available trust categories for each certificate, which are expressed in this order: "SSL, email, object signing". Specify the appropriate trust arguments so that the certificate is trusted. In each category position, you may use zero or more of the following attribute arguments:

p

Valid peer.

P

Trusted peer. This argument implies p.

Important! This is a required argument for the SSL trust category.

-i *server_certificate_path*

Specifies the path to the server certificate. Consider the following:

- The path must include the certificate name.
- Valid extensions for a certificate include .cert, .cer, and .pem.

Note: If the file path contains spaces, bracket the path in quotes.

-d *certificate_database_directory*

Specifies the path to the directory that contains the certificate database.

Note: If the file path contains spaces, bracket the path in quotes.

NSS adds the server certificate to the certificate database.

Example: Adding a Server Certificate to the Certificate Database

```
certutil -A -n "My Server Certificate" -t "P,," -i C:\certificates\servercert.cer -d C:\certdatabase
```

List the Certificates in the Certificate Database

You list the certifications to verify that they were added to the certificate database.

Note: The following procedure details the specific options and arguments to complete the task. For a complete list of the NSS utility options and arguments, refer to the Mozilla documentation on the NSS project page.

Important! If you are running a SOA Security Manager utility or executable on Windows Server 2008, be sure to open the command–line window with Administrator permissions, even if you are logged into the system as an Administrator. For more information, see the release notes for your SOA Security Manager component.

To list the certifications in the certificate database

1. From a command prompt, navigate to the bin directory in the location to which you extracted the NSS utility.

   Example: C:\nss\bin

   Note: Windows has a native certutil utility. Ensure you are working from the bin directory of the NSS utility or you may inadvertently run the Windows certutil utility.

2. Run the following command:

   certutil -L -d *certificate_database_directory*

   -L

   Lists all of the certificates in the certificate database.

   -d *certificate_database_directory*

   Specifies the path to the directory that contains the certificate database.

   Note: If the file path contains spaces, bracket the path in quotes.

   NSS displays the root CA alias, the server certificate alias, and the trust attributes you specified when adding the certificates to the certificate database.

Example: List the Certificates in the Certificate Database

certutil -L -d C:\certdatabase

Configure the User Directory Connection for SSL

You configure the user store connection to ensure that an SSL connection is used when the Policy Server and user store communicate.

Note: When you create or modify a Policy Server object in the [set the ufi variable for your book], use ASCII characters. Object creation or modification with non-ASCII characters is not supported.

To configure the user store connection for SSL

1. Log in to the Administrative UI.
2. Click Infrastructure, Directory.
3. Click User Directory, Modify User Directory.

   The Modify User Directory pane appears with a list of existing user directory connections.

4. Select the user directory connection you want, and click Select.

   User directory settings appear.

5. Select the Secure Connection check-box, and click Submit.

   The user directory connection is configured to communicate over SSL.

Point the Policy Server to the Certificate Database

You point the Policy Server to the certificate database to configure the Policy Server to communicate with the user directory over SSL.

Note: When you create or modify a Policy Server object in the [set the ufi variable for your book], use ASCII characters. Object creation or modification with non-ASCII characters is not supported.

To point the Policy Server to the certificate database

1. Start the Policy Server Management Console.

   Important! If you are accessing this graphical user interface on Windows Server 2008, open the shortcut with Administrator permissions, even if you are logged into the system as an Administrator. For more information, see the release notes for your SOA Security Manager component.

2. Click the Data tab.

3. Enter the path to the Netscape certificate database file in the Netscape Certificate Database File field.

   Example: C:\certdatabase\cert7.db

   Note: The key3.db file must also be in the same directory as the cert7.db file.

4. Restart the Policy Server.

The Policy Server is configured to communicate with the user directory over SSL.

Verify the SSL Connection

You verify the SSL connection to ensure the user directory and the Policy Server are communicating over SSL.

Note: When you create or modify a Policy Server object in the [set the ufi variable for your book], use ASCII characters. Object creation or modification with non-ASCII characters is not supported.

To verify the SSL connection

1. Log in to the Administrative UI.
2. Click Infrastructure, Directory.
3. Click User Directory, View User Directory.

   The View User Directory pane appears with a list of existing user directory connections.
4. Select the connection you want, and click Select.

   User directory settings appear.
5. Click View contents.

   If SSL is properly configured, the Directory Content pane appears and lists the contents of the user directory.

LDAP Load Balancing and Failover

The Policy Server can spread LDAP queries over multiple LDAP servers to enable failover and load balancing. If configured for failover, the Policy Server uses one LDAP server to fulfill requests until that server fails to respond. When the default server does not respond, the Policy Server routes the request to the next server specified for failover. This process can be repeated over multiple servers. Once the default server is able to fulfill requests again, the Policy Server routes requests to the original server.

If configured for load balancing, the Policy Server spreads requests over the specified LDAP servers. This distributes requests evenly across LDAP servers. Coupled with failover, load balancing provides faster, more efficient access to LDAP user directory information, with the added benefit of redundancy in the event of a server failure.

Port Number Considerations

You can assign ports to individual LDAP servers and failover groups, or let the Policy Server use the default port numbers for LDAP servers.

The following guidelines apply when specifying port numbers:

If	Then
any server in a failover group other than the last server contains a port number	The Policy Server assumes that servers in the group that do not have a specific port are using a default port. The default for SSL is 636. The default for non-SSL is 389. For example, a failover group of servers includes the following: 123.123.12.12:350 123.123.34.34 The first server in the failover group includes port 350. Communication with that server takes place on port 350. If the first server fails, the Policy Server communicates with the second server using the default port 389 because no port was specified for the second server in the failover group.

Configure Failover

You configure failover to provide for redundancy if the primary LDAP directory connection becomes unavailable.

Note: If you are adding a server for failover, the failover directory must use the same type of communication (SSL or non-SSL) as the primary directory, since both directories share the same port number.

To configure failover

1. Click Configure on the Directory Setup group box on the User Directory pane.

   The Directory Failover and Load Balancing Setup pane opens. The primary user directory opens in the Failover Group.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

2. Click Add Failover.

   Host and Port fields open.

3. Enter the host name and port of the server to which the Policy Server should failover.

   Note: If you do not specify a port number, the Policy Server uses the default port. The default port for SSL is 636. The default port for non-SSL is 389.

4. Repeat steps two and three to define additional failover servers.

   Note: If you specify a port for the last server, but do not specify a port for any other servers in the group, the Policy Server uses the specified port for every server in the group.

5. Click OK.

   The User Directory pane opens. The Server field lists the servers designated for failover. A space separates each server designated for failover.

Configure Load Balancing

You configure load balancing to have the Policy Server distribute requests evenly across LDAP servers.

To configure load balancing

1. Click Configure in the Directory Setup group box.

   The Directory Failover and Load Balancing Setup pane opens. The primary user directory opens in the Failover Group.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

2. Click Add Load Balancing.

   A new Failover Group opens.

3. Enter the host name and port of the server to which the Policy Server should load balance.

4. Repeat steps two and three to define additional load balancing servers.

5. Click OK.

   The User Directory pane opens. The Server field lists the servers designated for load balancing. A comma (,) separates each server designated for load balancing.

Configure Load Balancing and Failover

You configure load balancing and failover to spread requests over multiple servers, and to provide for redundancy if the primary directory connection becomes unavailable.

To configure load balancing and failover

1. Click Configure in the Directory Setup group box.

   The Directory Failover and Load Balancing Setup pane opens. The primary user directory opens in the Failover Group.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

2. Enter the host name and port of the server to which the Policy Server should failover.

   Note: If you do not specify a port number, the Policy Server uses the default port. The default port for SSL is 636. The default port for non-SSL is 389.

3. Repeat steps two and three to define additional failover servers.

   Note: If you specify a port for the last server, but do not specify a port for any other servers in the group, the Policy Server uses the specified port for every server in the group.

4. Click Add Load Balancing.

   A new Failover Group opens.

5. Enter the host name and port of the server to which the Policy Server should load balance.

   Note: You can add the same server multiple times for load balancing, which forces more requests to be serviced by a specific system. For example, consider two servers in a group: Server1 and Server2. Server1 is a high-performance server and Server2 is a lesser system. You can add Server1 to the load balancing list twice so that it will process two requests for each request processed by Server2.

6. Repeat steps five and six to define additional load balancing servers.

7. Click OK.

   The User Directory pane opens. The Server fields lists the servers designated for failover and load balancing. A space separates each server designated for failover. A comma (,) separates each server designated for load balancing.

Use Case - Load Balancing and Failover

In this example, a SOA Security Manager environment contains two user directories, A and B, which must meet the following requirements:

- User directory A must (1) failover to user directory B; and (2) load balance with B.
- User directory B must (1) failover to user directory A; and (2) load balance with and user directory A.

Where spaces represent failover and commas represent load balancing, the requirement is written as:

A B, B A

Solution:

The configuration requires two failover groups.

1. Add user directory B to the first failover group.

   The current configuration is A B.

2. Add a load balancing group.

   Note: load balancing groups open as new failover groups.

3. List user directory B as the first server in the load balancing group.

   The current configuration is A B, B.

4. List user directory A as the second sever in the load balancing group.

The result is two failover groups: "A B" and "B A", which load balance each other. If both directories are available, load balancing occurs between the first directories in each failover group: A and B. If user directory A becomes unavailable, failover occurs to user directory B. This results in user directory B handling all of the requests until user directory A becomes available.

Configure ODBC Data Source Failover

You configure failover to provide for redundancy if the primary and subsequent ODBC data sources become unavailable.

To configure failover

1. Click Configure in the Directory Setup group box.

   The ODBC Failover Setup pane opens. The primary data source opens in the data sources group.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

2. Click Add.

   A data source field opens.

3. Enter the data source to which the Policy Server should failover.

4. Repeat steps two and three to add additional data sources.
5. Click OK.

   The User Directory pane opens. The Server field lists the data sources designated for failover. A comma (,) separates each data source designated for failover.

SQL Query Schemes

The Policy Server uses SQL Query Schemes to build queries that find user data in a relational database. You create and edit SQL Query Schemes using the SOA Security Manager SQL Query Scheme dialog.

Note: The "SM_" prefix in column names is reserved for additional special names required by SOA Security Manager. Column names in your user directory should not begin with the "SM_" prefix. Policy Server errors will occur during user lookups if this prefix appears in column names.

Configure a SQL Query Scheme

You can configure a SQL Query Scheme that finds user data in the relational database that you are using as a user store.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure a SQL Query Scheme

1. Click Infrastructure, Directory.
2. Click SQL Query Scheme, Create SQL Query Scheme.

   The Create SQL Query Scheme pane opens.
3. Verify that Create a new object is selected, and click OK.

   The SQL Query Scheme: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Type a name and description in the fields on the General group box.

5. Update the contents of the query fields to correspond to your database schema.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

   You must configure each of the queries to work with your relational database. You must replace the following database table and column names with the table and column names from your relational database:

   - Name

     When you configure the Policy Server to use a user store residing in a relational database, the Name parameter for a user must be unique so that SOA Security Manager can correctly identify the user. Thus, two users cannot have the same user name.

   - SmUser—table
   - SmGroup—table
   - Password
   - SmUserGroup—table
   - Id
   - UserID
   - FirstName
   - LastName
   - TelephoneNumber
   - EmailAddress
   - Mileage
   - PIN
   - GroupID
   - Disabled

6. Select a query type, and click Submit.

   The query is saved. You can associate query scheme with a user directory connection

7. Restart the Policy Server using the Policy Server Management Console.

More information:

Configure ODBC Directory Connections (see page 130)

Add SQL Query Schemes to ODBC User Directory Connections

You can select an SQL Query Scheme using the User Directory Dialog.

To select an SQL Query Scheme

1. Open the User Directory pane for an existing user directory connection object.
2. Select the SQL query scheme from the SQL Query Scheme list, and click Submit.

   The SQL query scheme is saved to the directory connection.
3. Restart the Policy Server using the Policy Server Management Console.

Note: If you are using MS SQL Server, and your queries are returning names that include the apostrophe character (for example, O'Neil), you must replace any instance of '%s' in the query strings to ''%s''. To avoid this problem, base your queries on user IDs that do not include apostrophes, or modify the query strings that include '%s'.

How to Configure SQL Query Schemes for Authentication via Stored Procedures

When stored procedures are required for authentication with ODBC user directories, configure the SQL query scheme to call the stored procedure as follows:

SQLServer

Syntax: Call *Procedure_Name* %s , %s

Example: Call EncryptPW %s , %s

Stored procedures in SQLServer must meet the following requirements:

- The first parameter must be the username, and the second parameter must be the password.
- All parameters must be defined using the keyword OUT.
- The stored procedure must return an integer value.

The following example shows how to create a stored procedure for a SQLServer user directory:

```
CREATE PROCEDURE EncryptPW
@UserName varchar(20) OUT ,
@PW  varchar(20) OUT
AS
SELECT Smuser.name from Smuser where Smuser.name= @UserName and  password = @PW
SELECT Smuser.password from Smuser where name= @UserName and  password = @PW
return 0
```

MySQL

Syntax: Call *Procedure_Name* %s, %s

Example: Call EncryptPW %s, %s

Stored procedures in MySQL must meet the following requirements:

- The first parameter must be the username, and the second parameter must be the password.
- The stored procedure does not return a value.

The following example shows how to create a stored procedure for a MySQL user directory:

```
CREATE PROCEDURE EncryptPW(INOUT p_UserName varchar(20), INOUT p_PW varchar(20))
BEGIN
SELECT SmUser.Name into p_UserName from test.SmUser where SmUser.Name =
p_UserName and SmUser.Password = p_PW;
SELECT SmUser.Password into p_PW from test.SmUser where SmUser.Name =
p_UserName and SmUser.Password = p_PW;
END;
```

Oracle Functions

For Oracle user directories, you can create the following functions using the templates below:

- EncryptPW
- ChangePW

EncryptPW Function

The EncryptPW function must return an integer value, as follows:

- value = 0

  Specifies success.

- value = 1

  Specifies failure.

You can use the following template to create the EncryptPW function:

```
CREATE OR REPLACE FUNCTION EncryptPW(p_UserName IN OUT SmUser.Name%type, p_PW IN OUT
SmUser.Password%type)
RETURN INTEGER IS
nRet INTEGER :=1;
nCount NUMBER := 0;
BEGIN
select count(*) into nCount
from SmUser
where SmUser.Name = p_UserName and SmUser.Password = p_PW;
IF (nCount = 1) THEN
SELECT SmUser.Name  into p_UserName
from SmUser
where SmUser.Name = p_UserName and SmUser.Password = p_PW;
SELECT SmUser.Password into p_PW
from SmUser
where SmUser.Name = p_UserName and SmUser.Password = p_PW;
RETURN 0;
END IF;
RETURN nRet;
END EncryptPW;
```

ChangePW Function

The ChangePW function must return an integer value, as follows:

- value = 1

  Specifies success.

- value = 0

  Specifies failure.

You can use the following template to create the ChangePW function:

```
CREATE OR REPLACE FUNCTION ChangePW(p_PW IN SmUser.Password%type, p_UserName IN
SmUser.Name%type)
RETURN INTEGER IS
nRet INTEGER :=1;
nCount NUMBER := 0;
BEGIN
select count(*) into nCount
from SmUser
where SmUser.Name = p_UserName;
IF (nCount = 1) THEN
UPDATE SmUser
SET SmUser.Password = p_PW
where SmUser.Name = p_UserName;
COMMIT;
RETURN 0;
END IF;
```

Asynchronous Call Support During Failover and Connection Pooling

Synchronous calls are reliable, returning only after the request is complete. Asynchronous calls return immediately. A caller can choose to abandon an asynchronous call and avoid delays associated with network failures.

SOA Security Manager supports asynchronous calls to the following databases:

- SQLServer
- Oracle 8 on Windows NT and Solaris

Asynchronous Call Support Configuration

The following registry options are stored under the registry sub-key *Netegrity\SiteMinder\CurrentVersion\Database.*

AsynchronousCalls

Determines whether database calls are made asynchronously.

Values: 0 (no); 1 (yes)

Default: 0

AsynchronousSleepTime

Specifies the amount of time between calls to wait before checking the status of an outstanding SQL call.

Values: 0 to n milliseconds

Default: 15 milliseconds

LoginTimeout

The amount of time to allow for a connection to log in to the database.

Values: minimum of 1 second

Default: 15 seconds

QueryTimeout

The amount of time to allow for a query to complete before canceling it.

Values: minimum of 1 second

Default: 15 seconds

Note: When SQL Server is running on Windows NT, asynchronous call support causes a very small memory leak per abandoned connection. You may choose to extend the timeouts to reduce the number of failovers in an unreliable network by adjusting the settings discussed in the table above.

Configure Oracle 8 on Solaris for Asynchronous Calls

The Merant ODBC driver for Oracle on Solaris 2.6 and 2.7 may cause a core dump when asynchronous calls are supported. This is due to an Oracle bug which is fixed as follows:

Oracle 8.0.5

To each data source in system_odbc.ini in the <install_directory>/db directory, add the entry:

ArraySize=1

Note: This change turns off multi-row fetches and will affect performance when loading large policy stores.

Oracle 8.1.5

1. Remove or rename libclntsh.so in *siteminder*/bin and/or *siteminder*/odbc/lib.
2. Verify that the Oracle client has libclntsh.so installed in $ORACLE_HOME/lib. Refer to the Oracle documentation for installation and rebuilding instructions.
3. Make sure that LD_LIBRARY_PATH references the Oracle client library directory $ORACLE_HOME/lib.

If you get the following log message:

> [MERANT][ODBC Oracle 8 driver][Oracle 8] ORA-03106: fatal two-task communication protocol error

add an entry to the affected data source in system_odbc.ini in the <install directory>/db directory:

> ArraySize=<*value_greater_than 60000*>

This increases the size of the multi-row fetch buffer to eliminate the error. The default value of this variable is 60000 bytes. The maximum allowed value is 4 Gigabytes.

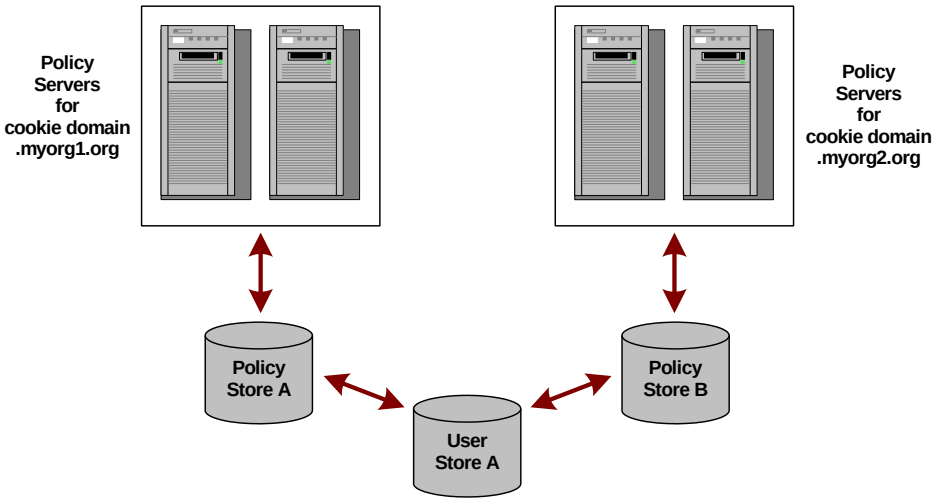
ODBC Connection Pooling

The following applies to ODBC connection pooling:

- Connections are pooled by ODBC data source. If a data source is used by one or more user directories AND (the policy store or another store on the policy server), the number of connections available may reach the total of the individual values specified for user directories and stores.
- SQL Server is more limited than Oracle in how connections may be shared. Two callers may not share an open result set while the results are being fetched to the client. Although the Policy Server uses server side cursors for SQL Server, there are limitations to concurrent activity, especially on multi-processor machines. Increasing the number of connections will generally improve concurrency.
- Oracle allows for multiple callers to share a connection, but may serialize calls internally. Again, you may see improved concurrency by increasing the number of connections allowed.
- If connections are shareable, the number of active requests will be balanced across the connections in a pool.
- Once a connection is opened, it is not closed until the Policy Server is shut down.

Define the Same User Directory Connection in Multiple Policy Stores

Every Policy Server is connected to a policy store. Multiple Policy Servers may be configured to point to a single policy store. When you open an instance of the Administrative UI, the objects that you add and modify are stored in the policy store associated with the Policy Server. As shown in the following figure, your SOA Security Manager environment may contain multiple independent policy stores for maintaining Policy Server data.



The Policy Servers for myorg1 are connected to Policy Store A. The Policy Servers for myorg2 are connected to Policy Store B. However, both organizations require data from User Store A.

To define a connection from multiple policy stores to a single user directory

1. Open the Administrative UI associated with one of the policy stores in your SOA Security Manager deployment.
2. Configure a user directory connection.

   When defining the user directory connection, note the value you supply in the Name field.
3. Open the Administrative UI associated with another policy store in your SOA Security Manager deployment.

4. Configure the same user directory connection.

   When defining the user directory connection, use the same Name that you used in step 2.

   For example, if you used a value of User Store A in the Name field when defining the user directory connection in the first policy store, to maintain single sign-on, you must configure the second policy store using a value of User Store A in the Name field of the User Directory Dialog.

5. Repeat this process for all independent policy stores in your SOA Security Manager deployment that will access the same user store.

   If you use the same user directory name when defining the connections to the user store in each independent policy store, SOA Security Manager can maintain single sign-on for users who access resources protected by policies in the different policy stores.

More information:

How to Configure a CA Directory User Directory Connection (see page 101)

View User Directory Contents

The Administrative UI lets you view the contents of a user directory.

To view user directory contents

1. Open the User Directory dialog for an existing user directory connection.

   Note: You cannot view the contents of a directory until you have saved the directory connection.

2. Click View Contents.

   The Directory Contents pane opens and lists the directory contents.

   Note: The default view lists groups. Use the search features to view individuals.

Search User Directories

The Administrative UI contains a user directory search feature. User directory searches let you view users or groups of users based on search expressions or directory attributes. User directory searches vary for each type of user directory.

Note: You can also access user directory searches from the Policy Users/Groups pane. You use this pane when adding or removing users and groups in a policy. The Policy Users/Groups pane contains the same search icon used to access a user search as described below.

To search a user directory

1. Open the User Directory pane for the directory that you want to search.
2. Click View Contents.

   The Directory Contents pane opens. The contents of the pane differ based on the type of directory you are viewing.
3. Enter your search criteria, and click Go.

   Results that match your criteria open.

Universal IDs

A Universal ID (UID) is a customer-specific user identifier to any application that is under SOA Security Manager control. UIDs are often different from user login names.

UIDs allow SOA Security Manager to bridge the gap between new applications and legacy applications or to avoid changes in underlying user repositories. The goal is to make the process of delivering this ID to applications automatic, regardless of the number or types of applications. For example, a company may have legacy applications that look up user information according to an employee ID number. Since the Policy Server uses a login name to identify a user in a directory, the UID provides a means for the Policy Server to identify the user, while still collecting the employee ID number from a user directory for use by other applications.

When you configure a user directory connection in the Administrative UI, you can specify a UID in the User Attributes group box on the User Directory pane.

More information:

How to Configure a CA Directory User Directory Connection (see page 101)

How SiteMinder Uses UIDs

When you configure a user directory connection with a UID, once a user logs into SOA Security Manager, the Policy Server fetches the UID from the designated attribute in the user's directory profile.

This value is placed in the session ticket (SESSIONSPEC) and returned to the requesting SOA Security Manager Agent. Web Agents make this value available to web-based applications in a header variable (HTTP_SM_UNIVERSALID). This value can be passed to applications or objects designed using the Agent API to validate the session ticket or to ask for an authorization. In either case the UID is returned as part of successful outcome.

Named Expressions

User directories store user attributes such as organizational information, user and group attributes, and individual credentials. SOA Security Manager can read some user attribute values directly from the user directory, while other values must be calculated each time that they are needed. These calculations are stored as expressions that can be named or unnamed.

Unnamed expressions are stored in policy store objects like responses and rules. *Named expressions* are policy store objects that can be referenced by name and reused. SOA Security Manager evaluates all expressions, both named and unnamed, to determine the values of calculated user attributes.

To create named expressions, an administrator must have the appropriate privileges.

Note: Active expressions and named expressions are not the same. While both types of expressions are evaluated at run-time, they differ in the following ways:

- While active expressions are Boolean expressions, named expressions can return a string, number, or Boolean value.
- While active expressions are referenced as is and must be reentered each time that they are used, named expressions are referenced by name and can be referenced from anywhere and reused.

Benefits of Named Expressions

Named expressions:

- Apply to multiple user directories

  Named expressions are stored in the policy store as objects that can be referenced by name and reused. SOA Security Manager evaluates named expressions to determine the values of calculated user attributes.

- Facilitate ease of reuse

  System administrators create each named expression once. Domain administrators reference the expression name, not the underlying expression, to obtain user information. Administrators do not have to reenter the entire expression each time that the user information is required.

- Reduce data entry errors

  System administrators create and manage named expressions in one place. If an expression must be changed, the administrator only makes the change once.

- Ease maintenance tasks

  If business logic requires a change to an expression, system administrators only make the change once. Domain administrators can continue to reference the expression name without regard for the underlying change.

- Enforce security

  Only administrators who have the appropriate privileges can create named expressions. Named expressions can call privileged built-in functions and any named expression, including those that are marked as private.

  For example, a named expression can call a private expression that adds the current user to a group, while an unnamed expression cannot. This restriction prevents a domain administrator from bypassing security, such as adding the current user to an administrative group.

Define Named Expressions

Named expressions are policy store objects that can be referenced by name and reused. SOA Security Manager evaluates named expressions to determine the values of calculated user attributes.

There are two types of named expressions:

- Virtual user attributes (see page 170)
- User classes (see page 172)

Virtual User Attributes

A virtual user attribute lets you define a re-usable expression to calculate user information. You use this type of expression when the user attribute is not uniquely referenced by the user directory. Rather, the user attribute must be calculated using attributes and other criteria that is established by business logic.

Virtual user attributes name expressions that result in values having one of the following data types:

- string
- number
- Boolean

Virtual user attributes are prefixed by the "pound" sign (#). The "pound" sign prevents name clashes with user attribute names and mappings and is a visual reminder that the user attribute value is calculated.

As an expression, a virtual user attribute can include:

- User attributes, either directory-specific or mapped
- References to other named expressions
- SOA Security Manager built-in functions and expression syntax

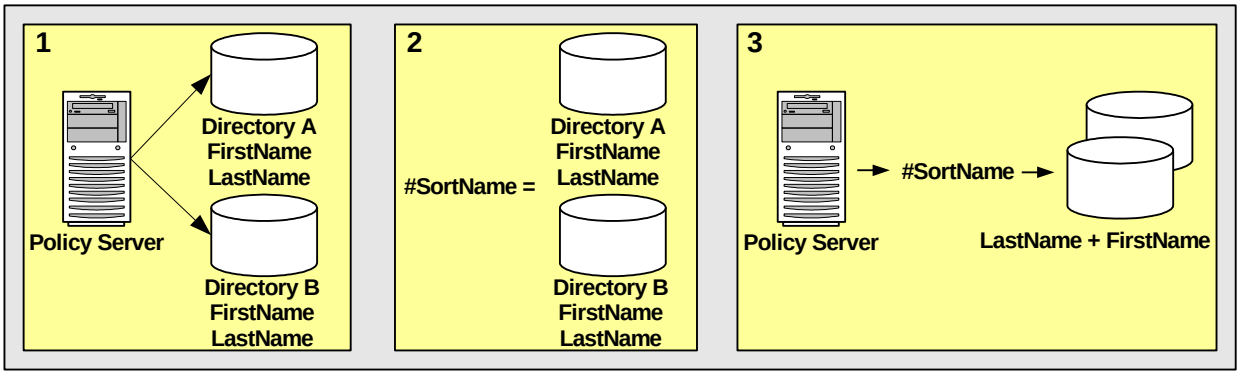
More information:

Expression Syntax Overview (see page 466)

Virtual User Attribute Use Case

This use case represents a basic scenario in which two LDAP user directories identify the last and first names of users with different underlying schema.

The following illustration shows how the virtual user attribute *#SortName* (LastName,FirstName) can be calculated for users in different user directories through user attribute mapping. User attribute mapping lets you map one common name to different user attribute names in different user directories.



1. Two user directories identify the last and first names of users differently. To create a common view of this information, you can create user attribute mappings:
   - FirstName maps to the underlying directory schema that identify the first names of users in Directory A and Directory B.
   - LastName maps to the underlying directory schema that identify the last names of users in Directory A and Directory B.
2. *#SortName* is a virtual user attribute that can calculate the sort name of users in both directories with the following expression:

   (LastName + "," + FirstName)

3. Instead of entering the expression (LastName + "," + FirstName) repeatedly, you can create a virtual user attribute named *#SortName* that is defined as: (FirstName + "," + LastName). Then, you can enter *#SortName* each time that the expression is needed.

Define a Virtual User Attribute

You define a virtual user attribute to calculate user information that is not uniquely referenced by one or more user directories.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To define a virtual user attribute

1. Click Policies, Expressions.
2. Click Named Expression, Create Named Expression.

   The Create Named Expression pane opens.
3. Verify that a new object of type Expression is selected, and click OK.

   The Create Named Expression: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Select Virtual User Attribute, and type the name and a description in the fields on the General group box.
5. Type the expression in the Expression field on the Add Named Expression group box.
6. (Optional) Select the Disabled check box on the Add Named Expression group box.

   The named expression is marked as disabled, is not listed in the expression editor, and cannot be called by another expression, named or unnamed.
7. (Optional) Select the Private check box on the Add Named Expression group box.

   The named expression is marked as private and can only be called by other named expressions; it cannot be called by unnamed expressions.
8. (Optional) Click Edit on the Add Named Expression group box.

   The Expression Editor pane opens.
9. Click Submit.

   The Create Named Expression task is submitted for processing.

User Classes

A user class lets you define a re-usable expression to calculate user information. You use this type of expression when the user attribute is not uniquely referenced by the user directory. Rather, the user attribute must be calculated using attributes and other criteria that is established by business logic.

A user class names an expression that returns a TRUE value if a user is a member of a specified class or a FALSE value if not.

User classes are prefixed by the "at" symbol (@). The "at" symbol prevents name clashes with user attribute names and mappings and is a visual reminder that the user attribute value is calculated.

As an expression, a user class can include:

- User attributes, either directory-specific or mapped
- References to other named expressions
- SOA Security Manager built-in functions and expression syntax

A user class is not a role. A role is a feature of Enterprise Policy Management. While roles can use user classes, they have additional information associated with them. For more information about roles, see the Enterprise Policy Management.

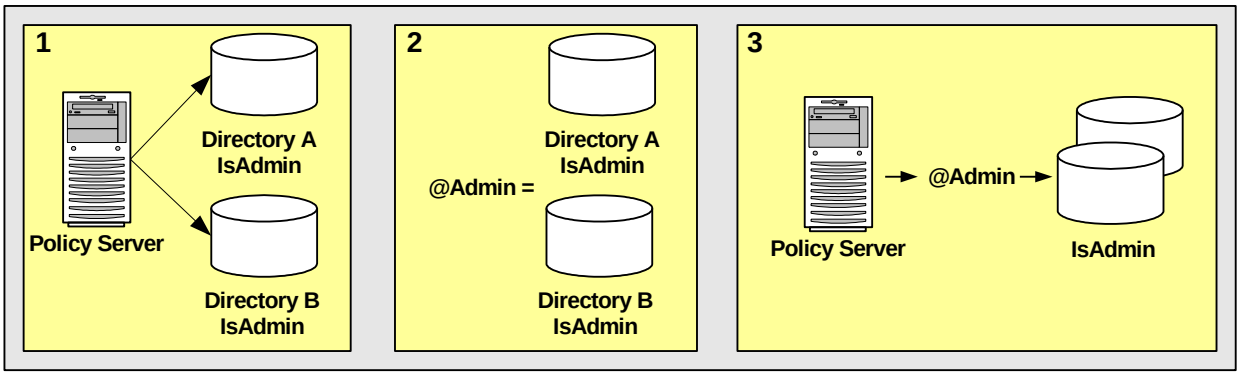
More information:

Expression Syntax Overview (see page 466)

User Class Use Case

This use case represents a basic scenario in which two LDAP user directories identify membership in the Administrator group using different underlying schema.

The following illustration details how the user class *@Admin* can be calculated for users in different user directories through user attribute mapping. User attribute mapping lets you map one common name to different user attribute names in different user directories.



1. Two user directories identify membership in the Administrator group differently. To create a common view of this information, you can create user attribute mappings:
   - IsAdmin maps to the underlying directory schema that identifies membership in the Administrator group in Directory A.
   - IsAdmin maps to the underlying directory schema that identifies membership in the Administrator group in Directory B.
2. *@Admin* is the named expression of type user class that SOA Security Manager evaluates to determine if users in both directories are Administrators:

   (IsAdmin)
3. Instead of entering the expression (IsAdmin) repeatedly, you can create a user class named *@Admin* that is defined as: (IsAdmin). Then, you can enter *@Admin* each time that the expression is needed.

Define a User Class

You define a user class attribute to calculate user information that is not uniquely referenced by one or more user directories. The result of the calculation can only be TRUE or FALSE. The result either applies to the user or it does not.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To create a user class

1. Click Policies, Expressions.
2. Click Named Expression, Create Named Expression.

   The Create Named Expression pane opens.
3. Verify that a new object of type Expression is selected, and click OK.

   The Create Named Expression: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select User Class, and type the name and a description in the fields on the General group box.
5. Type the expression in the Expression field on the Add Named Expression group box.

   Note: The expression must be a Boolean expression.

6. (Optional) Select the Disabled check box on the Add Named Expression group box.

   The named expression is marked as disabled, is not listed on the Expression Editor pane, and cannot be called by another expression, named or unnamed.

7. (Optional) Select the Private check box on the Add Named Expression group box.

   The named expression is marked as private and can be called by other named expressions, but not by unnamed expressions.

8. (Optional) Click Edit on the Add Named Expression group box.

   The Expression Editor pane opens.

9. Click Submit.

   The Create Named Expression task is submitted for processing.

How to Use the Expression Editor

You can use the expression editor to:

- Look up SOA Security Manager functions and operators and user-defined named expressions
- Build a Boolean expression

Note: If you prefer to enter an expression directly, you can click Cancel and return to the Create Expression: *Name* pane, where you can type the expression in the Expression field on the Add Named Expression group box.

Building a Boolean expression in the expression editor is a two-part process. The parts of the process can be repeated in any order:

1. Create conditions
2. Edit the expression

In the first part of the process, you can create conditions and add them to the Infix Notation group box. A *condition* is a simple Boolean expression that consists of a single SOA Security Manager function or operation. In the editor, a function can have up to three parameters and has the following format:

```
FUNCTION_NAME(parameter_1[, parameter_2][, parameter_3])
```

An operation requires two operands and has the following format:

```
left_operand operator right_operand
```

Since conditions are Boolean expressions, they result in a Boolean value. If a condition contains a function or operation that results in a string, it will be converted to a Boolean value. Specifically, the following string values are converted to TRUE: "TRUE", "true", "YES", and "yes". All other string values are converted to FALSE.

Likewise, if a condition contains a function or operation that results in a number, it will be converted to a Boolean value. All non-zero numbers are converted to TRUE, while zero is converted to FALSE.

Each condition is displayed on a separate line in the field on the Infix Notation group box and is connected to the condition in the line above by one or two Boolean operators, as follows:

condition_1
AND | OR | XOR [NOT] condition_2

In the second part of the process, you can edit the expression by modifying and deleting the conditions, changing the parentheses that group the conditions, and by changing the Boolean operators that connect the conditions in the field on the Infix Notation group box. For example, you can change how the conditions are grouped:

(condition_1
AND condition_2)
OR NOT condition_3

can become

condition_1
AND (condition_2
OR NOT condition_3)

Create a Condition Containing a Function

You can create a condition containing a built-in SOA Security Manager function and add the condition to an expression in the expression editor.

To create a condition containing a built-in SOA Security Manager function

1. Select a name from the drop-down list of functions or type a name in the Function field on the Condition group box on the Expression Editor pane.
2. Specify the first parameter by clicking Named Expression or by typing it in the First Parameter field on the Condition group box.

   Note: Clicking Named Expression opens the Variable Lookup group box.

3. (Optional) Specify the second parameter by clicking Named Expression or by typing it in the Second Parameter field on the Condition group box.

   Note: Clicking Named Expression opens the Variable Lookup group box.

4. (Optional) Specify the last parameter by selecting TRUE or FALSE from the drop-down list or by typing it in the Last Parameter field on the Condition group box.

5. Click Add.

   The specified function is added to the Infix Notation and Resulting Notation group boxes.

Create a Condition Containing an Operation

You can create a condition containing a built-in SOA Security Manager operation and add the condition to an expression in the expression editor.

To create a condition containing a built-in SOA Security Manager operation

1. Select an Operator Type and an Operator from the drop-down lists on the Condition group box on the Expression Editor pane.

2. Specify the left operand by clicking Named Expression or by typing it in the Left Operand field on the Condition group box.

   Note: Clicking Named Expression opens the Variable Lookup group box.

3. Specify the right operand by clicking Named Expression or by typing it in the Right Operand field on the Condition group box.

   Note: Clicking Named Expression opens the Variable Lookup group box.

4. Click Add.

   The specified operation is added to the Infix Notation and Resulting Notation group boxes.

How to Edit an Expression

Each condition that you create in the expression editor is displayed on a separate line in the field on the Infix Notation group box. As you build an expression, you can change the parentheses that group the conditions and the Boolean operators that connect the conditions by using the buttons on the Infix Notation group box.

Editing an expression is a three-step process. The first step includes four options, which can be repeated in any order:

1. Select an option:
   - Modify a Condition in an Expression (see page 178)
   - Delete a Condition from an Expression (see page 178)
   - Group the Conditions in an Expression (see page 179)
   - Change a Boolean Operator in an Expression (see page 180)
2. (Optional) Repeat step 1.
3. Close the expression editor by clicking OK.

Modify a Condition in an Expression

You can modify a condition in an expression by clicking the Modify button on the Infix Notation group box in the expression editor.

To modify a condition in an expression

1. Select a condition by clicking it.
2. Click Modify.

   The Edit group box opens, and the condition is displayed in the group box.

Delete a Condition from an Expression

You can delete one or more conditions from an expression by clicking the Remove button on the Infix Notation group box in the expression editor.

To delete a condition from an expression

1. Select a condition by clicking it.

   Note: To select multiple adjacent conditions, hold down the Shift key while clicking.

2. Click Remove.

   The selected condition is removed from the expression.

   Note: If multiple conditions are selected, clicking Remove deletes them one at a time.

Group the Conditions in an Expression

You can change the grouping of conditions in an expression by clicking the buttons that add and remove parentheses on the Infix Notation group box in the expression editor.

To change the grouping of conditions in an expression

1. Select two or more adjacent conditions by clicking them.

   Note: To select multiple adjacent conditions, hold down the Shift key while clicking.

2. Click one of the two following buttons:

   ( )

   Adds parentheses to the outside of the selected conditions.

   Example:

   condition_1

   AND condition_2

   becomes

   (condition_1

   AND condition_2)

   Remove( )

   Deletes parentheses from the outside of the selected conditions.

   Example:

   (condition_1

   OR condition_2

   OR condition_3)

   becomes

   condition_1

   OR condition_2

   OR condition_3

   The edited expression is displayed in the fields on the Resulting Notation and Infix Notation group boxes in the expression editor.

Change a Boolean Operator in an Expression

You can change a Boolean operator in an expression by clicking one of the following buttons on the Infix Notation group box in the expression editor:

- And/Or
- Not
- XOR
- Conditional?YES:NO

To change a Boolean operator in an expression

1. Select one condition or group of conditions by clicking it.

   Note: To select multiple adjacent conditions, hold down the Shift key while clicking.

2. Click one of the following buttons:

   And/Or

   Switches between the Boolean operators AND and OR.

   Example:

   AND condition_1

   becomes

   OR condition_1

   Note: The AND/OR button switches XOR to AND.

   Not

   Switches between adding and removing the Boolean operator NOT.

   Example:

   AND condition_1

   becomes

   AND NOT condition_1

   XOR

   Switches the Boolean operators AND and OR to XOR.

   Example:

   AND condition_1

   becomes

   XOR condition_1

   Note: The exclusive OR (XOR) operator takes two Boolean operands and returns TRUE if either operand is TRUE, but not both.

Conditional?YES:NO

Adds the conditional decision operator.

Example:

condition_1

becomes

condition_1 ? "YES" : "NO"

The edited expression is displayed in the fields on the Resulting Notation and Infix Notation group boxes in the expression editor.

Apply Named Expressions

This use case represents a scenario in which a retail clothing company wants to define a role that prevents customers from making Web-based credit purchases if they have met or exceeded their credit limit. The company policy dictates that customers have a $1,000 credit limit, while company employees have a $2,000 credit limit.

In this use case, the SOA Security Manager environment contains two user directories:

- Directory A stores employees. Employees can also be customers. Therefore, Directory A identifies customers as those employees who are members of the group: cn=Customers,ou=Groups,o=acme.com.
- Directory B only stores customers. Because every user is a customer, Directory B does not have a user attribute that identifies customers.

The following details how you can use attribute mapping, virtual user attributes, and user classes to satisfy the company's credit policy.

1. Create user attribute mappings and a universal schema or common name that identifies customers for each user directory:
   a. Create a *group name* attribute mapping for Directory A (employees):
      - Name the mapping **IsCustomer**.
      - Define IsCustomer as **cn=Customers,ou=Groups,o=acme.com**.
   b. Create a *constant* attribute mapping for Directory B (customers):
      - Name the mapping **IsCustomer**.
      - Define IsCustomer as **TRUE**.

      Note: IsCustomer is a common name that maps to the same user information in Directories A and B. To access this information, you can use IsCustomer in an expression.

2. Create *constant* attribute mappings and a universal schema or common name that identifies the company's credit limit for each user directory:
   a. Create a *constant* attribute mapping for Directory A (employees):
      - Name the mapping **CreditLimit**.
      - Define CreditLimit as **2000**.
   b. Create a *constant* attribute mapping for Directory B (customers):
      - Name the mapping **CreditLimit**.
      - Define CreditLimit as **1000**.

      Note: CreditLimit is a common name that maps to the same user information in Directories A and B. To access this information, you can use CreditLimit in an expression.
3. Assume that **#CreditBalance** is a virtual user attribute that retrieves the user's credit balance from the accounting database.
4. Create a user class that returns a TRUE value if a customer's credit balance is under the credit limit:
   - Name the user class **@IsUnderCreditLimit**.
   - Define @IsUnderCreditLimit as:

     (IsCustomer AND (#CreditBalance < CreditLimit))

     Note: This expression conforms to the syntax rules of a SOA Security Manager expression.
5. Create an EPM Role that lets customers make Web-based purchases if their credit balance is less than their credit limit:
   - Name the Role **PurchaseWithCredit**
   - Define the Role as **@IsUnderCreditLimit**

Note: For more information about EPM Roles, see Enterprise Policy Management.

More information:

Attributes and Expressions Reference (see page 463)

User Attribute Mapping

When you configure a connection from the Policy Server to a user directory, you can map SOA Security Manager's seven built-in attributes to directory-specific user attribute names.

- Universal ID
- Disabled Flag
- Password Attribute
- Password Data
- Anonymous ID
- Email
- Challenge/Response

User attribute mapping extends this capability by allowing you to define your own common names in SOA Security Manager and to map each one to user attribute names in multiple user directories with different underlying schema. After the connections from the Policy Server to the user directories are configured, you can use one common name to reference the same user information in different user directories.

User Attribute Mapping Overview

There are five types of user attribute mappings and two types of named expressions. The attribute mapping types are:

- alias
- group name
- mask
- constant
- expression

The named expression types are:

- virtual user attributes
- user classes

User attribute mappings are similar to named expressions, but there are important differences, as follows:

- Access
  - While some types of user attribute mappings are read only (R) and map to user attribute values that cannot be changed, other types of user attribute mappings are read/write (RW) and map to user attribute values that can be read or changed:

    Read Only (R): Designates a mapping whose target can be read, but not changed.

    Read/Write (RW): Designates a mapping whose target can be read or changed.
  - All named expressions are read only (R).
- Data Types
  - User attribute mappings map to user attributes that have specified data types.
  - Named expressions, when evaluated, result in specified data types.
- Visibility
  - User attribute mappings are not global and must be defined for each user directory to which they apply.
  - Named expressions are global and can apply to any user in any user directory.
- Prefix?
  - User attribute mappings follow the same syntax rules as user attribute names.
  - Named expressions follow the same syntax rules as user attribute names and have a prefix:
    - Virtual user attributes must begin with the "pound" sign (#).
    - User classes must begin with the "at" sign (@).

For a summary of these differences, see the following tables:

User Attribute Mapping Type	Data Types	Visibility	Prefix?
alias (RW)	string, number, Boolean	directory-specific	no
group name (RW)	Boolean	directory-specific	no
mask (RW)	Boolean	directory-specific	no
constant (R)	string, number, Boolean	directory-specific	no
expression (R)	string, number, Boolean	directory-specific	no

Named Expression Type	Data Types	Visibility	Prefix?
virtual user attribute (R)	string, number, Boolean	global	#
user class (R)	Boolean	global	@

How Attribute Mapping Works

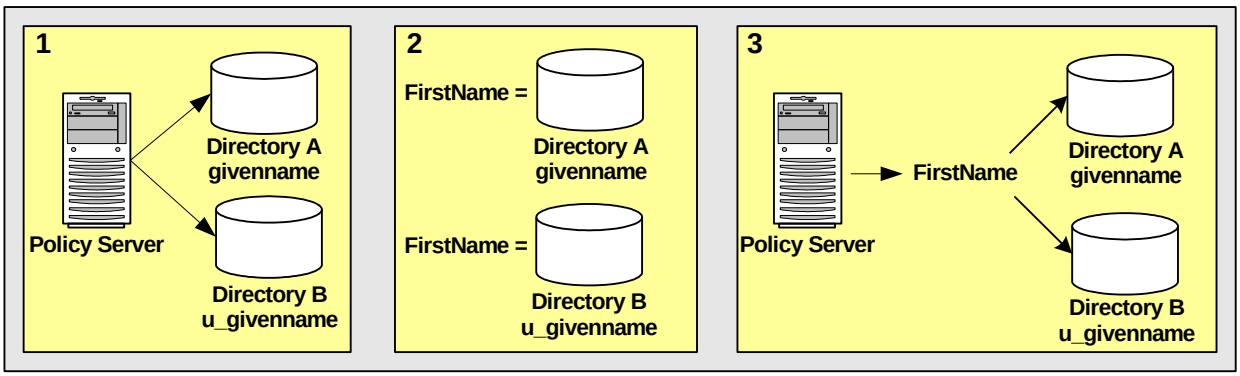
User directories store user information such as organizational information, user and group attributes, and individual credentials. Multiple user directories in a SOA Security Manager environment often store the same user information, but use different underlying schema and user attribute names to identify them. This results in a disparate view of the same user information from a SOA Security Manager perspective.

The purpose of user attribute mapping is to create a common view of the same user information by defining a universal schema. SOA Security Manager uses this universal schema to resolve user information across multiple user directories.

You can define a user attribute mapping by mapping a *common name* to the underlying directory schema that identifies a user attribute. Mapping the same common name to the underlying schema of each user directory in the environment results in a universal schema for the user attribute. This creates a common view of the same user information.

Creating such a view lets SOA Security Manager reference user attributes without regard for the directory type, greatly reducing the number of policies or other objects that must be configured to account for multiple user directories. Each user attribute mapping is specific to the user directory in which it is defined.

The following illustrates the basic concept of user attribute mapping:



1. Two user directories identify the first name of users differently:
   - Directory A identifies the first name of users with givenname.
   - Directory B identifies the first name of users with u_givenname.

   This results in two different representations and views of the same user information.
2. FirstName is a common name that is mapped to the underlying directory schema:
   - FirstName is mapped to givenname in Directory A.
   - FirstName is mapped to u_givenname in Directory B.
3. FirstName results in a common view of the same user information. You can reference FirstName when defining policies, expressions, or other objects that require the first name of users, without concern for the directory-specific schema, because the directories are operationally identical. SOA Security Manager determines that FirstName is givenname in Directory A and u_givenname in Directory B.

Define an Attribute Mapping

User attribute mapping lets you map one common name to the underlying directory schema of multiple user directories. Mapping the same common name to the underlying schema of each user directory in the environment results in a universal schema for the user information. Each user attribute mapping is specific to the user directory in which it is defined.

You can use one or more of the following attribute mapping types to define mappings that identify the same user information across multiple user directories:

- Alias (see page 187)
- Group Name (see page 189)
- Mask (see page 192)
- Constant (see page 196)
- Expression (see page 198)

Alias

Alias attribute mapping lets you map a common name to the name used by the underlying directory schema to identify a user attribute. *Alias* attribute mappings map common names to user attribute values that can be read or changed. This type of access is called read/write (RW).

Alias attribute mappings can map to user attributes that have any of the following data types:

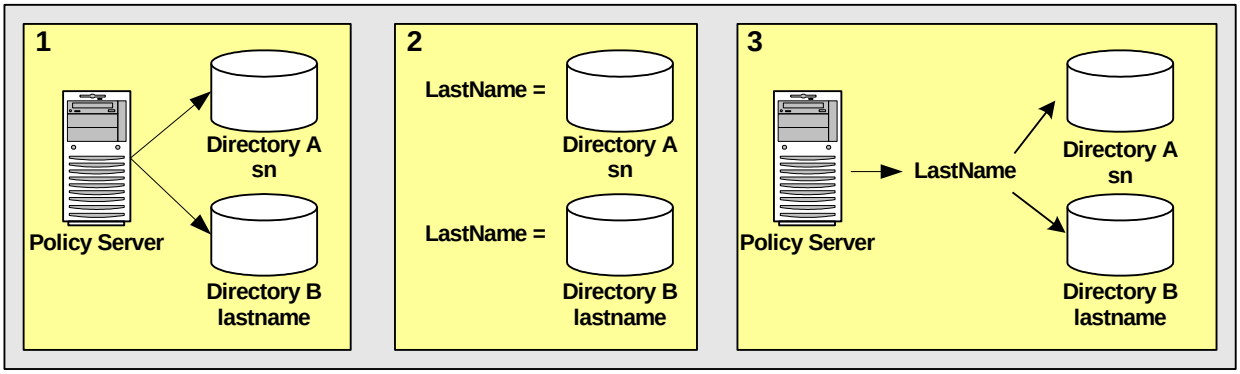
- string
- number
- Boolean

Alias Attribute Use Case

This use case represents a basic scenario in which two LDAP user directories identify the last name of users with different underlying schema.

Note: Advanced use cases are shown in the Apply Attribute Mapping section. The advanced use cases detail how you use different attribute mapping types to identify the same user attribute across different directory types.

The following illustration details how two *alias* attribute mappings can define a universal schema and create a common view of the same user information.



1. Two user directories identify the last name of users differently:
   - Directory A identifies the last name of users with sn.
   - Directory B identifies the last name of users with lastname.

   This results in two different representations and views of the same user information.

2. LastName is the common name or *alias* that is mapped to the underlying directory schema:
   - LastName is mapped to sn in Directory A.
   - LastName is mapped to lastname in Directory B.
3. LastName results in a common view of the same user information. You can reference LastName when defining policies, expressions, or other objects that require the last name of users, without concern for the directory-specific schema, because the directories are operationally identical. SOA Security Manager determines that LastName is sn in Directory A and lastname in Directory B.

More information:

Named Expressions (see page 169)

Create an Alias Attribute Mapping

You can map a common name or *alias* to the user attribute name specified by the user directory's underlying schema. The user attribute's data type can be string, number, or Boolean. You can use an alias attribute mapping to read or change a user attribute value in a user directory. User attribute mappings are directory-specific.

To create an alias attribute mapping

1. Navigate to the User Directory: *Name* pane.
2. Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create Attribute Mapping: *Name* pane opens.
4. Type the common name and a description of the attribute mapping in the fields on the General group box.

   Note: Common names must conform to the same rules as user attribute names.
5. Select Alias on the Properties group box.
6. Type the definition in the Definition box on the Properties group box.

   Example:

   Name: FirstName

   Definition: givenname

   Description: The common name FirstName is mapped to the user attribute name givenname.

7. (Optional) Select Disabled to disable this attribute mapping.
8. Click OK.

   The Create Attribute Mapping task is submitted for processing, and the new attribute mapping is added to the list on the Attribute Mapping List group box.

Group Name

A *group name* attribute lets you map a common name to the underlying directory schema that identifies whether a user belongs to a specific group. Group name attribute mappings map common names to user attributes that can be read or changed. This type of access is called read/write (RW).

Group name attribute mappings result in a Boolean value. If the user is a member of the specified group, the mapping results in a TRUE value. Otherwise, the result is FALSE.

Group name attribute mapping and user classes, one type of named expression, are similar in the following ways:

- Both are used to determine group membership.
- Both result in a Boolean value.

Group name attribute mapping differs from user classes as follows:

- *A* group name attribute mapping is defined for particular user directories. User classes are global and can be applied to any user in any user directory.
- A group name attribute mapping can change a user's membership in a group in a user directory. User classes are Boolean expressions and cannot be changed.
- User classes must begin with the "at" sign (@). A group name mapping does not.

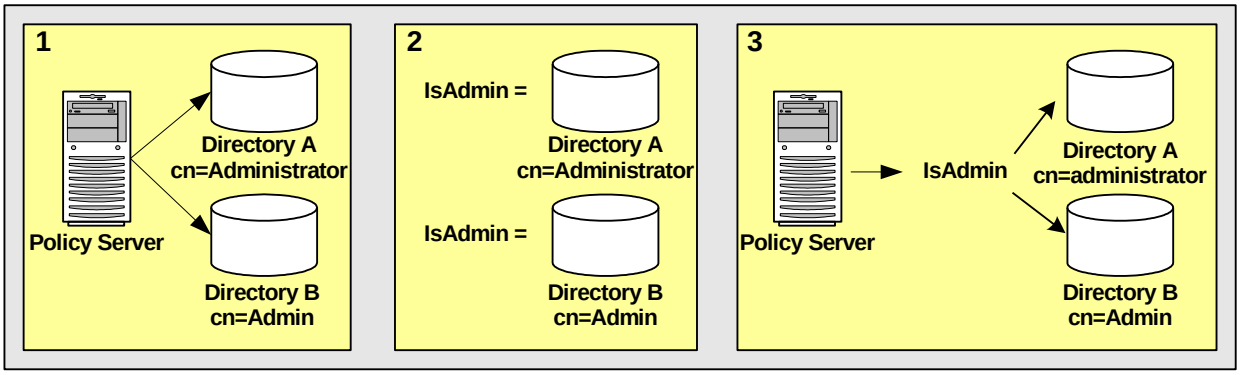
Note: For more information about user classes, see the Named Expressions.

Group Name Use Case

This use case represents a basic scenario in which two LDAP user directories use different underlying schema to identify users that belong to an Administrator group.

Note: Advanced use cases are shown in the Apply Attribute Mapping section. The advanced use cases detail how you use different attribute mapping types to identify the same user attribute across different directory types.

The following illustration details how two *group name* attribute mappings can define a universal schema and create a common view of the same user information.



1. Two user directories identify membership to the administrator group differently:

   - Directory A identifies membership in the administrator group as cn=Administrators,ou=groups,o=acme.com.
   - Directory B identifies membership in the administrator group as cn=Admin,ou=groups,o=acme.com.

   This results in two different representations and views of the same user information.

2. IsAdmin is the common name that is mapped to the underlying directory schema:

   - IsAdmin is mapped to cn=Administrators,ou=groups,o=acme.com in Directory A.
   - IsAdmin is mapped to cn=Admin,ou=group,o=acme.com in Directory B.

3. IsAdmin results in a common view of the administrator group. You can reference IsAdmin when defining policies, expressions, or other objects that apply to the Administrator group, without concern for the directory-specific schema, because the directories are operationally identical. SOA Security Manager determines that IsAdmin is cn=Administrators,ou=groups,o=acme.com in Directory A and cn=Admin,ou=group,o=acme.com in Directory B.

More information:

Named Expressions (see page 169)

Create a Group Name Attribute Mapping

You define a group name attribute to map a common name to the underlying directory schema that identifies whether a user belongs to a specific group. The result of a group name attribute mapping is a Boolean value. You can use a group name attribute mapping to read or change a user's group name in a user directory. User attribute mappings are directory-specific.

Note: For information about creating a global object that returns group membership, see user classes in the Named Expression chapter in this guide.

To Create a User Attribute Mapping of Type Group

1. Navigate to the User Directory: *Name* pane.
2. Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create Attribute Mapping: *Name* pane opens.
4. Type the common name and a description of the attribute mapping in the fields on the General group box.

   Note: Common names must conform to the same rules as user attribute names.
5. Select Group on the Properties group box.
6. Type the definition in the Definition box on the Properties group box.

   Example:

   Name: IsAdmin

   Definition: cn=administrators,ou=groups,o=acme.com

   Description: The common name IsAdmin is mapped to the group name cn=administrators,ou=groups,o=acme.com. If the user is a member of cn=administrators,ou=groups,o=acme.com, IsAdmin is TRUE.
7. (Optional) Select Disabled to disable this attribute mapping.
8. Click OK.

   The Create Attribute Mapping task is submitted for processing, and the new attribute mapping is added to the list on the Attribute Mapping List group box.

Mask

Mask attribute mapping lets you map a common name to the name used by the underlying directory schema to identify a user attribute that stores a bit pattern. *Mask* attribute mappings map common names to user attributes that can be read or changed. This type of access is called read/write (RW).

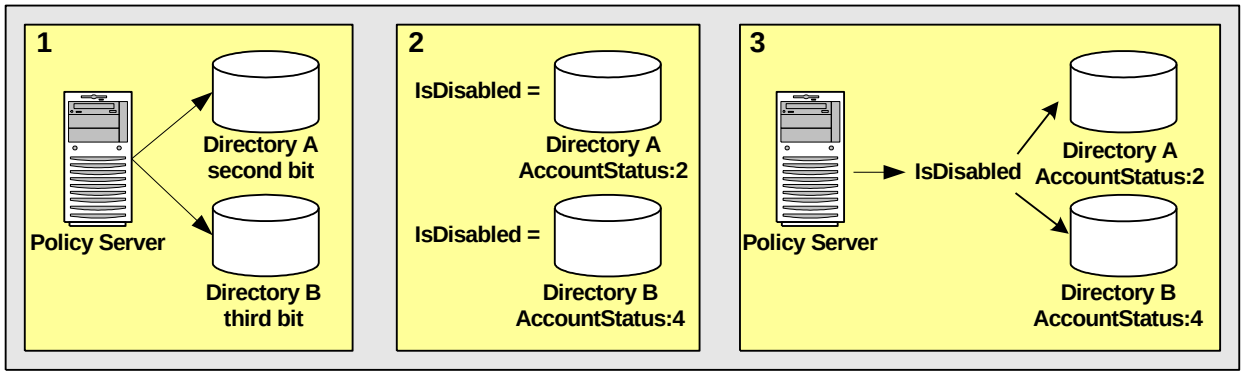
Mask attribute mappings result in a Boolean value. If the bit pattern in the user directory matches the specified mask, the mapping results in a TRUE value. Otherwise, the result is FALSE.

Mask Use Case

This use case represents a basic scenario in which two Active Directory user directories identify disabled user accounts with different underlying schema.

Note: Advanced use cases are shown in the Apply Attribute Mapping section. The advanced use cases detail how you use different attribute mapping types to identify the same user attribute across different directory types.

The following illustration details how two *mask* attribute mappings can define a universal schema and create a common view of the same user information.



1. Two user directories contain a user attribute named AccountStatus. AccountStatus stores user information in a bit pattern, where each bit is a flag.
   - In Directory A, the second bit flags a disabled account. When the second bit equals 1, the account is disabled.
   - In Directory B, the third bit flags a disabled account. When the third bit equals 1, the account is disabled.

   This results in two different representations and views of the same user information.

2. IsDisabled is the common name that is mapped to the underlying directory schema. In both directories, IsDisabled is mapped to AccountStatus.
   - In Directory A, SOA Security Manager uses the bit mask 2 (decimal) to determine whether the second bit of AccountStatus is set and the account is disabled.
   - In Directory B, SOA Security Manager uses the bit mask 4 (decimal) to determine whether the third bit of AccountStatus is set and the account is disabled.
3. IsDisabled results in a common view of disabled user accounts. You can reference IsDisabled when defining policies, expressions, or other objects that require the account status of users, without concern for the directory-specific schema, because the directories are operationally identical. SOA Security Manager determines that IsDisabled is AccountStatus:2 for Directory A and AccountStatus:4 for Directory B.

More information:

Named Expressions (see page 169)

Create a Mask Attribute Mapping

You can map a common name to a bit pattern whose user attribute name is specified by the user directory's underlying schema. The result of a mask attribute mapping is a Boolean value. You can use a mask mapping to read or change a user attribute value in a user directory. User attribute mappings are directory-specific.

To Create a User Attribute Mapping of Type Mask

1. Navigate to the User Directory: *Name* pane.
2. Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create Attribute Mapping: *Name* pane opens.
4. Type the common name and a description of the attribute mapping in the fields on the General group box.

   Note: Common names must conform to the same rules as user attribute names.
5. Select Mask on the Properties group box.

6. Type the definition in the Definition box on the Properties group box.

   Example:

   Name: IsDisabled

   Definition: AccountStatus:4

   Description:

   The common name IsDisabled is mapped to the user attribute name AccountStatus. AccountStatus stores one or more states in a bit pattern. For example, AccountStatus stores the account state in the third bit. When the account is disabled, the third bit is set to 1. Conversely, when the account is enabled, the third bit is set to 0.

   SOA Security Manager performs a bitwise AND operation on AccountStatus and the specified state or *mask*, which in this example is 4, to determine whether the account is disabled. If the account is disabled, IsDisabled is TRUE.

7. (Optional) Select Disabled to disable this attribute mapping.
8. Click OK.

   The Create Attribute Mapping task is submitted for processing, and the new attribute mapping is added to the list on the Attribute Mapping List group box.

Bit Masks in Mask Attribute Mapping

The purpose of a bit mask in mask attribute mapping is to test the value of one or more bits by masking the values of the other bits in a user attribute.

A mask attribute mapping is defined as follows:

```
user_attribute_name:bit_mask
```

For example, assume that the user attribute is named AccountStatus and that AccountStatus stores the states of the following three flags in a bit pattern, as follows:

Bit Pattern	Flag
00?	account disabled?
0?0	password expired?
?00	gold member?

When a bit equals one, the flag is TRUE, as follows:

Bit Pattern	Account Status
000 (0)	no flags are TRUE
001 (1)	account disabled
010 (2)	password expired
100 (4)	gold member
011 (3)	password expired, account disabled
101 (5)	gold member, account disabled
110 (6)	gold member, password expired
111 (7)	gold member, password expired, account disabled

Note: Equivalent decimal values are shown in parentheses.

Assume that you only want to test whether a user is a gold member. To test this bit, select the bit pattern that corresponds to a gold member as the bit mask or 100 (binary) and specify it as 4 (decimal). The resulting mask attribute mapping is defined as follows:

AccountStatus:4

SOA Security Manager performs a bitwise AND operation on AccountStatus and the bit mask and tests whether the result is equal to the bit mask. If they are equal, the value of the tested bit is one, and the flag is TRUE, as shown in the following table:

Account Status	Bit Mask	Result of Bitwise AND	Gold Member?
000 (0)	100 (4)	000 (0)	FALSE
001 (1)	100 (4)	000 (0)	FALSE
010 (2)	100 (4)	000 (0)	FALSE
011 (3)	100 (4)	000 (0)	FALSE
100 (4)	100 (4)	100 (4)	TRUE
101 (5)	100 (4)	100 (4)	TRUE
110 (6)	100 (4)	100 (4)	TRUE
111 (7)	100 (4)	100 (4)	TRUE

Note: Equivalent decimal values are shown in parentheses.

You can also use a bit mask to test the value of a bit set or more than one bit at a time. Assume that you want to know whether the account is disabled and the password has expired. To test these bits, specify a bit mask of 011 (binary) or 3 (decimal). The resulting mask attribute mapping is defined as follows:

```
AccountStatus:3
```

SOA Security Manager performs a bitwise AND operation on AccountStatus and the bit mask and tests whether the result is equal to the bit mask. If they are equal, the value of both tested bits is one, and both flags are TRUE, as shown in the following table:

Account Status	Bit Mask	Result of Bitwise AND	Both Flags Set?
000 (0)	011 (3)	000 (0)	FALSE
001 (1)	011 (3)	001 (1)	FALSE
010 (2)	011 (3)	010 (2)	FALSE
011 (3)	011 (3)	011 (3)	TRUE
100 (4)	011 (3)	000 (0)	FALSE
101 (5)	011 (3)	001 (1)	FALSE
110 (6)	011 (3)	010 (2)	FALSE
111 (7)	011 (3)	011 (3)	TRUE

Note: Equivalent decimal values are shown in parentheses.

Constant

Constant attribute mapping lets you map a common name to a value that is the same or *constant* for every user in a directory. Since *constant* attribute mappings map common names to constants, which are read only (R), they cannot be changed (except by a system administrator).

Constant attribute mappings can map to constants that have any of the following data types:

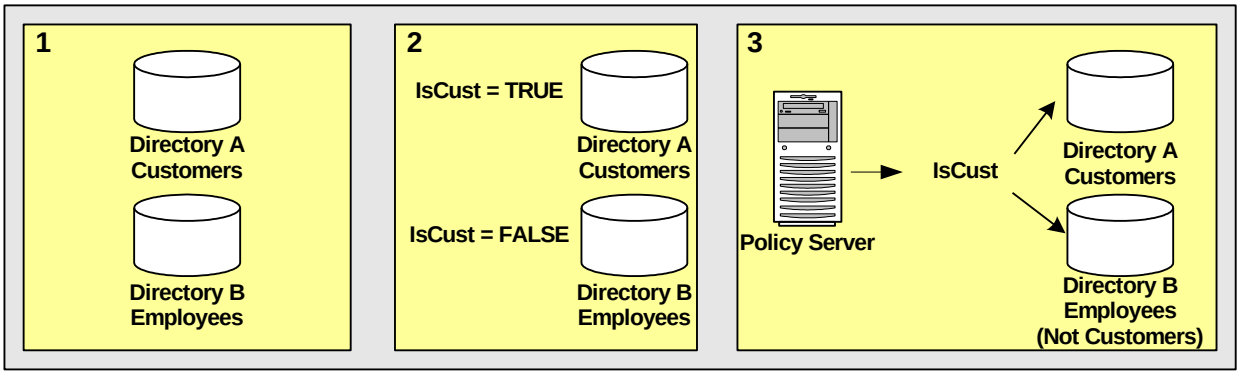
- string
- number
- Boolean

Constant Use Case

This use case represents a basic scenario in which one user directory only stores customers, while another user directory only stores employees.

Note: Advanced use cases are shown in the Apply Attribute Mapping section. The advanced use cases detail how you use different attribute mapping types to identify the same user attribute across different directory types.

The following illustration details how two *constant* attribute mappings can represent different values for different user directories.



1. Directory A only stores customers. Directory B only stores employees.
2. IsCust is the common name that is mapped to different values in different directories:
   - IsCust is mapped to TRUE in Directory A.
   - IsCust is mapped to FALSE in Directory B.
3. You can reference IsCust when defining policies, expressions, or other objects that must determine if a user is a customer, without attending to the particular directory in which the user is stored. SOA Security Manager determines that every user in Directory A is a customer, while every user in Directory B is not a customer.

More information:

Named Expressions (see page 169)

Create a Constant Attribute Mapping

You can map a common name to a constant value that conveys information about every user in a directory. The constant's data type can be string, number, or Boolean. You can use a constant attribute mapping to read a user attribute value that applies to every user in a user directory. User attribute mappings are directory-specific.

To create a constant attribute mapping

1. Navigate to the User Directory: *Name* pane.
2. Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create Attribute Mapping: *Name* pane opens.
4. Type the common name and a description of the attribute mapping in the fields on the General group box.

   Note: Common names must conform to the same rules as user attribute names.
5. Select Constant on the Properties group box.
6. Type the definition in the Definition box on the Properties group box.

   Example:

   Name: IsCustomer

   Definition: TRUE

   Description: The common name IsCustomer is mapped to the constant value TRUE. Because the user directory only stores customers, the user is always a customer, and IsCustomer is always mapped to TRUE.
7. (Optional) Select Disabled to disable this attribute mapping.
8. Click OK.

   The Create Attribute Mapping task is submitted for processing, and the new attribute mapping is added to the list on the Attribute Mapping List group box.

Expression

An *expression* attribute mapping lets you map a common name to an expression. The expression can contain one or more user attribute names specified by the user directory's underlying schema and must conform to the syntax rules of a SOA Security Manager expression.

Expression attribute mappings map common names to expressions that can be read, but not changed. This type of access is called read only (R). When evaluated, the expressions result in a string, number, or Boolean value.

Expression attribute mapping and virtual user attributes, one type of named expression, are similar in the following ways:

- Both are SOA Security Manager expressions.
- As expressions, both are read only (R).
- As expressions, both result in one of the following data types:
  - string
  - number
  - Boolean

Expression attribute mapping differs from virtual user attributes as follows:

- An expression attribute mapping is defined for particular user directories. Virtual user attributes are global and can be applied to any user in any user directory.
- Virtual user attributes must begin with the pound sign (#). An expression mapping does not.

More information:

Named Expressions (see page 169)

Expression Use Case

This use case shows how you can use an expression attribute mapping to simplify references to multiple user attributes in one directory.

Given:

This use case represents a basic scenario in which a protected resource requires the sort name of users (last name,first name). The user directory does not uniquely reference this user attribute, but does store the last name of users as surname and the first name of users as givenname.

Solution:

Map a common name to an expression that creates the sort name using the user attribute names specified by the user directory's underlying schema.

- Name the mapping **SortName**.
- Define SortName as:

  {surname + "," + givenname}

Note: The expression must conform to the syntax rules of a SOA Security Manager expression.

Result:

You can reference SortName when defining policies, expressions, or other objects that require the sort name of users, without concern for the directory-specific schema. SOA Security Manager calculates the sort name using the expression.

Note: Advanced use cases are shown in the Apply Attribute Mapping section. The advanced use cases detail how you use different attribute mapping types to identify the same user attribute across different directory types.

More information:

Named Expressions (see page 169)
Expression Syntax Overview (see page 466)

Create an Expression Attribute Mapping

You can map a common name to an expression that references one or more user attribute names specified by the user directory's underlying schema. An expression attribute mapping's data type is string, number, or Boolean. You can use expression attribute mapping to read the result of an expression, but not to write a value to a user directory.

Note: User attribute mappings are directory-specific. To create a global object that is defined as an expression, create a named expression.

To create an expression attribute mapping

1. Navigate to the User Directory: *Name* pane.
2. Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create Attribute Mapping: *Name* pane opens.

4. Type the common name and a description of the attribute mapping in the fields on the General group box.

   Note: Common names must conform to the same rules as user attribute names.

5. Select Expression on the Properties group box.

   Note: Selecting Expression activates the Edit button. Clicking Edit opens the Expression Editor. The expression must conform to the syntax rules of a SOA Security Manager expression.

6. Type the definition in the Definition box on the Properties group box.

   Example:

   Name: SortName

   Definition: (surname + ',' + givenname)

   Description: The common name SortName is mapped to an expression that references the user attribute names surname and givenname.

7. (Optional) Select Disabled to disable this attribute mapping.

8. Click OK.

   The Create Attribute Mapping task is submitted for processing, and the new attribute mapping is added to the list on the Attribute Mapping List group box.

Apply User Attribute Mapping

Multiple user directories in a SOA Security Manager environment often store the same user attributes, but use different underlying schema to identify them. In this example, a retail clothing company uses two user directories of different types. Directory A is an internal LDAP user directory for employees only. Directory B is an ODBC user directory for customers only. Each user attribute mapping is specific to the user directory in which it is defined.

The following table details how Directory A and Directory B use different underlying schema to identify the same user information. The accompanying use cases explain how you can use different attribute mappings to define a universal schema that creates a common view of the same user information, thereby making the directories operationally identical from a SOA Security Manager perspective.

Attribute Description	Directory A (LDAP)	Directory B (ODBC)
The first name of users	givenname	u_first_name
The last name of users	surname	u_last_name

The sort name of users (last name, first name)	The user directory does not uniquely store the user attribute.	sort_name
Is the user a customer?	group:cn=customer,ou=groups,o=acme.com	Users are always customers
Is the user account disabled?	The user directory has an AccountStatus attribute, which is a set of flags. The second bit indicates a disabled account.	u_disabled

First Name Use Case

This use case shows how you can use two *alias* attribute mappings to represent the first name user attribute in Directory A and Directory B.

Given:

User Directory A identifies the first name of users with givenname. Directory B identifies the first name of users with u_first_name. This results in two different representations and views of the same user information.

Solution:

1. Create an a*lias* attribute mapping for Directory A.
   - Name the mapping **FirstName**.
   - Define FirstName as **givenname**.
2. Create an *alias* attribute mapping for Directory B.
   - Name the mapping **FirstName**.
   - Define FirstName as **u_first_name**.

Result:

FirstName results in a common view of the same user information. You can reference FirstName when defining policies, expressions, or other objects that require the first name of users, without concern for the directory-specific schema, because the directories are operationally identical. SOA Security Manager determines that the first name of users is identified by givenname when referencing users in Directory A and is identified by u_first_name when referencing users in Directory B.

Last Name Use Case

This use case shows how you can use two *alias* attribute mappings to represent the last name user attribute in Directory A and Directory B.

Given:

User Directory A identifies the last name of users with surname. Directory B identifies the last name of users with u_last_name. This results in two different representations and views of the same user information.

Solution:

1. Create an *alias* attribute mapping for Directory A.
   - Name the mapping **LastName**.
   - Define LastName as **surname**.
2. Create an a*lias* attribute mapping for Directory B.
   - Name the mapping **LastName**.
   - Define LastName as **u_last_name**.

Result:

You can reference LastName when defining policies, expressions, or other objects that require the last name of users, without concern for the directory-specific schema, because the directories are operationally identical. SOA Security Manager determines that the last name of users is identified by surname when referencing users in Directory A and is identified by u_last_name when referencing users in Directory B.

Sort Name Use Case

This use case shows how you can use a *calculated expression* attribute mapping and an *alias* attribute mapping to represent the sort name of a user in Directory A and Directory B.

Given:

1. Directory A does not uniquely identify a user's sort name. Directory A does store a user's first name as givenname and a user's last name as surname.
2. Directory B identifies a user's sort name with sort_name.

   This results in two different representations and views of the same user information.

Solution:

1. Create a *calculated expression* attribute mapping for Directory A:
   - Name the mapping **SortName**.
   - Define SortName as:

     ```
     (surname + "," + givenname)
     ```

     Note: The expression must conform to the syntax rules of a SOA Security Manager expression.
2. Create an *alias* attribute mapping for Directory B:
   - Name the mapping **SortName**.
   - Define SortName as **sort_name**.

Result:

You can reference SortName when defining policies, expressions, or other objects that require the sort name of users, without concern for the directory-specific schema, because the directories are operationally identical. SOA Security Manager determines that the sort name must be calculated when referencing users in Directory A and is sort_name when referencing users in Directory B.

Customer Use Case

This use case shows how you can use a g*roup membership* attribute mapping and a *constant* attribute mapping to identify customers in Directory A and Directory B.

Given:

1. Directory A stores employees, and because an employee of the company can also be a customer, Directory A identifies customers as those employees that belong to:

   ```
   cn=Customers,ou=Groups,o=acme.com
   ```
2. Directory B only stores customers. Directory B does not have a user attribute that identifies customers, because to store a user in Directory B implies that the user is a customer.

Solution:

1. Create a g*roup membership* attribute mapping for Directory A:
   - Name the mapping **IsCustomer**.
   - Define IsCustomer as:

     cn=Customers,ou=Groups,o=acme.com
2. Create a c*onstant* attribute mapping for Directory B:
   - Name the mapping **IsCustomer**.
   - Define IsCustomer as **TRUE**.

Result:

You can reference IsCustomer when defining policies, expressions, or other objects that must determine if the user is a customer, without concern for the directory-specific schema, because the directories are operationally identical. SOA Security Manager determines that a user is a customer if they belong to cn=Customers,ou=Groups,o=acme.com when referencing Directory A and that every user is a customer when referencing Directory B.

Account Status Use Case

This use case shows how you can use a *mask* attribute mapping and a c*alculated expression* attribute mapping to identify user accounts that are disabled in Directory A and Directory B.

Given:

1. Directory A identifies disabled accounts with a user attribute named AccountStatus, which is a set of flags. The second bit indicates a disabled account.
2. Directory B identifies disabled accounts with a user attribute named u_disabled. When u_disabled is equal to "y", the account is disabled. When u_disabled is equal to "n", the account is active.

Solution:

1. Create a *mask* attribute mapping for Directory A:
   - Name the mapping **IsDisabled**.
   - Define the mapping as **AccountStatus:2**, where:
     - The bit pattern is stored in AccountStatus.
     - The bit mask is 2 (decimal).

2. Create a c*alculated expression* attribute mapping for Directory B:
   - Name the mapping **IsDisabled**.
   - Define the mapping as:

     (u_disabled = "y")

     where u_disabled is a Boolean expression.

Result:

You can reference IsDisabled when defining policies, expressions, or other objects that must determine the account status of users, without concern for the directory-specific schema, because the directories are operationally identical. SOA Security Manager checks the bit pattern to determine if a user is disabled when referencing Directory A and performs the calculation to determine if a user is disabled when referencing Directory B.

Chapter 6: Directory Mapping

This section contains the following topics:

Directory Mapping Overview (see page 207)
Directory Mapping Requirements (see page 209)
Supported Directory Mappings (see page 209)
How to Configure an Authentication and Authorization Directory Mapping (see page 209)
Configure an AuthValidate Directory Mapping (see page 211)
Directory Mapping Examples (see page 211)
Directory Mapping and Responses (see page 214)

Directory Mapping Overview

SOA Security Manager assumes that a user will be authenticated and authorized against the same user directory. Although this default behavior is sufficient in many cases, SOA Security Manager also provides the ability to authenticate users against one directory, and authorize users against a separate directory. This feature is called directory mapping. It is especially useful when authentication information is stored in a central directory, but authorization information is distributed in separate user directories that are associated with particular network applications.

Note: Impersonation is not supported by directory mapping. The impersonatee, the user being impersonated, must be uniquely present in the authentication directories associated with the domain or the impersonation fails.

Mapping from an authentication directory to an authorization directory is a three-step process.

1. Set Up User Directory Connections

   Directory connections you want to specify as authentication or authorization directories in a mapping must be configured on the User Directory pane.

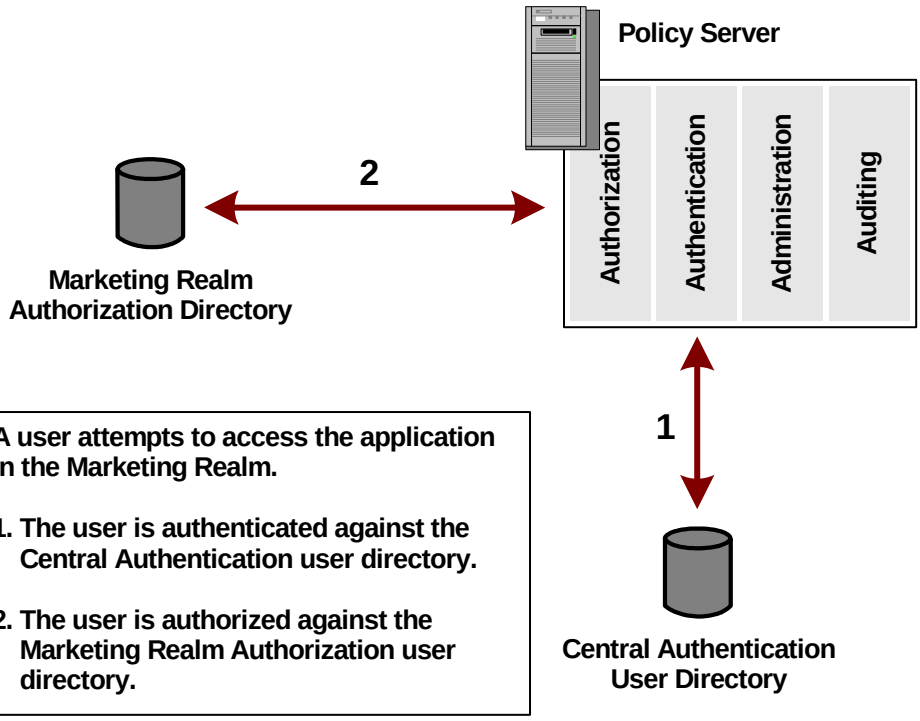
2. Configure a Directory Mapping

   The Policy Server uses directory mappings to locate authenticated users in separate authorization directories.

3. Assign a Directory Mapping to a Realm

   By associating a directory mapping with a specific realm, you can define the directory against which a user will be authorized for specific resources in a network.

   For example, in the following diagram, all of the users in a company are authenticated against a single central user directory, but the marketing organization has a separate user directory that contains authorization data for Marketing staff. Using the Policy Server, you can configure a directory mapping to the Marketing authorization user directory, then you can create a realm for the Marketing application that uses the authorization directory specified in the mapping. Whenever a user tries to access the Marketing application, the Policy Server authenticates the user against the central user directory, but authorizes the user against the Marketing user directory.



More information:

How to Configure a CA Directory User Directory Connection (see page 101)
Configure a Directory Mapping (see page 210)
Realms (see page 329)

Directory Mapping Requirements

Directory mapping requires that the user directory connections to the Policy Server must already exist for the authentication directory, as well as the authorization or validation directory.

Note: More information on creating user directory connections exists in User Directory Connections Overview.

Supported Directory Mappings

The following table describes supported types of directory mapping, and the method that can be used to map the authentication directory to the authorization or validation directory.

	Authorization Directory/Validation Directory		
Authentication Directory	**LDAP**	**Relational Database**	**WinNT**
LDAP	Identical DN Universal ID	Universal ID	N/A
AD	Identical DN Universal ID	Universal ID	N/A
Relational Database	Universal ID	Identical DN Universal ID	N/A
WinNT	Universal ID	Universal ID	Identical DN

How to Configure an Authentication and Authorization Directory Mapping

Configuring an authentication and authorization directory mapping is a two-step process:

1. Configure the Directory Mapping
2. Assign an Authorization Directory to a Realm

Configure a Directory Mapping

You can configure a directory mapping to authenticate users against one directory and authorize users against another directory.

To configure a directory mapping

1. Click Infrastructure, Directory.
2. Click Auth/Az Mapping, Create Directory Mapping.

   The Create Directory Mapping pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Select the authentication and authorization directories from the respective lists.
4. Select the Identical DN or Universal ID radio button.
5. Click Submit.

   The Create Directory Mapping task is submitted for processing.

More information:

Universal IDs (see page 168)

Assign an Authorization Directory to a Realm

You assign a directory mapping to a realm so the Policy Server may authenticate a user in one directory and authorize a user in another directory. The Policy Server uses the authorization directory specified in the realm to authorize users.

To assign a directory mapping to a realm

1. Open the realm to which you want to assign a directory mapping.
2. Select the user directory for which the realm should use to authorize an authenticated user from the Directory Mapping list.

   Note: The Default value indicates that there is no directory mapping; the authentication directory will be used as the authorization directory when a user attempts to access a resource in the realm. The list only contains user directories that have been configured as authorization directories in an existing directory mapping.
3. Click Submit.

   The Policy Server saves the directory mapping. Users that access the realm authenticate normally and authorize against the directory specified in the realm.

Configure an AuthValidate Directory Mapping

You can configure an AuthValidate directory mapping to authenticate users against one directory and validate users against another directory.

To configure an AuthValidate Directory Mapping

1. Click Infrastructure, Directory.
2. Click AuthValidate Mapping, Create AuthValidate Directory Mapping.

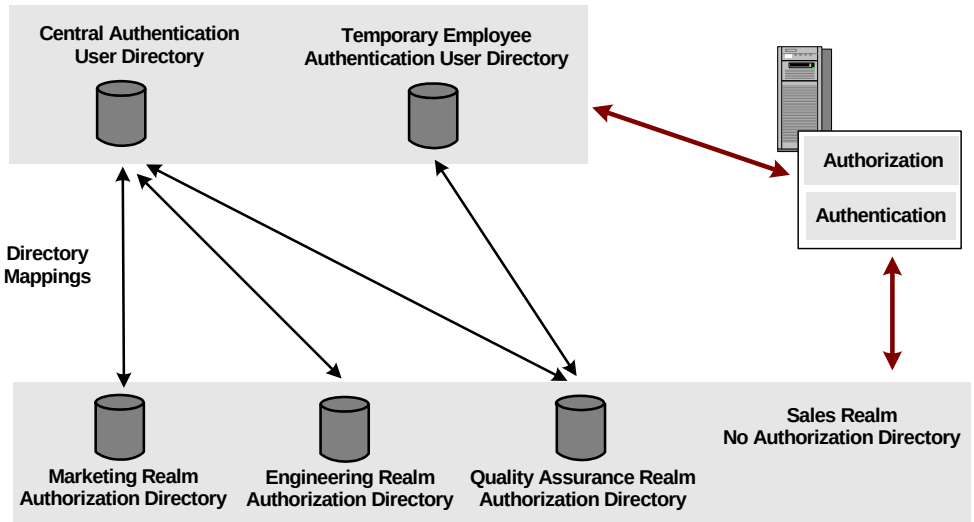
   The Create AuthValidate Directory Mapping pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Type the name of the directory that will be used to authenticate users in the Authentication Directory field.
4. Select the directory that will be used to validate users from the Validation Directory list.
5. Select the Identical DN or Universal ID radio button.
6. Click Submit.

   The Create AuthValidate Directory Mapping task is submitted for processing.

Directory Mapping Examples

Multiple directory mappings may be needed in order to correctly authenticate and authorize users that have access to network resources. The following figure illustrates a simple sample case where multiple directory mappings may be required.



In the example, there are three realms that have separate authorization user directories and one realm that does not have its own authorization user directory. User authentication information is maintained in two distinct authentication directories. The Policy Server uses one of the authentication directories based on where the user's information is stored and one of the authorization directories if the requested resource is in the Marketing, Engineering, or Quality Assurance realms.

More information:

Realms (see page 329)

Employee Accesses an Engineering Realm Resource

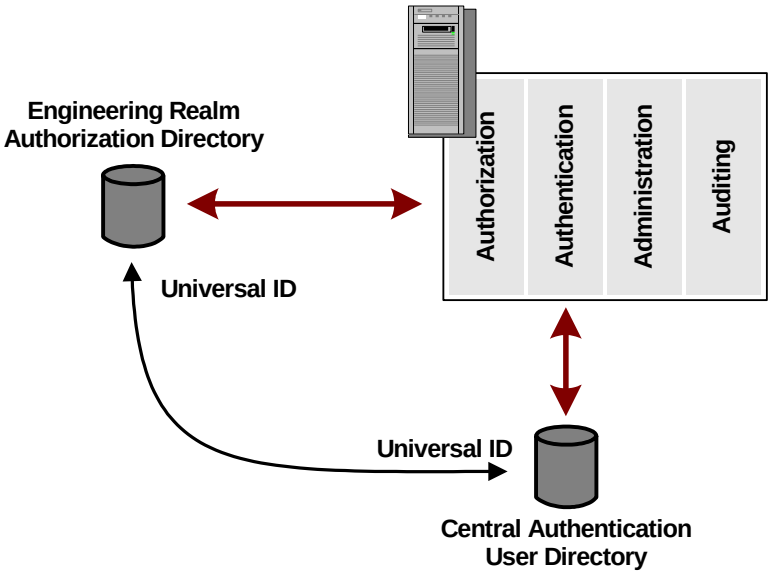
In the SOA Security Manager protected network described in the previous figure, a regular employee's authentication data resides in the Central Authentication user directory, and the employee attempts to access a resource in the Engineering Realm. When the employee is properly authenticated, the Policy Server recognizes that the Engineering realm uses its own authorization directory. The Policy Server looks for the directory mapping between the Central Authentication user directory and the Engineering Realm authorization user directory, then maps the users identity to the authorization directory. Once this is done, the Policy Server can verify if the employee has access to the requested realm.

Temporary Employee Accesses a Quality Assurance Realm Resource

In the SOA Security Manager protected network described in the previous figure, a temporary employee's authentication data resides in the Temporary Employee Authentication user directory, and the employee attempts to access a resource in the Sales Realm. The Sales Realm is not associated with its own authorization directory. SOA Security Manager authenticates the employee in the Temporary Employee Authentication directory, then checks to see if a directory mapping is set up. Since the Sales Realm does not have its own authorization directory, SOA Security Manager attempts to authorize the employee using information in the same directory where the employee was authenticated.

Directory Mapping by Universal ID

In the SOA Security Manager protected network described in the previous figure, an employee's authentication data resides in the Central Authentication user directory, and the employee attempts to access a resource in the Engineering Realm. When the employee is properly authenticated, the Policy Server uses a directory mapping to find the employee in the Engineering Authorization directory, based on the employee's Universal ID (see the following figure).



In the diagram above, assume the directory mapping uses a Universal ID. The Policy Server uses the attribute in the authentication directory that is defined as the universal ID to find a matching universal ID in the authorization directory. Once the universal ID is located in the authorization directory, the SOA Security Manager can finish processing policies to determine whether or not the user can access a protected resource.

Directory Mapping Case Sensitivity

Case-sensitive directories, such as an Oracle database, treat the values "ROBIN" and "robin" as two different usernames. Other directories, such as an LDAP directory, are not case-sensitive and treat the values "Robin", "ROBIN", "robin", and "RobIn" as the same username. This can be a problem when a user is authenticated using a directory that is not case-sensitive, but authorized using a directory that is case-sensitive.

When authentication fails because the authentication directory is case-sensitive, the user can recover by reentering the username in the format required by the directory. If the directory requires the username to be in the format "Name", for example, the user can reenter the name correctly as "Robin". When authorization fails because the authorization directory is case-sensitive, however, the Policy Server has no way to recover.

When the authorization directory is case-sensitive, you can change the format of the authenticated username, so that it matches the format required by the authorization directory. If the authenticated username is "RoBiN", but the authorization directory requires the username to be in the format "Name", you can first change "RoBiN" to "Robin" and then authorize the user.

Directory Mapping and Responses

SOA Security Manager allows you to configure responses that collect the information in user directory attributes. If you use directory mappings, you must consider the effects that the mappings will have on some responses. For example, the directory mapping described in the diagram above would retrieve values from the Central Employee Authentication directory for an OnAuth event and from the Engineering Realm Authorization directory for an OnAccess event. For a description of events associated with responses, see Responses and Response Groups (see page 351).

Chapter 7: Authentication Schemes

This section contains the following topics:

Authentication Scheme Overview (see page 215)
Authentication Scheme Processing (see page 215)
XML Document Credential Collector Authentication (see page 216)
XML Digital Signature Authentication (see page 230)
SAML Session Ticket Authentication (see page 233)
WS-Security Authentication (see page 243)
Certificate Mapping (see page 269)

Authentication Scheme Overview

SOA Security Manager authentication schemes provide a way to collect credentials from an XML message request sent to a protected web service and determine the identity of the user represented by those credentials.

Authentication schemes must be configured using the Administrative UI. During authentication, SOA Agents communicate with the Policy Server to determine the proper credentials that must be retrieved from the request message.

This chapter discusses general information for working with authentication schemes in the Administrative UI, then provides separate sections that explain how to configure each supported scheme using authentication scheme templates. These templates provide the Policy Server with most of the information it needs to process a scheme. An administrator must complete the configuration of an authentication scheme by supplying implementation specific information, such as server IP addresses, or shared secrets required to initialize a scheme.

Authentication Scheme Processing

When a request message is sent to a protected web service resource, the Policy Server uses the authentication scheme associated with a web service resource to determine how to identify the requesting user. The authentication scheme specifies the credentials that must be supplied with the request for authentication, as well as the method used by the Policy Server to validate the identity of the user represented by those credentials.

You can use the Administrative UI to configure authentication schemes and assign the schemes to application components.

XML Document Credential Collector Authentication

The XML Document Credential Collector (XML DCC) authentication scheme is a document-based authentication scheme. This means that information in the XML document is collected and used by the Policy Server to authenticate web service consumers. Web service consumers include business applications, business partners, or individual users.

To implement the XML DCC scheme, the web service provider gives the XML schema to web service consumers who request web services so that the web service consumers send XML documents that comply with the schema. This enables the Policy Server to map the credentials to entries in the user store.

When a web service consumer requests a web service, the SOA Agent extracts the information from the XML document and passes the credentials to the Policy Server. For the Policy Server to verify the credentials, you must map attributes in the XML document to fields in the Policy Server's user store using the Policy Server User Interface or by defining XPath expressions. For example, you may map a XML document attribute called "customer" to the user name field in the user store and a document attribute called "customer ID" to the PIN number field. You also specify whether these attributes are located within the message header or body.

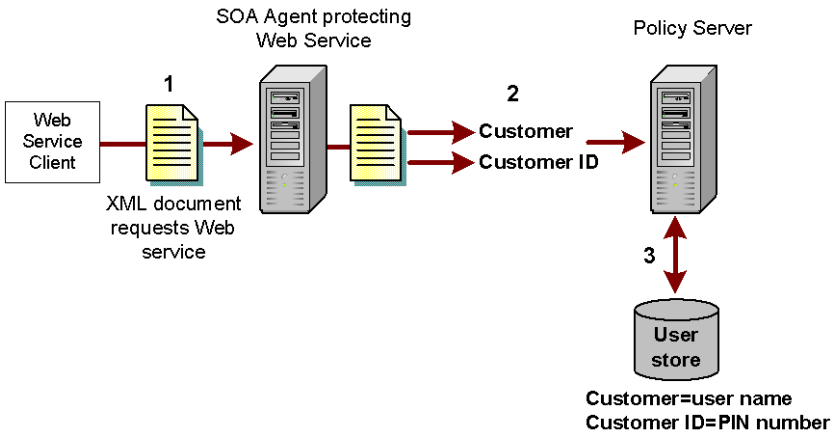
Optionally, you can specify an XPath function (count, div, index, mod, sum) that you want to apply to the mapping. The Function option lets you create more complex mappings by processing functions that further evaluate the XML document. For more information about these functions, navigate to the XPath specification at http://www.w3.org.

Note: The Policy Server assumes that all the authentication data in the XML message is in clear text. Therefore, we recommend that you configure this authentication scheme over an SSL connection to ensure security.

How Data Flows During XML DCC Authentication

The XML DCC authentication scheme collects information from an incoming XML document and passes it to the Policy Server to authenticate web service consumers

The following illustration shows the flow of the XML DCC authentication scheme.



1. The client sends a XML document requesting access to a web service.
2. The SOA Agent extracts credentials from the document and passes them to the Policy Server.
3. The Policy Server maps the credentials to fields in the user store to authenticate the client.

Configure the XML DCC Authentication Scheme

To obtain authentication information from an incoming XML document, you configure the XML DCC authentication scheme.

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click SOA Authentication Scheme, Create SOA Authentication Scheme.

   The Create Authentication Scheme pane opens.

3. Click OK.

   Authentication scheme settings open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

4. Enter a name and a description for the scheme in the General group box.
5. Select XML Document Credential Collector from the Authentication Scheme Type list.
6. Specify a protection level.
7. Configure XML DCC Scheme field mappings in the Scheme Setup group box.
8. Select the Require Secure Transport Layer check box to require that authentication only take place over an SSL connection.

   Important! The Policy Server expects the information in the XML document to be in clear text. To enforce security, we recommend that you configure this authentication scheme over an SSL connection.

9. Click Submit.

   The authentication scheme is saved and may be assigned to application components (realms).

More information:

XML DCC Scheme Mappings (see page 218)
Configure the Required "user" Mapping (see page 219)
Configure Additional Mappings (see page 220)

XML DCC Scheme Mappings

To create XML DCC mappings in the Policy Server User Interface, map a user store field name to an XPath string that identifies an element of an XML document. Create these field mappings by browsing a specific XML schema file (.xsd or .dtd) or by entering the XPath string directly.

XPath is a query language for XML that lets you find elements and attributes in an XML document. Because XML documents have a tree structure, an XPath expression uses a path notation to search through the XML document, such as /to/billto/customerID. The XPath string is relative to the document header or body.

The XML DCC authentication scheme *requires* only one mapped field—named "User"—to identify the XML document element that will be used to look up the user to be authenticated in the user store. To meet this requirement, the Field Mapping dialog forces the first field mapping that you create to be named "User".

The XPath string to which the User field is mapped identifies where in the incoming XML document that the value for the User field should be obtained. At run time, when an XML document is processed, the XPath string associated with the User field is used to locate the element in the document that the Policy Server should use to look up the user in the user store. The Policy Server uses this value with the configured User Directory attributes to determine the LDAP Universal ID that will be used to look up the user. In the case of other directory types such as ODBC, specific SQL queries are set up to look up users. In any case, it is the value of the user obtained from the XML DCC authentication scheme that is passed to these directory specific lookup mechanisms to find the user.

The only other specific field mapping name is "Password." To look up users by username/password, in addition to the User mapping, you must configure a second mapping named "Password." Again, the XPath expression from which the value of the Password field is obtained is dependent on the schema.

Other fields may be defined and user directories may be configured to make use of these fields.

Configure the Required "user" Mapping

When you create an XML DCC authentication scheme, a *required* "user" mapping, which is used to map the user name field in the user store, is created for you. Before you configure any further mappings, you must map this value to a field in the XML document.

Two methods for creating mappings are as follows:

- Open a schema and select elements from it that you want to map to the user store. This method is easier because SOA Security Manager builds the XPath query for you as you select fields for mappings.
- Use the Advanced XPath query option to build more complex XPath queries manually.

To configure XML DCC field mappings

1. Locate the Scheme Setup group box on the Create Authentication Scheme pane and click the Edit button beside the "user" field mapping entry.

   Field mapping settings open.

2. Do one of the following:
   - Type an XPATH query defining the mapping for "user" in the XPath field.
   - Load a schema (.xsd) file and select the element to map to "user" by browsing using the following procedure:
     a. Unset the Advance XPath query option.
     b. If the schema file you require is remote (for instance, if it is typically accessed over HTTP using its URL), download it to a local drive.
     c. Click Browse and navigate to a schema file in the File Upload dialog that appears.
     d. Click Upload XSD File.

        The schema is uploaded.
     e. Select the schema element that you want to map to the 'user" field name in the Select a node group box.

        The Select a node group box displays the selected schema using a standard tree-style hierarchical view. Click the plus sign (+) next to an element to expand it. Click the minus sign (-) beside an expanded element to contract it. Elements marked with an asterisk (*) are repeatable within the XML document (that is, incoming XML documents may contain multiple instances of that element).
3. (Optional) Specify the XPath function (count, div, index, mod, sum) that you want to apply to the mapping by choosing it from the Function drop-down list.

   The Function option lets you create more complex mappings by processing functions that further evaluate the XML document. For more information about these functions, navigate to the XPath specification at http://www.w3.org.
4. Specify whether the mapped information is located relative to the message body or message header by selecting the Message Body or Message Header option button.

   This defines the root of the XML document and tells XPath where to search for the relevant information. If the document has multiple headers, XPath uses the value of the first header that resolves.
5. Click OK to save your changes and return to the Create Authentication Scheme pane.

Configure Additional Mappings

You can define any number of field-to-document mappings for the XML DCC authentication scheme in addition to the required "user" mapping used to map the user name field in the user store.

Two methods for creating mappings are as follows:

- Open a schema and select elements from it that you want to map to the user store. This method is easier because SOA Security Manager builds the XPath query for you as you select fields to map.
- Use the Advanced XPath query option to build more complex XPath queries manually.

To configure XML DCC field mappings

1. Locate the Scheme Setup group box on the Create Authentication Scheme pane and click Add.

   Field mapping settings open.

2. Enter the name of a field from the user store, such as "email" in the Name field. This is the field to which you are mapping the XML element.

   Note: This name must match an entry in the user store; it is not case sensitive.

3. Do one of the following:

   – Type an XPATH query defining the mapping in the XPath field.

   – Load a schema (.xsd) file and select the element to map to the specified field name by browsing using the following procedure:

      a. Unset the Advance XPath query option.

      b. Click Browse and navigate to a schema file in the File Upload dialog that appears.

      c. Click Upload XSD File.

         The schema is uploaded.

      d. Select the schema element that you want to map to the specified field name in the Select a node group box.

         The Select a node group box displays the selected schema using a standard tree-style hierarchical view. Click the plus sign (+) next to an element to expand it. Click the minus sign (-) beside an expanded element to contract it. Elements marked with an asterisk (*) are repeatable within the XML document (that is, incoming XML documents may contain multiple instances of that element).

   Note: A loaded schema is not persistent; even when creating multiple mapping from the same schema file, you must reload the schema for each mapping.

4. (Optional) Specify the XPath function (count, div, index, mod, sum) that you want to apply to the mapping by choosing it from the Function drop-down list.

   The Function option lets you create more complex mappings by processing functions that further evaluate the XML document. For more information about these functions, navigate to the XPath specification at http://www.w3.org.

5. Specify whether the mapped information is located relative to the message body or message header by selecting the Message Body or Message Header option button.

   This defines the root of the XML document and tells XPath where to search for the relevant information. If the document has multiple headers, XPath uses the value of the first header that resolves.

6. Click OK to save your changes and return to the Create Authentication Scheme pane.

XML DCC XPath Mapping Examples

The following examples show XPath expressions to perform complex mappings.

Example Namespace-aware XPath Query

In the following XML file, the username and password are in the SOAP body, and the first element below the body is prefixed by a namespace. It would not therefore be possible to obtain these elements using the schema browsing method.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <env:Header>
 </env:Header>
 <env:Body>
  <n1:sayHello testnum="purchaseOrder11c" xmlns:n1="http://www.xyz.com/examples/Trader">
   <BillingInformation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\data\CredentialHeader.xsd">
            <CustomerCredentials>
                  <username>Robm</username>
                  <password>password</password>
                  <PIN>String</PIN>
            </CustomerCredentials>
   </BillingInformation>
  </n1:sayHello>
 </env:Body>
</env:Envelope>
```

To obtain the username, specify the following XPath query:

```
/*[local-name()='sayHello' and
namespace-uri()='http://www.xyz.com/examples/Trader']/BillingInformation/CustomerCredentials/username
```

To obtain the password, specify the following XPath query:

```
/*[local-name()='sayHello' and
namespace-uri()='http://www.xyz.com/examples/Trader']/BillingInformation/CustomerCredentials/password
```

Example XPath Query to Obtain Credentials From Embedded XML Documents

Sometimes, the required credentials are in a SOAP body payload, but the XML is screened from the parser either by a CDATA section or by use of replacement of angle brackets by entity references.

The following XPath queries will work for either CDATA or entity-reference screened XML.

XPath query for username

The following XPath query can be used to obtain a username mapping from the CDATA section or from entity-reference screened XML

```
substring-before(substring-after(/*[local-name()='sayHello' and
namespace-uri()='http://www.bea.com/examples/Trader']/text(),'<username>'),'</username>')
```

XPath query for password

The following XPath query can be used to obtain a password mapping from the CDATA section or from entity-reference screened XML

```
substring-before(substring-after(/*[local-name()='sayHello' and
namespace-uri()='http://www.bea.com/examples/Trader']/text(),'<password>'),'</password>')
```

Sample document containing credentials in CDATA section

The following sample XML document shows username and password credentials screened by a CDATA section.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<env:Header>
<BillingInformation>
        <CustomerCredentials>
                <username>Robm</username>
                <password>password</password>
        </CustomerCredentials>
</BillingInformation>
</env:Header>
<env:Body>
<n1:sayHello testnum="purchOrder05-cdata" xmlns:n1="http://www.bea.com/examples/Trader">

<![CDATA[<!--Sample XML file generated by XMLSpy v2005 rel. 3 U (http://www.altova.com)-->

<BillingInformation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\data\CredentialHeader.xsd">
```

```
        <CustomerCredentials>
                <username>Robm</username>
                <password>password</password>
                <PIN>String</PIN>
        </CustomerCredentials>

</BillingInformation>
]]>
</n1:sayHello>
</env:Body>
</env:Envelope>
```

Sample document containing credentials in entity-referenced screened XML:

The following sample XML document shows username and password credentials screened by the use of replacement of angle brackets by entity references.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<env:Header>
<BillingInformation>
        <CustomerCredentials>
                <username>Robm</username>
                <password>password</password>
        </CustomerCredentials>
</BillingInformation>
</env:Header>
<env:Body>
<n1:sayHello testnum="purchaseOrder04" xmlns:n1="http://www.bea.com/examples/Trader">

&lt;!--Sample XML file generated by XMLSpy v2005 rel. 3 U (http://www.altova.com)--&gt;

&lt;BillingInformation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\data\CredentialHeader.xsd"&gt;

        &lt;CustomerCredentials&gt;
                &lt;username&gt;Robm&lt;/username&gt;
                &lt;password&gt;password&lt;/password&gt;
                &lt;PIN&gt;String&lt;/PIN&gt;
        &lt;/CustomerCredentials&gt;
&lt;/BillingInformation&gt;

</n1:sayHello>
</env:Body>
</env:Envelope>
```

Example XPath Query to Obtain Credentials with a Default Namespace for all Elements

In the following XML file, the sayHello element has a default namespace specified by xmlns="http://www.xyz.com/examples/Trader".

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <env:Header>
 </env:Header>
 <env:Body>
  <sayHello testnum="purchaseOrder11c" xmlns="http://www.xyz.com/examples/Trader">
   <BillingInformation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\data\CredentialHeader.xsd">
            <CustomerCredentials>
                  <username>Robm</username>
                  <password>password</password>
                  <PIN>String</PIN>
            </CustomerCredentials>
   </BillingInformation>
  </sayHello>
 </env:Body>
</env:Envelope>
```

This following XPath query searches for the element "username" with the namespace "http://www.xyz.com/examples/Trader" anywhere in the document:

```
/*[local-name()='username' and namespace-uri()='http://www.xyz.com/examples/Trader']
```

Example XPath Query that Explicitly Specifies the Namespace Prefix

To extract the username and password (without namespace prefix) from a SOAP message, you can use an XPath query with an explicit tag including the namespace prefix and colon (:) as a simple text string.

For example, you could use the following XPath query to extract the username and password (without namespace prefix) from the sample SOAP message

Example XPath query with explicit tag

This XPath query could be used to extract the username and password (without namespace prefix) from the sample SOAP message that follows.

```
//*[name()='wsu:dccuser']
//*[name()='wsu:dccpwd']
```

Sample SOAP message

The preceding XPath query could be used to extract the username and password (without namespace prefix) from this sample SOAP message.

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
<soap:Header>
 <wsa:Action>http://example.com/services/XMLProcess_WebReq_portWebReq/Operation_1</wsa:Action>
 <wsa:MessageID>urn:uuid:e0b940b0-7d44-4e1e-b391-2e65c5b1de3f</wsa:MessageID>
 <wsa:ReplyTo>
  <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:Address>
 </wsa:ReplyTo>
 <wsa:To>http://ex01/XMLProcess_Proxy/XMLProcess_WebReq_portWebReq.asmx</wsa:To>
 <wsse:Security>
  <wsu:Timestamp wsu:Id="Timestamp-cedeb96e-9b12-45e6-bdf1-6e6c323a24cb">
   <wsu:Created>2006-04-12T18:31:33Z</wsu:Created>
   <wsu:Expires>2006-04-12T18:36:33Z</wsu:Expires>
   <wsu:dccuser>catest1</wsu:dccuser>
   <wsu:dccpwd>msimsi</wsu:dccpwd>
  </wsu:Timestamp>
 </wsse:Security>
</soap:Header>
<soap:Body>
 <Operation_1 xmlns="http://example.com/services">
  <XMLClaim MemberID="123456789" SubscriberID="987654321" TimeStamp="20060412 13:31:32:952"
TranNumber="270" ControlID="1" xmlns="http://Example.Claim_XML" />
 </Operation_1>
```

```
 </soap:Body>
</soap:Envelope>
```

Example XPath Query With Namespace and Element-by-Element Navigation

Use this XPath query, with namespace and element-by-element navigation:

```
/*[local-name()='Security' and
namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'][1]/*[local-name()='Timestamp' and
namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'][1]/*[local-name()='dccuser' and
namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'][1]

/*[local-name()='Security' and
namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'][1]/*[local-name()='Timestamp' and
namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'][1]/*[local-name()='dccpwd' and
namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'][1]

/*[local-name()='Security' and
      namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'][1]
/*[local-name()='Timestamp' and
      namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'][1]
/*[local-name()='dccuser' and
      namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'][1]
```

To extract the username and password from the following SOAP message:

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
<soap:Header>
 <wsa:Action>http://example.com/services/XMLProcess_WebReq_portWebReq/Operation_1</wsa:Action>
 <wsa:MessageID>urn:uuid:e0b940b0-7d44-4e1e-b391-2e65c5b1de3f</wsa:MessageID>
 <wsa:ReplyTo>
  <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:Address>
 </wsa:ReplyTo>
 <wsa:To>http://ex01/XMLProcess_Proxy/XMLProcess_WebReq_portWebReq.asmx</wsa:To>
 <wsse:Security>
  <wsu:Timestamp wsu:Id="Timestamp-cedeb96e-9b12-45e6-bdf1-6e6c323a24cb">
   <wsu:Created>2006-04-12T18:31:33Z</wsu:Created>
   <wsu:Expires>2006-04-12T18:36:33Z</wsu:Expires>
   <wsu:dccuser>catest1</wsu:dccuser>
   <wsu:dccpwd>msimsi</wsu:dccpwd>
  </wsu:Timestamp>
 </wsse:Security>
</soap:Header>
```

```
<soap:Body>
  <Operation_1 xmlns="http://example.com/services">
   <XMLClaim MemberID="123456789" SubscriberID="987654321" TimeStamp="20060412 13:31:32:952"
TranNumber="270" ControlID="1" xmlns="http://Example.Claim_XML" />
  </Operation_1>
 </soap:Body>
</soap:Envelope>
```

XML Digital Signature Authentication

The XML Digital Signature (XML-DSIG) authentication scheme protects web services by validating XML documents digitally signed with valid X.509 certificates.

How XML Digital Signature Authentication Works

The XML Digital Signature authentication scheme validates XML documents digitally signed with valid X.509 certificates.

The authentication process works as follows:

1. The web service consumer submits a signed XML document requesting a service. This document includes the required information, including the certificate and the public key.
2. The SOA Agent passes the URL of the requested resource to the Policy Server. The Policy Server informs the SOA Agent which credentials need to be collected. The SOA Agent then does the following:
   a. Verifies the signature and checks that the document is not altered.

      The SOA Agent detects two types of signatures:

      - **Enveloping signature**—A signature (defined in a <signature> element) that wraps an entire document. This would be typical of a raw XML document that does not use an enveloping technology, such as SOAP.
      - **Enveloped signature**—A signature embedded in an enveloped document. SOA Security Manager detects only enveloped signatures included in the SOAP headers. If multiple SOAP headers exist, the Agent uses only the first header it detects.

b. Checks that the required elements configured as part of the authentication scheme are included with the signature.

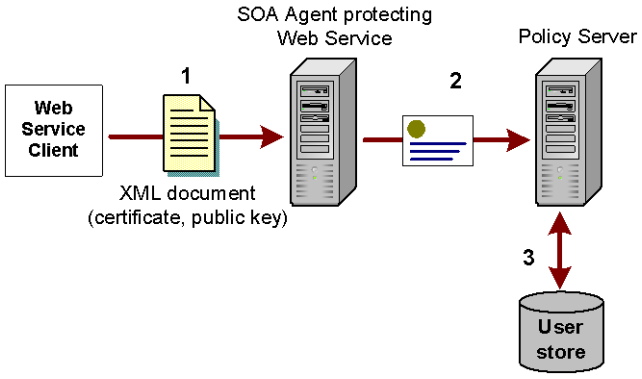
c. Takes the certificate and passes it to the Policy Server.

3. The Policy Server validates the certificate to ensure that it trusts the client by checking that the web service consumer has a valid entry in the user store. This is done using Policy Server's certificate mapping feature.

   To use an X.509 certificate to identify a user, the Policy Server uses a certificate mapping to compare a client's certificate with the information in the user directory. A certificate mapping defines how data in the certificate is mapped to form a user Distinguished Name (DN), which the Policy Server uses to authenticate the client.

If the web service consumer identified in the certificate is a valid user, the authentication process is complete.

The following illustration shows the process.



More information:

Certificate Mapping (see page 269)

Required XML Document Elements for XML-DSIG Authentication

For the XML-DSIG authentication scheme to work, the XML document sent by the web service consumer must contain the following elements:

<Signature>

As the parent element for the XML signature, it specifies all information relevant to the digital signature.

To verify the signature, SOA Security Manager requires that an X.509 certificate be part of the <Signature> element in the XML document.

Because the Policy Server does not interact with a Certificate Authority for this scheme, you *must configure a certificate mapping* that maps the Issuer DN in the certificate to a corresponding entry in the referenced user store. For LDAP user directories only, you can configure the certificate mapping to require that a copy of the certificate is in the user store to be compared against the certificate in the document.

<KeyInfo>

This element specifies the key needed to validate the signature. This information may include keys, names, and certificates for the sender.

For the Policy Server to authenticate a client, this element must have enough information to determine the public key that created the signature.

<KeyName>

This is a child element of <KeyInfo>; it contains a string value that identifies the key to the recipient of the XML document. This string could be a key index, a distinguished name (DN), or an email address, for example.

The Policy Server maps the value of this element to an entry in the user store.

More information:

Certificate Mapping (see page 269)

Configure the XML DSIG Authentication Scheme

To obtain authentication information from digital signatures associated with incoming XML documents, you configure the XML DSIG authentication scheme.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click SOA Authentication Scheme, Create SOA Authentication Scheme.

   The Create Authentication Scheme pane opens.

3. Click OK.

   Authentication scheme settings open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

4. Enter a name and a description for the scheme in the General group box.
5. Select XML Digital Signature from the Authentication Scheme list.
6. Specify a protection level.
7. In the Scheme Setup group box, select how much of the XML document content is signed. A digital signature can apply only to one portion of an XML document. The choices are as follows:

   - Must cover the entire document
   - Must cover the body of the message
   - Only needs to apply to headers

   Note: If the XML document uses raw XML, select the Must cover entire document option, because the entire document is the payload. With raw XML, no envelope headers or body tags exist to distinguish the payload from other content.

8. To perform authentication over an SSL connection, select the Require Secure Transport Layer check box.
9. Click Submit.

   The authentication scheme is saved and may be assigned to application components (realms).

10. Configure certificate mapping for the XML-DSIG scheme (see page 269).

    A certificate mapping defines how data in the certificate is mapped to form a user Distinguished Name (DN), which the Policy Server uses to authenticate the client.

SAML Session Ticket Authentication

The SAML Session Ticket authentication scheme provides a mechanism for single sign-on across web services protected by the same policy store. The scheme authenticates XML messages using credentials obtained from SAML Session Ticket assertions in an HTTP header, a document's SOAP envelope, or a cookie. These strongly secure assertions are generated by a SOA Agent in the same Policy Server domain after initial authorization of the request, making the SAML Session Ticket authentication scheme the ideal basis for multiple web services deployed using the multistep authentication model or the chain authentication service model within a single enterprise.

A SAML Session Ticket assertion is a data structure that contains a SiteMinder session ticket and a public key (both encrypted). This assertion is used by the SAML Session Ticket authentication scheme to do the following:

- Verify that a valid SiteMinder session exists.
- Ensure the integrity of the signed XML document.

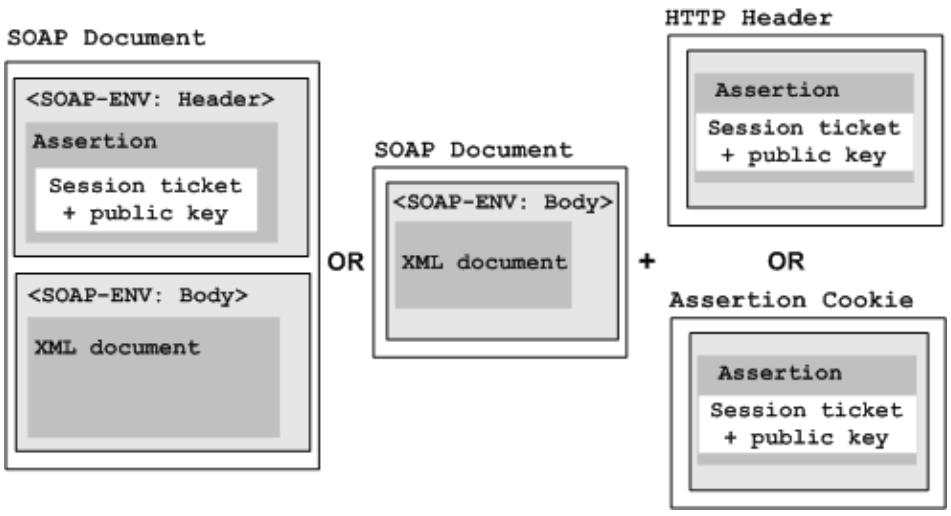
By including the session ticket and the public key in the assertion, a web service consumer can access web services protected by SOA Agents in the same Policy Server domain without being rechallenged for credentials.

How SAML Session Ticket Authentication Works

SAML Session Ticket authentication provides a mechanism for single sign-on across web services protected by the same policy store.

To generate an assertion, a web service consumer must first be authorized by the Policy Server. The authorizing policy must have a response configured with it that issues SAML Session Ticket response data. This data is used by the SOA Agent to generate the assertion. This assertion is then passed back to web service consumers who then use it to gain access to web services protected by the SAML Session Ticket authentication scheme.

When setting up the SAML Session Ticket response, you can configure whether the SOA Agent puts the assertion in a SOAP document, in an HTTP header separate from the XML document, or in a cookie as shown in the following illustration.



Assertions found within assertion cookies take precedence over assertions in the SOAP envelope or HTTP header. The SOA Agent first collects all SAML Session Ticket assertions from the cookie and the header or envelope as specified in the authentication scheme. The agent then tests each assertion until it finds the first one with a valid session ticket (that is, it can be decrypted with the agent key) and valid signatures, if they are required. Authentication is then performed using this assertion.

Note: If the authentication fails later in the authentication process because the first valid session ticket is found to be expired or revoked, authentication will fail—potential session tickets included in other assertions are *not* subsequently evaluated.

How Signing Assertions Affects SAML Session Ticket Authentication

If the TXM_Sign_Assertion variable is used to sign an assertion, the SAML Session Ticket authentication scheme behaves as follows:

- If the authentication scheme does not require a signed document, all signatures are ignored.
- If the authentication scheme does require a signed document, the following must be true:
  - The document must be signed and the signature must be valid.
  - If the assertion is signed, it must also be valid.
  - If only the assertion is signed but the document is not, the document is invalid.

Benefits of the SAML Session Ticket Authentication Scheme

The SAML Session Ticket authentication scheme allows SOA Security Manager to provide two-factor authentication. This means that authentication can rely on two things:

- Information the web service consumer knows— the password or credentials used to obtain the session ticket.
- Information the web service consumer has—the private key used with the public key bound to the assertion.

  Note: A public key does not have to be bound to the assertion.

The benefits to this type of authentication are as follows:

- For the multistep authentication service model:
  - If a request is a signed document with an assertion, the Policy Server can ensure that the message is from the entity holding the private key that matches the public key in the assertion.
  - After the initial authentication, it is the public key in the assertion that secures the transaction with the subsequent web service. Even if an unauthorized party obtains the assertion, they still cannot breach security because they do not have the client's private key.
  - Use of digital signatures in authentication service environments eliminates the need for SSL connections to protect data integrity. However, SSL still has value for encryption purposes.
- For the chain web service model, the assertion is bound to the document. This allows any web service using the SAML Session Ticket authentication scheme to identify the original web service consumer making the request, no matter how far down the chain the assertion is passed.

Multistep Authentication Service Model Using SAML Session Tickets

The multistep authentication service model is an environment in which one authentication service is responsible for authenticating all web service consumer requests. When the authentication service verifies a requestor's identity, it returns a SAML Session Ticket assertion that the web service consumer can use for subsequent requests without reauthenticating.

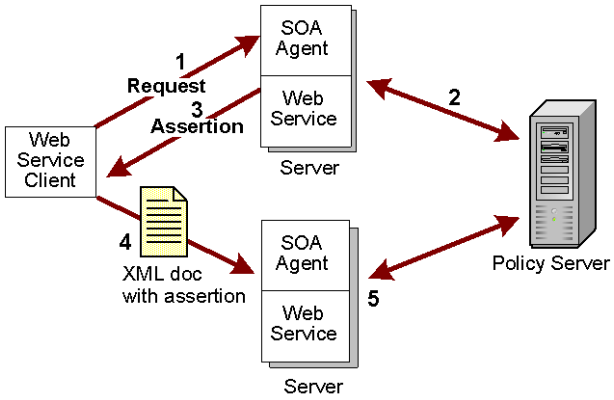
How SAML Session Ticket Multistep Authentication Using Signed XML Documents Works

In the multistep authentication service model, one authentication service is responsible for authenticating all web service consumer requests. In this model, the web service consumer must sign each XML document request using a private key associated with the public key that was provided during authentication.

The process that the web service consumer goes through when making a request has two phases:

1. Obtaining the assertion
2. Using the assertion to access other web services

The following illustration shows the complete flow of data in the multistep authentication service model.



In this illustration, the data flows as follows:

1. The web service consumer sends a request. As part of the request, a public key must be made available to the SOA Agent from the request or the policy store.
2. The SOA Agent receives the request and passes it to the Policy Server, which authenticates the web service consumer with an authentication scheme other than the SAML Session Ticket scheme, such as XML Document Credential Collector. A session ticket is generated.

   The request goes through the authorization process after authentication. If the web service consumer is authorized and a SAML Session Ticket response attribute is associated with the authorizing policy, the Policy Server generates a response and sends it to the SOA Agent.

3. The SOA Agent uses the response to generate a SAML Session Ticket assertion, which contains an encrypted session ticket and a public key. The SOA Agent puts the assertion in an HTTP header, SOAP envelope header, or cookie and passes it to the web service, which returns it to the web service consumer.

   Note: The assertion is returned to the web service consumer, not the response data returned from the Policy Server. That data is returned only to the SOA Agent, which uses it to generate the assertion.

4. For subsequent requests, the web service consumer passes a new signed XML document that includes the assertion to another web Service. The web service consumer signs the document using the private key that corresponds to the public key in the assertion.

   These subsequent requests must be in the form of SOAP documents, as required by the SAML Session Ticket authentication scheme.

5. The request is granted access to the web service without having to reauthenticate because of the information in the assertion, in effect, providing single sign-on.

How the Consumer Obtains the Assertion

When a web service consumer makes a request for a web service, the web service consumer must get the assertion from the authentication service, which is protected by the SOA Agent. The assertion can be obtained using any method of authentication, making sure the method is highly secure. If the web service consumer intends to sign documents containing the assertion, the initial request must provide a public key to the SOA Agent. The key can be provided in an XML document, by the XML-DSIG authentication scheme, or by the Policy Server from the user store.

After the web service consumer is authenticated, the web service consumer goes through the authorization process. If the web service consumer is successfully authorized, the SOA Agent responds by returning a SAML Session Ticket assertion containing an encrypted CA SiteMinder session ticket and the client's public key to the authentication service.

Note: The initial request for an assertion can be in the form of a signed XML document, but this is optional and is determined by the client's authentication service setup.

How the Consumer Uses the Assertion

The assertion is passed by the authentication service to the web service consumer for use in subsequent web service requests. Because the requesting SOAP document contains the assertion and the XML document, the web service request is highly secure and can be readily validated by other web services hosted by the same Policy Server.

Multistep Authentication Using SAML Session Tickets Without Signed XML Documents

Multistep authentication using SAML Session Tickets without signed XML documents is a less secure model in which a public key is not required to be bound to the XML document.

If no public key is supplied by the web service consumer or the Policy Server) with a request, the assertion is still generated based on a successful authentication alone. The assertion can be used by the SAML Session Ticket authentication scheme only if the scheme is configured so that it does not require a signature for the XML document.

How the Consumer Obtains the Assertion

When a web service consumer makes a request for a web service, the web service consumer must get the assertion from the authentication service, which is protected by the SOA Agent. The assertion can be obtained using any method of authentication.

After the web service consumer is authenticated, the web service consumer goes through the authorization process. If the web service consumer is successfully authorized, the SOA Agent responds with a SAML Session Ticket assertion containing only an encrypted CA SiteMinder session ticket.

How the Consumer Uses the Assertion (Multi-step)

The assertion is passed by the authentication service to the web service consumer for subsequent web service requests. Because the requesting SOAP document contains the assertion and the XML document, the web service consumer is not rechallenged by other web services hosted by the same Policy Server.

In this model, the web service consumer does not sign each XML document request, assuming the destination URL does not require a signed document.

Chain Authentication Service Model Using SAML Session Tickets

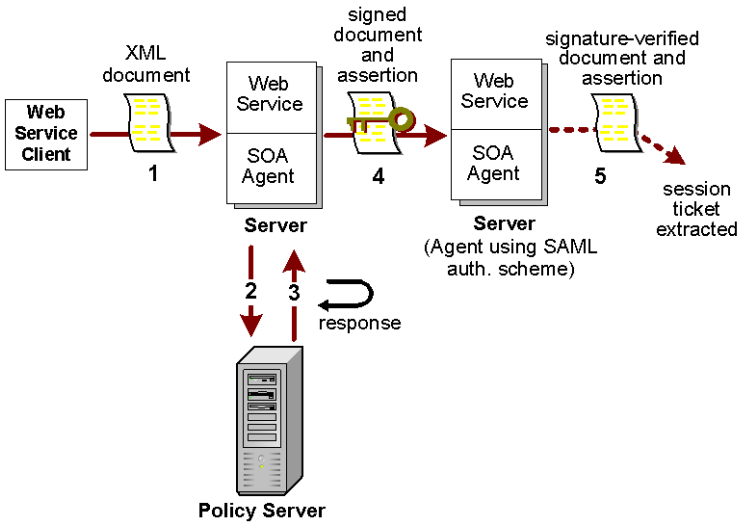
The chain authentication service model is an environment in which the first web service in the chain is responsible for authenticating all web service consumer requests.

When the authentication service verifies a requestor's identity, it binds a SAML Session Ticket assertion to the requesting XML document and passes the document to downstream web services for further processing.

How Chain Authentication with SAML Session Tickets Works

In the chain authentication service model, the first web service in the chain is responsible for authenticating all web service consumer requests.

The following illustration shows the communication flow using the SAML Session Ticket authentication in a chain environment.



In the previous illustration, the data flows as follows:

1. The web service consumer sends a request for access to a protected web Service. This request must be in the form of an XML document.
2. The SOA Agent receives the request and the Policy Server authenticates the web service consumer with an authentication scheme other than the SAML Session Ticket scheme, such as XML Document Credential Collector.

   The SAML Session Ticket authentication scheme does not protect the first web service in the chain.
3. Because the authorizing policy has a SAML Session Ticket response associated with it, after the web service consumer is authenticated and authorized the Policy Server generates a response that it sends to the SOA Agent.
4. The SOA Agent uses the response to generate a SAML Session Ticket assertion, which contains an encrypted session ticket and a public key provided by the SOA Agent.

   The SOA Agent binds the document by signing it with its private key, places the assertion and the signature into an HTTP header, SOAP envelope header, or cookie, and returns the document and the assertion to the authentication web service.

   Note: The assertion is returned to the client, not the response data returned from the Policy Server. That data is returned only to the SOA Agent, which uses it to generate the assertion.

5. The web services are configured so that the initial service passes the assertion and the document to the next web service in the chain, which is protected by the SAML Session Ticket authentication scheme.

   The assertion is validated by the SOA Agent and the session ticket credentials allow the web service consumer access without reauthenticating.

How the Consumer Obtains the Assertion

In the chain authentication service model, the web service consumer obtains a SAML Session Ticket assertion from the first web service in the chain upon successful authentication. That assertion is subsequently used by other web services in the chain to authenticate the request.

To obtain the assertion, the web service consumer must send an XML document to the first web service in the chain; the web service provider need not supply a public key.

The SOA Agent dynamically generates a public/private key pair, and then creates the assertion. The assertion contains a session ticket and the public key corresponding to the generated private key. The SOA Agent then signs the document with its private key, which binds the document to the assertion.

How the Consumer Uses the Assertion

In the chain authentication service model, the web service consumer obtains a SAML Session Ticket assertion from the first web service in the chain upon successful authentication. That assertion is subsequently used by other web services in the chain to authenticate the request.

After the assertion and document are issued by the first web service, it passes the document to the next web service in the chain. When a downstream web service receives the document, the SAML Session Ticket authentication scheme verifies the document's signature and validates the originator of the document based on the session ticket in the assertion. The application receiving this document may now process it and send it along to other web services protected by the SAML authentication scheme.

Configure the SAML Session Ticket Authentication Scheme

To obtain security information from SAML Session Ticket assertions in an HTTP header, a SOAP envelope, or a cookie associated with an incoming message, you must configure the SAML Session Ticket authentication scheme.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click SOA Authentication Scheme, Create SOA Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Enter a name and a description for the scheme in the General group box.
5. Select SAML Session Ticket from the Authentication Scheme Type list.
6. Enter a protection level.
7. In the Scheme Setup group box, do the following:
   - Specify where you want the SAML assertion to be placed. Select Envelope Header or HTTP Header option, as appropriate.
   - Optionally, if the incoming documents need to be signed, select the Require signature on XML message check box.
   - Optionally, to perform authentication over an SSL connection, select the Require Secure Transport Layer check box.

     This option is not necessary with this scheme; however, you can use SSL if you want to encrypt the communication over the network.
8. Click Submit.

   The authentication scheme is saved and may be assigned to application components (realms).

WS-Security Authentication

The WS-Security standard defines a set of SOAP header extensions that provide mechanisms for securely passing authentication data and protecting message content between web services, particular those at federated enterprises. WS-Security allows web service implementers to do the following:

- Send authentication data as part of a message using one of several supported security token types.
- Ensure message integrity using digital signatures.
- Ensure message confidentiality using XML encryption.

These mechanisms can be used independently (for example, to pass authentication data in a security token) or in combination (for example, signing and encrypting a message and providing authentication data in a security token).

For more information about the WS-Security standard, see the OASIS Standard, *Web Services Security: SOAP Message Security 1.0*.

How WS-Security Headers Are Produced

WS-Security headers are generated by a SOA Agent (or a third-party security application) after initial authorization of the request, making the WS-Security authentication scheme the ideal basis for multiple web services at federated enterprises.

For SOA Security Manager to produce WS-Security headers, a web service consumer request must first be authorized by the Policy Server using an appropriate authentication scheme (not every authentication scheme obtains everything that is required from the incoming request to create any type of token). The authorizing policy must have a response configured with it that issues WS-Security response data. This data is used by the SOA Agent to generate WS-Security headers. These headers are inserted into the SOAP message header and delivered to the protected web service application. The web service may then pass these headers to the following locations:

- Back to web service consumer applications that can then use them in further requests to gain access to other web services protected by the WS-Security authentication scheme.
- Downstream to further web services protected by the WS-Security authentication scheme.

More information:

Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

Supported Authentication Schemes for Producing Each WS-Security Header Type

You can configure responses to produce any type of WS-Security token upon successful authorization of a request. However, not every authentication scheme gathers all the necessary information (username, clear text password, SOAP message) from an incoming request to create every type of token.

If a response is configured to create a token that requires anything that the configured authentication scheme does not provide, header creation fails. Verify that the authentication method that you plan to use is suitable to produce the WS-Security token that you want to produce in response.

The following table shows which WS-Security tokens can be produced for each authentication method.

	WS-Security Token Types That Can be Produced			
Authentication Method	***Username and Password***	***Username and Password Digest***	***SAML***	***X.509***
Basic (SOA Agent for Web Servers only)	No	Yes	No	No
XML-DCC	Yes	Yes	Yes	Yes
XML-DSIG	No	No	Yes	Yes
SAML Session Ticket	No	No	Yes	Yes
WS-Security Username and Password Token	Yes	Yes	Yes	Yes
WS-Security Username and Password Digest Token	No	Yes	Yes	Yes
WS-Security SAML Token	No	No	Yes	Yes

	WS-Security Token Types That Can be Produced			
WS-Security X.509 Token	No	No	Yes	Yes
SiteMinder Session (SMSESSION) Cookie	No	No	No	No

How WS-Security Headers Are Consumed

The SOA Security Manager WS-Security authentication scheme authenticates XML messages using credentials obtained from WS-Security headers that define security tokens in a message's SOAP header. The authentication scheme also validates digital signatures and decrypts XML encrypted headers as necessary.

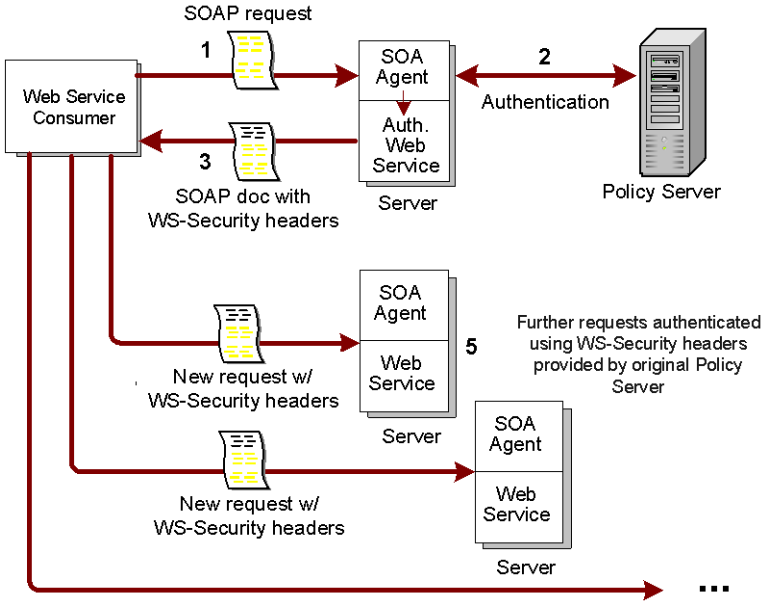
How the Multistep Authentication Service Model Works Using WS-Security

The multistep authentication service model is an environment in which one authentication service is responsible for authenticating all web service consumer requests. When the authentication service verifies a requester's identity, it returns WS-Security headers that the web service consumer can use for highly secure authentication of subsequent requests.

The process that the web service consumer goes through when making a request has two phases:

- Obtaining the WS-Security headers
- Using the WS-Security headers to access other web services

The following illustration shows the multistep authentication service model using WS-Security headers.



1. The web service consumer sends a request in the form of a SOAP document.
2. The SOA Agent receives the request and passes it to the Policy Server, which authenticates the web service consumer using a supported authentication scheme.

   The XML request goes through the authorization process after authentication. If the web service consumer is authorized, a WS-Security response attribute associated with the authorizing policy causes the Policy Server to generate a response and send it to the SOA Agent.
3. The Agent uses the response to generate and adds WS-Security headers to the request's SOAP headers. The SOA Agent then passes the SOAP request to the web service, which returns it to the web service consumer.

   Note: The WS-Security token is included in the SOAP message forwarded to the next web service in the chain, not the response data returned from the Policy Server. That data is returned only to the SOA Agent, which includes it in WS-Security headers.
4. For subsequent requests, the web service consumer passes the original or a new SOAP document that includes the WS-Security token to another web Service (within the Policy Server domain or at a federated enterprise) protected with the WS-Security authentication scheme.
5. The request is authenticated based on the WS-Security token and granted access to the web service.

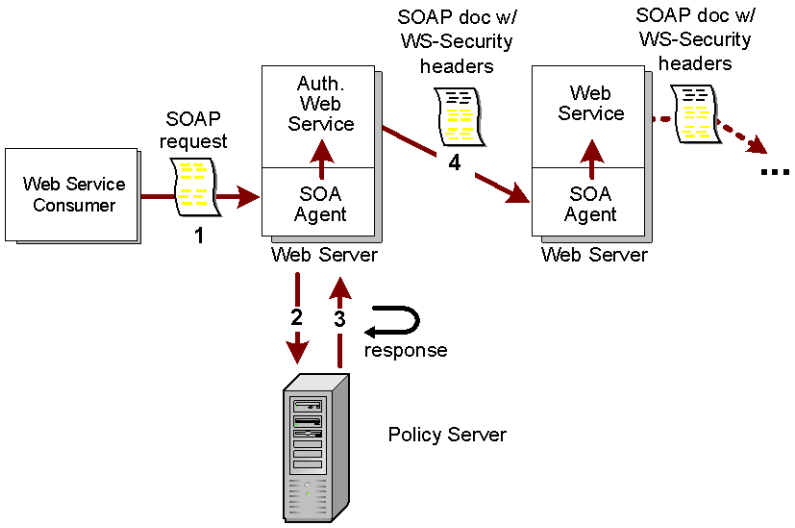
More information:

Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

How the Chain Authentication Service Model Works Using WS-Security

The chain authentication service model is an environment in which the first web service in the chain is responsible for authenticating all web service consumer requests. When the authentication service verifies a requestor's identity, it add WS-Security headers to the requesting SOAP document headers and passes the document to downstream web services for further processing.

The following illustration shows the communication flow using WS-Security tokens in a chain environment.



1. The web service consumer sends a request for access to a protected web service in the form of a SOAP document.
2. The SOA Agent receives the request and the Policy Server authenticates the web service consumer using a supported authentication scheme.
3. The XML request goes through the authorization process after authentication. If the web service consumer is authorized, a WS-Security response attribute associated with the authorizing policy causes the Policy Server to generate a response and send it to the SOA Agent.

The SOA Agent uses the response to generate and add WS-Security headers to the request's SOAP headers. The SOA Agent then passes the SOAP request along to the authentication web service.

4. The authentication web service sends the SOAP document with WS-Security headers to the next web service downstream, which is protected using the WS-Security authentication scheme.

More information:

Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

Choose a WS-Security Token Type

When using WS-Security authentication to protect resources, choosing the token type to use is by far the most significant decision to be made. The token type you use determines the level of the following:

- Security
- Flexibility in terms of authorization data that can be passed
- Configuration complexity

SOA Security Manager provides complete support for production and consumption of SOAP messages with WS-Security headers containing the following supported token types:

Username and Password Digest Token

Secures a message using username and digested password.

Username and Password Token (Clear Text)

Secures a message using a username and clear text password.

X509v3 Certificate Token

Secures a message with a X509v3 binary security token.

SAML Assertion Token

Secures a message using a SAML assertion.

Username and Password Digest Token

The Username and Password Digest token provides password element confidentiality without requiring channel-level security for the entire document.

The Username token includes a username and password, a cryptographic nonce (a parameter that varies with time) and, optionally, a timestamp. The password is hashed as an SHA1 digest using the nonce, timestamp, and password:

password_digest = SHA1[*nonce* + *timestamp* + *password*]

When a timestamp is included, creating SHA1 password digests provides protection against replay attacks that prevents an eavesdropper from cutting out and replaying the <wsse:UsernameToken> element in a different document at a later time. Also, hashes of the same password along with the same nonce still resolve to different digest values, assuming that the timestamp has been updated.

The Username and Password Digest Token authentication scheme provides protection against replay attacks (where an eavesdropper might cut out and replay the token at a later time) by imposing a limit (60 minutes by default) on the age of the token. That is, if a token was created more than 60 minutes ago according to its <wsu:Created> timestamp, authentication fails.

Note: The Username and Password Digest token is supported only with LDAP and ODBC-based user directories. For LDAP user directories, SOA Security Manager must be configured (using the Credentials and Connection tab in the Policy Server User Interface) to connect to the user store using an LDAP administrative identity if the directory implementation requires such credentials to return the userPassword attribute. For ODBC user directories, a "password" user property must be added to the SQL query scheme used by the directory.

Note: The password storage schemes used by the Username token-generating site must be consistent with the password storage scheme used by the Username token-consuming site. For instance, if the generating site uses SHA-1 password hashes in its user directory, then the consuming site must do the same.

More information:

Variables for Generating Username and Password and X.509 Certificate Tokens (see page 367)

Username and Password Token (Clear Text)

The Username and Password token provides the token subject's username and clear-text password.

Note: The password storage schemes used by the Username and Password token-generating site must be consistent with the password storage scheme used by the Username token-consuming site. For instance, if the generating site uses SHA-1 password hashes in its user directory, then the consuming site must do the same.

Important! CA recommends that you always use Username and Password tokens with digital signatures or XML encryption to prevent malicious parties from intercepting the message and obtaining the username and password from it.

More information:

Variable for Specifying the Generated WS-Security Token Type (see page 366)

X509v3 Certificate Token

The X509v3 certificate security token provides the token subject's X.509v3 certificate in a SOAP document.

When configured to require X509v3 certificate tokens, the WS-Security authentication scheme provides basically the same functionality as the XML Digital Signature authentication scheme, but without requiring certificate mapping, since the signature and key information are contained in standard header elements.

Using the X509v3 certificate token enables the SOA Agent to do the following:

- Verify the signature
- Ensure that the signature is signed by using a trusted certificate
- Confirm that the document has not been altered.

After the signature is verified, the Policy Server does the following:

- Uses the digital signature and other information in the user store entry to confirm that the XML document is actually from a trusted client.
- Checks that the web service consumer has a valid entry in the user store.

When generating X.509v3 tokens, SOA Security Manager uses the host web service enterprise's certificate, which it obtains from the Smkeydatabase.

More information:

Smkeydatabase Overview (see page 281)

SAML Assertion Token

The SAML assertion token specification (see OASIS Working Draft 14, *Web Services Security: SAML Token Profile*, July 12, 2004) extends the token-independent processing model defined by the core WS-Security specification, allowing SAML assertions to be used to provide secure authentication data.

The SAML assertion includes the identity of the web service consumer (typically as its subject) and, optionally, other associated attributes. Additionally, the SAML token specification provides for the use of digital signatures to guarantee the integrity and authenticity of the SAML assertion, its issuer, and the subject of the assertion, using one of the following:

- The Enterprise certificate/public key of the assertion issuer (a trusted site where the request is authenticated and the SAML token generated) to sign the assertion and its contents.
- The Enterprise private key of the assertion subject (the organization being represented by the web service consumer making the request) or the assertion issuer to sign the SOAP document (depending on the subject confirmation method.

So, when using the WS-Security authentication scheme to authenticate requests with SAML assertion tokens, SOA Security Manager validates the request, to ensure that the assertion comes from a trusted source, by authenticating the assertion subject and the assertion issuer. For example, in a multiple web service implementation using SAML tokens, SOA Security Manager would validate the assertion subject (the web service consumer that made the initial web service request) and the assertion issuer (a SOA Security Manager-protected authentication service configured to produce SAML WS-Security tokens).

More information:

Token Subject and Issuer Confirmation and Validation (see page 252)

Token Subject and Issuer Confirmation and Validation

The WS-Security authentication scheme can use multiple aspects of an incoming SOAP document to validate the subject and issuer of a SAML token:

- The web service consumer represented as the subject of the assertion can be validated against an associated user database.
- The assertion issuer can be validated against a corresponding certificate or public key in the Smkeydatabase.
- The integrity of the message content can be verified using the signature of the issuer, the subject, or both.

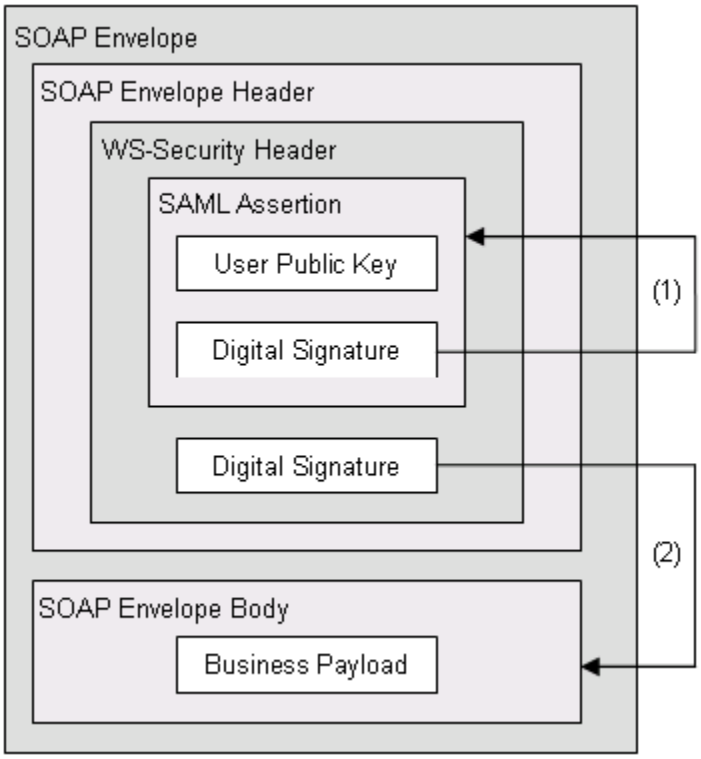
The exact manner in which the authentication scheme assertion subject and issuer are validated depends on the subject confirmation method that you specify. SOA Security Manager supports both methods defined in the *Web Services Security SAML Token Binding* specification.

Holder-of-Key Subject Confirmation Method

The holder-of-key method confirms that the subject of the SAML token and the sender of the request containing that token have the same identity, mandating the following:

- The SAML token must contain the public key/certificate of the subject and be signed by the token issuer.
- The request sender must sign the SOAP envelope with its private key and the signature reference must point to a matching public key/certificate in the SAML token.

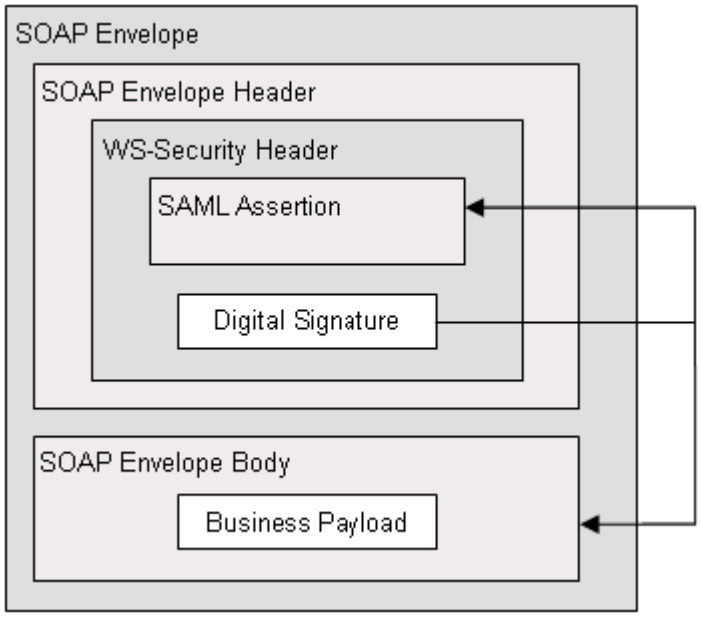
SAML tokens with the holder-of-key subject confirmation method are reusable. That is, when returned to the web service consumer (as in the multistep authentication method), a holder-of-key SAML token can subsequently be associated with *other* SOAP documents—not just with the document sent to the authentication service. This allows the web service consumer to send one request to the authentication service and then associate the returned token with multiple documents and send them directly to target web services.



Sender-Vouches Subject Confirmation Method

The sender-vouches method confirms that the subject of the SAML token was authenticated by the trusted token issuer, which vouches that the identity of the token's subject is the web service consumer making the request. The token issuer then binds the assertion and the message body together by signing them with its private key.

Note: The sender-vouches token is valid only in association with the SOAP document to which it is bound by the authentication service—it *cannot* be reused with other documents.



The identity of the token issuer can be validated during authentication by doing the following:

- Verifying the issuer's digital signature, which can cover the assertion (holder-of-key) or the assertion and message content (sender-vouches), and then validating that the issuer's certificate is one of the trusted issuer public keys/certificates stored in the Smkeydatabase.
- (Sender-vouches only) Using an SSL link (established using the SSL client cert option) between the issuer of the token and the downstream web service that consumes it. In this case, the assertion and contents do not necessarily need to be signed.

The main distinction between these validation mechanisms is that the digital signature mechanism *requires* signed elements within the message being processed. When using the SSL case, you can improve security by configuring SOA Security Manager to obtain the token issuer's client certificate from the link and use it to validate against the Smkeydatabase.

The following table provides a complete summary of the security features and associated validation methods available for messages using the different subject and issuer confirmation methods.

Confirmation Method	Message Security Features	Subject/Issuer Validation Method
Holder-of-key	The web service consumer's authenticated ID is the subject of the SAML assertion token (default, alternative subject can be specified using an explicit XPath statement); subject's public key/certificate is included in the assertion. The SAML token issuer signs the assertion using its certificate/public key. The assertion content is signed by the subject and is validated based on the fact that the user is the holder of the private key that corresponds to the public key in the assertion that is used to verify the content signature.	Assertion subject is validated against an associated user directory Token subject's public key/certificate is validated against the Smkeydatabase. Token issuer's public key/certificate is validated against the Smkeydatabase.
Sender-vouches with signature-based issuer validation	The web service consumer's authenticated ID is the subject of the SAML assertion token (default, alternative subject can be specified using an explicit XPath statement); subject's public key/certificate is not included in the assertion. The SAML token issuer signs the assertion. The SAML token issuer also signs the complete SOAP document.	Assertion subject is validated against an associated user directory Token issuer's public key/certificate is validated against the Smkeydatabase

Confirmation Method	Message Security Features	Subject/Issuer Validation Method
Sender-vouches with SSL-based issuer confirmation	The web service consumer's authenticated ID is the subject of the SAML assertion token (default, alternative subject can be specified using an explicit XPath statement); its public key/certificate is not included in the assertion. The SAML token issuer is the holder of the client certificate and private key used to set up the SSL connection.	Assertion subject is validated against an associated user directory. The token issuer and content, including the SAML token, are implicitly validated by being encrypted and decrypted over SSL using SSL encryption keys derived from the issuer's client and server certificates used to set up the SSL link. That is, the encryption may be thought of as a form of the client certificate indirectly signing the content. The token issuer's public key/certificate can be validated against the SSL key store on the web server or the Smkeydatabase.

Note: Sender-vouches with SSL-based issuer confirmation is weaker than sender-vouches with signature-based issuer validation. With signature-based issuer validation, each message is protected by the holder of the issuer private key, whereas in the SSL case only the channel is protected by the holder of the issuer private key. This means if an intruder gets access to the channel, anything sent on that channel is considered valid; whereas, in the general case the intruder would have to actually get access to the issuer private key itself to sign individual messages.

How SAML Token Multistep Authentication, Holder-of-Key Works

In this highly secure model, the web service consumer must sign each XML document request using a private key associated with the public key that was provided during authentication.

- **Obtaining the SAML Token**—When a web service consumer makes a request for a web service, the web service consumer must obtain the SAML token from the authentication service, which is protected by the SOA Agent. The SAML token can be obtained using any supported authentication scheme. Also, the initial request must provide a public key to the SOA Agent. The key can be provided in an XML document, using the XML-DSIG authentication scheme or provided by the Policy Server from the user store.

After the web service consumer is authenticated, the web service consumer goes through the authorization process. If the web service consumer is successfully authorized, the SOA Agent generates a SAML token containing the client's public key and signs it with its own public key/certificate.

- **Using the SAML Token**—The token is passed by the authentication service back to the web service consumer in a SOAP document. When making subsequent web service requests, the web service consumer places the token in the SOAP request and signs the entire document using its private key. Because the requesting SOAP document is signed by the token subject and the token is signed by the issuer, the request can be authenticated by other web services configured to use the WS-Security authentication scheme with a high degree of confidence that it came from a trusted source.

More information:

Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

How SAML Token Multistep Authentication, Sender-Vouches Works

This model does not require the assertion subject's public and private keys bound to the SOAP document. The web service consumer's public key is not supplied (by the web service consumer or the Policy Server) with a request. Upon validation of the request, the authentication service vouches for the web service consumer by generating a SAML token and binding it to the message body by signing them both with its private key using the sender-vouches subject confirmation method.

- **Obtaining the SAML Token**—When a web service consumer makes a request for a web service, the web service consumer must get the SAML token from the authentication service, which is protected by the SOA Agent. The SAML token can be obtained using any method of authentication.

  After the web service consumer is authenticated, the web service consumer goes through the authorization process. If the web service consumer is successfully authorized, the SOA Agent responds generating a WS-Security SAML token signed using the token issuer's private key.

- **Using the SAML Token**—The signed SOAP document (SAML token and request message body) is passed by the authentication service to the web service consumer for use in subsequent web service requests. Because requests take the form of a SOAP document in which the SAML token is bound to the message payload using the signature of the trusted token issuer, they can be accepted by other web services with a high degree of confidence that it came from a trusted source.

  Note: The sender-vouches SAML token is valid bound only to the original request message—it cannot be reused with other requests.

  In this model, the web service consumer does not sign each XML document request, assuming the destination URL does not require a signed document.

More information:

Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

How Chain Authentication Service Model, Sender-Vouches with Signature-Based Issuer Validation Works

This service model does not require the SAML assertion subject's public and private keys bound to the request. A SAML token that uses the sender-vouches subject confirmation method is generated and used to authenticate the web service consumer by downstream web services; the token issuer's identity is validated against its private key, which is bound to the token and the SOAP request.

- **Obtaining the SAML Token**—When a web service consumer makes a request for a web service, the web service consumer must get the SAML token from the authentication service, which is protected by the SOA Agent. The SAML token can be obtained using any method of authentication.

  After the web service consumer is authenticated, the web service consumer goes through the authorization process. If the web service consumer is successfully authorized, the SOA Agent responds by adding a WS-Security SAML token signed using the token issuer's private key to the SOAP request. The SOA Agent then also signs the request using the token issuer's private key.

- **Using the SAML Token—**After the signed SOAP request containing the SAML token is generated, the authentication service passes the document to the next web service in the chain. When the downstream web service receives the document, the WS-Security authentication scheme verifies the token issuer by its signature and validates the originator of the document based on the SAML token contents. The application receiving this document may now process it and send it along to other web services protected by the WS-Security authentication scheme.

More information:

Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

How Chain Authentication Service Model, Sender-Vouches with SSL-based Issuer Confirmation Works

This model does not require the assertion subject's public and private keys bound to the request. A SAML token that uses the sender-vouches subject confirmation method is generated and used to authenticate the web service consumer by downstream web services. The token issuer is implicitly authenticated by being encrypted and decrypted over SSL using SSL encryption keys derived from the issuer's client and server certificates used to set up the SSL link.

- **Obtaining the SAML Token**—When a web service consumer makes a request for a web service, the web service consumer must get the SAML token from the authentication service, which is protected by the SOA Agent. The SAML token can be obtained using any supported authentication scheme.

  After the web service consumer is authenticated, the web service consumer goes through the authorization process. If the web service consumer is successfully authorized, the SOA Agent responds by adding a WS-Security SAML token to the SOAP request.

- **Using the SAML Token—**After the signed SOAP request containing the SAML token is generated, the authentication service establishes an SSL connection to the next web service in the chain and uses that secure connection to pass the request. When the downstream web service receives the document, the WS-Security authentication scheme validates the originator of the document based on the SAML token contents. Optionally, the token issuer's public key/certificate can be obtained from the SSL key store on the web server and used to validate the issuer identity. The web service receiving this document may now process it and send it along to other web services over further SSL connections.

More information:

Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

Additional WS-Security Features

SOA Security Manager authentication also provides support for supplemental WS-Security utility features:

- XML encryption
- Message timestamps
- Digital signature scope restrictions
- Choosing between multiple WS-Security headers using SOAP roles

XML Encryption

SOA Security Manager supports encryption and decryption of any combination of WS-Security message header elements and body elements using XML encryption compliant with OASIS Standard 200401, *Web Services Security: SOAP Message* Security *1.0*.

XML encryption of WS-Security messages provides end-to-end security for web service applications that require secure exchange of structured data. XML encryption provides security functionality that cannot be provided by point-to-point security protocols such as SSL. Specifically, it allows you to:

- Encrypt only part of the data being exchanged
- Secure sessions between more than two parties

How SOA Security Manager Obtains Credentials from Encrypted WS-Security Documents

The WS-Security authentication scheme automatically attempts to decrypt any XML-encrypted elements in incoming WS-Security messages for SOA Security Manager to use for authentication/authorization. No additional configuration is required.

Note: Where an incoming SOAP message contains multiple WS-Security header elements, each is identified by a unique SOAP actor/role attribute. In such cases, SOA Security Manager attempts to decrypt only XML-encrypted elements specified in the header from which the WS-Security authentication scheme is configured to obtain security tokens.

More information:

SOAP Actor/Role Attributes in Messages with Multiple WS-Security Headers (see page 263)

Configure SOA Security Manager to Perform Encryption and Decryption of WS-Security Documents

SOA Security Manager can encrypt any WS-Security message that contains the recipient's X.509 certificate in a WS-Security header. SOA Security Manager extracts the recipient's public key from their X.509 certificate and uses this to encrypt a symmetric key, which it then uses to encrypt the desired header and message elements. Multiple encryption algorithms are available; different encryption algorithms can be used for encryption of the symmetric key and header/message elements.

Configure SOA Security Manager to perform XML encryption on elements of outgoing messages by adding appropriate response attribute variables to a response configured to generate WS-Security headers.

Although the WS-Security authentication scheme automatically decrypts encrypted elements in incoming messages, the default behavior of SOA Security Manager is to deliver messages to the recipient web service in encrypted form. However, you can configure SOA Security Manager to deliver decrypted versions of incoming encrypted WS-Security messages by configuring a response with the TXM_WSSEC[_SAML]_ENCRYPT_DECRYPT response attribute variable and associating it with the authorizing policy.

XML Encryption and Decryption Service Use Case

In multistep and chain authentication service models, encryption or decryption may be considered part of message preparation before sending to the ultimate destination. Thus, the SOA Security Manager XML encryption and decryption functionality might typically be used to implement a WS-Security encryption or decryption web service.

For example, consider a business relationship between two companies—Company A and Company B. Company A wants to end detailed bids on contracts to Company B without unauthorized personnel at Company B seeing the message (as would be the case if it was simply sent over an SSL link).

To implement this business logic using SOA Security Manager-protected web services, Company A develops the following:

- A web service consumer application that takes the bid, places it in XML format and wraps it with SOAP headers, placing Company B's X.509 certificate in a WS-Security header. The application then sends it to Company A's encryption web service.

- An encryption web service protected by the WS-Security authentication scheme and an authorization policy configured to do the following:
  - Obtain the intended recipient's public key certificate (in this case Company B's certificate) from the message headers
  - Encrypt the required header and message elements.

  The encryption web service then forwards the encrypted message to a decryption web service at Company B.

Company B develops a Decryption web service protected by the WS-Security authentication scheme and an authorization policy configured to deliver the decrypted version of message header/body elements.

Encryption a lgorithms

SOA Security Manager supports the following XML encryption algorithms:

- **Key transport algorithms** (used to encrypt the symmetric key):
  - rsa-1_5
  - rsa-oaep-mgf1p
- **Block Encryption algorithms** (used to encrypt message data):
  - tripledes-cbc
  - aes128-cbc
  - aes256-cbc
  - aes192-cbc

Message Timestamps

Regardless of the particular security token used by any WS-Security document, a utility timestamp element, which specifies the expiry time of a message, can be specified. If this element is covered by an XML signature, then the timestamp provides a protection against replay attacks for the entire XML document (different from the replay attack defense provided by the Username and Password Digest token) by giving an indication of the document's "freshness."

Note: The expiry feature does not completely address the problems introduced by unsynchronized clocks. The receiving party in a WS-Security message exchange may receive a document before the timestamp's created time; the issue of acceptable skew is a receiving policy issue, while the expiry offset is a creation policy issue.

XML Signature Scope

SOA Security Manager provides three options for defining what elements of an incoming SOAP message are digitally signed when configuring WS-Security authentication using either Username and Password Digest or X509v3 tokens:

- Signature must cover the entire document
- Signature must cover the body of the message
- Signature need apply only to headers

Notes:

For the Username and Password Digest token, XML digital signatures are optional.

If the authentication scheme is configured to require the timestamp element, the digital signature must cover that timestamp.

SAML token authentication has its own requirements for what elements of a SOAP message must be digitally signed; these are defined implicitly based on the subject confirmation methods that you require.

SOAP Actor/Role Attributes in Messages with Multiple WS-Security Headers

If a SOAP document has multiple WS-Security headers (intended for different recipients), the WS-Security specification requires that each be identified uniquely using the SOAP actor/role attribute (at most, one header can omit the SOAP actor attribute).

The WS-Security authentication scheme lets you specify the value of the SOAP actor/role attribute that identifies the header element from which SOA Security Manager should obtain security tokens.

Note: If a message has only one WS-Security header, it does not need to include a SOAP actor attribute. However, if you specify an actor/role attribute when configuring the authentication scheme, a matching actor attribute must be present in the document to allow successful authentication.

Username and Password Digest Token Age Restrictions

The WS-Security authentication scheme provides protection against replay attacks using Username and Password Digest tokens by imposing a "freshness" restriction (60 minutes by default) on the age of the token. That is, if a token was created more than 60 minutes ago according to its <wsu:Created> timestamp, authentication fails.

The token age restriction for Username and Password Digest Tokens can be configured at the agent level. For more information, see the *SOA Agent Configuration Guide*.

Configure the WS-Security Authentication Scheme

To obtain security information from WS-Security headers in incoming XML messages, you must configure the WS-Security authentication scheme.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click SOA Authentication Scheme, Create SOA Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Enter a name and a description for the scheme in the General group box.
5. Select WS-Security from the Authentication Scheme Type list.
6. Specify a protection level.
7. In the Scheme Setup group box, select one of the following required Security Token Types:

   - Username and Password Digest (the default). Also valid for Username and Password tokens (clear text).
   - X509v3 Certificate
   - SAML Assertion

   If you select Username and Password Digest or X509v3 Certificate, the XML Signature Restrictions group box is activated. If you select SAML Assertion, the SAML Token Restrictions group box is activated.

8. If you selected the Username and Password Digest or X509v3 Certificate security token type, specify how restrictions should be applied in the XML Signature Restrictions group box:
   - No Signature Required (the default)
   - Must Cover Entire Document
   - Must Cover Body of Message
   - Only Needs to Apply to Headers
9. If you selected the SAML Assertion security token type, locate the SAML Token Restrictions group box and complete the following fields to specify how token restrictions should be applied:
   - **Audience**

     (Optional for SAML 1.1; required for SAML 2.0) Specifies a required value for a SAML assertion token's <saml:Audience> element. If left undefined, no restriction is placed on the Audience element's content.
   - **Attribute Name/XPATH**

     (Optional) Specifies an XPATH expression that can be used to obtain a user identity value if the user identity in the assertion token's saml:NameIdentifier element (which is used by default) is not suitable for authentication. The default behavior tends to be sufficient for associated non-LDAP user directories and SOA Security Manager-generated SAML tokens.

     For LDAP user directories, specify an XPATH expression that returns a user identity value that will resolve correctly. For example:

     //SMlogin/Username/text()

     For SAML tokens not generated by SOA Security Manager (which may place other information, such as E-mail addresses in the saml:NameIdentifier element), you should also specify an XPATH expression that extract only the required information.

     If you need to strip standard prefixes to return the required attribute value itself, see Stripping Standard Prefixes from XPath Queries.
   - **Attribute Search String**

     If you specified an Attribute Name/XPATH expression in the field above, specifies a search string to apply to the result of the XPATH expression to obtain a user's DN in the Attribute Search String field.

     For LDAP user directories, this search string should be in the form:

     attribute=LDAP:uid=%s

     (You may need to modify the "uid=%s" component depending on how lookups are performed in a specific directory.)

- **Allow sender-vouches**

  (Optional) Require digitally signed SAML assertions created with the sender-vouches subject confirmation method.

- **Allow holder-of-key**

  (Optional) Require digitally signed SAML assertions created with the holder-of-key subject confirmation method.

  Note: Select Allow sender-vouches *and* Allow holder-of-key options to allow signed assertions created with either confirmation method. Select neither option to allow unsigned SAML assertions.

- **Use SSL Keystore**

  (Optional) Configures SOA Security Manager to use the web server's SSL keystore to validate the assertion certificate instead of its own keystore.

10. Configure the following other fields as necessary:

    - **Issuer certificate match required**

      (Optional) Require that the assertion issuer and the certificate DN (found within the WS-Security document) have the same identity.

    - **Require Signature over WSU:Timestamp Element**

      (Optional) Require validation of any <wsu:Timestamp> and <wsu:TimestampTrace> elements in the message

    - **Require Secure Transport Layer**

      (Optional) Require all messages to be received over a secure (SSL or TLS) connection.

11. (Optional) For messages with multiple WS-Security headers, specify the value of the SOAP actor (role) attribute that identifies the header element from which SOA Security Manager should obtain security tokens in the SOAP Role field (located in the Advanced group box). For example:

    http://www.example.com/soap/MySOAPRole

12. Click Submit.

    The authentication scheme is saved and may be assigned to application components (realms).

Strip Standard Prefixes from XPath Queries

When specifying an XPath expression to identify a SAML assertion attribute that specifies the user identity for WS-Security authentication in the Attribute Name/XPATH field, you may need to strip standard prefixes to return the attribute value itself. The XPath substring-after function provides a standard method to perform this operation.

For example, consider a SAML assertion created by the CA SiteMinder SAML Assertion Generator. This assertion contains an attribute "username" that specifies the user identify that you want to use for authentication in the following format:

```
header:uid=username
```

To remove the unwanted prefix, "header:uid=", use the XPath substring-after function in the XPath query in which you specify the target attribute. For example the following Xpath query will return "username" rather than the whole string "header:uid=username":

```
substring-after(//SMprofile/NVpair[1]/text(),"header:uid=")
```

Federation Use Cases

The following use cases illustrate how WS-Security can be used to provide secure document-based federation using the SAML token type.

Sender-Vouches Scenario

In this use case, acmewidget.com has a purchasing agreement with widgetsupplies.com, and widgetsupplies.com has a business relationship with widget-ship.com.

The end user logs on to her procurement application with her username and password. The procurement application provides a list of acmewidget.com's various suppliers. The end user clicks on the PinSupplies button and is presented with a purchase order in an HTML page. She fills out the purchase order and then clicks the Submit button on the HTML form.

The procurement application turns the HTML form into an XML document that it inserts in the envelope body of a SOAP message. The procurement application then inserts the end user's credentials in the envelope header of the SOAP message, together with acmewidget.com's customer identity.

1. SOA Security Manager authenticates the user request for the Purchasing web service at widgetsupplies.com using the document credentials collector (DCC) authentication scheme. The purchase order is processed by widgetsupplies.com.
2. SOA Security Manager generates a SAML assertion (including the original end user's security information) and inserts it in the WS-Security (WSSE) header part of the SOAP message. SOA Security Manager signs the SAML assertion and the body of the SOAP message with widgetsupplies.com's private key.

3. The Purchasing web service at widgetsupplies.com posts the SOAP message generated by SOA Security Manager to the Shipping web service at widget-ship.com.
4. At widget-ship.com, SOA Security Manager authenticates the request using the WS-Security-SAML sender-vouches authentication scheme. SOA Security Manager checks the signature covering the SAML assertion and the message body and validates that the SAML assertion was issued by a trusted partner (widgetsupplies.com vouches for the user). widgetship.com can now process the shipment order for the original requester at acmewidget.com.

Note: This scenario demonstrates the ability of SOA Security Manager to be at both widgetsupplies.com and widget-ship.com, but either site can have a WS-Security/SAML-compliant third-party security application if desired.

Holder-of-Key Scenario

Tokenprovider.com is an identity provider, that is, tokenprovider.com generates security tokens to be used to access resources hosted by remote service providers. In this scenario, the end user is an employee of acmewidget.com, but the end user could also be an employee of tokenprovider.com.

WSprovider.com is a remote service provider that accepts credentials produced by tokenprovider.com.

tokenprovider.com and WSprovider.com have a business agreement. In this scenario, tokenprovider.com and WSprovider.com are two separate companies (different Internet domains), but they also could be two departments of the same company.

1. The end user's application signs the request for authentication using the user's private key and posts it to the Authentication web service at tokenprovider.com.
2. SOA Security Manager authenticates the user request using the XML Signature (XML DSIG) authentication scheme. SOA Security Manager generates the SAML assertion, inserts the user's certificate into the SAML assertion, and signs the assertion with tokenprovider.com's private key.
3. The Authentication web service at tokenprovider.com returns the signed SAML assertion just created (in a raw XML document or in a SOAP/WS-Security response).
4. The user's application at acmewidget.com inserts the SAML assertion in a WS-Security header as part of a SOAP request and posts the request to WSprovider.com's web service.

5. At WSprovider.com, SOA Security Manager authenticates the request using the WS-Security SAML holder-of-key authentication scheme. SOA Security Manager checks the SAML assertion signature and the user's signature on the message body, validates that the SAML assertion was issued by a trusted partner (in this case tokenprovider.com), and validates that the user is in the user store.

Note: This scenario demonstrates how SOA Security Manager can be at both tokenprovider.com and WSprovider.com, but either site can have a third party security application provided that application is compliant with the standards involved (XML signature, WS-Security, and SAML).

Certificate Mapping

X.509 client certificates provides strong user authentication. However, in order for SOA Security Manager to use a certificate to identify a user, the certificate must be compared to a user's information in a directory. SOA Security Manager uses a certificate mapping to determine how to compare a user's certificate with the information stored in the user directory.

SOA Security Manager supports certificate mapping for users whose authentication information is stored in a WinNT, Microsoft SQL Server, Oracle, or LDAP user directory. A certificate mapping defines how data in the certificate is mapped to form a user Distinguished Name (DN). The Policy Server uses this user DN to authenticate the user.

If certificates are stored in an LDAP directory, a certificate mapping can direct the Policy Server to verify that the certificate presented by the user matches the certificate associated with the user DN in the LDAP directory.

More information:

User Directories (see page 85)

Configure a Certificate Mapping

You can configure a certificate mapping that lets SOA Security Manager determine how to compare a user's certificate with the information stored in the user directory.

To configure an Impersonation authentication scheme

1. Click Infrastructure, Directory.
2. Click Certification Mapping, Create Certificate Mapping.

   The Create Certificate Mapping pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Type the certificate issuer DN in the Issuer DN field.

   Note: You must escape reserved special characters with a backslash (\). More information on reserved special characters for LDAP distinguished names exists at http://www.faqs.org/rfcs/rfc2253.html. Special characters include:

   - commas (,)
   - semicolons (;)
   - quotes (")
   - backslashes (\)
   - plus character (+)
   - greater than character (>)
   - less than character (<)

   Note: Issuer DNs cannot exceed 255 characters if a relational database is used as a policy store; Issuer DNs cannot exceed 1000 characters if an LDAP directory server is used as a policy store.
4. Specify how the X.509 client certificate is to map user authentication information in the authentication directory on the Mapping group box.
5. (Optional) Select Perform CRL Checks on the Certificate Revocation List (CRL) Checking group box, and specify the CRL settings on the group box.
6. Click Submit.

   The Create Certificate Mapping task is submitted for processing.

Test a Certificate Mapping

Testing a certificate mapping displays the search string the Policy Server is to use to map client certificates to user directory attributes.

To test a certificate mapping

1. Open the certificate mapping.
2. Click Test in Mapping group box.

   The Certificate Map Test group box opens.
3. Select a user directory connection from the Directory list.

   Note: The Directory list includes all of the existing directory connections of the type you selected when creating the certificate mapping.

   The contents of the Directory Information group box change based on the type of user directory connection. For WinNT, ODBC and OCI user directory connections, the group box displays the Directory Type you are testing. For LDAP directory connections, the group box displays the Directory Type, as well as the Lookup Start and Lookup End values used to locate a user's DN within the LDAP directory.

   The Policy Server tests the certificate mapping and the Certificate Map Test group box provides the results.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Click Close.

   The Certificate Map Test group box closes.

Confirm the Validity of Certificates

SOA Security Manager can confirm the validity of certificates using either of the following two methods:

- Certificate Revocation List (CRL) checking
- Online Certificate Status Protocol (OCSP) checking

Certificate Revocation List Checking,

A certificate revocation list (CRL) is a list of revoked X.509 client certificates published by the Certificate Authority (CA) to an LDAP user directory. Comparing certificates against CRLs is one method of ensuring that certificates are valid.

Note: The Policy Server can support CRLs greater than 1.7 MB in size, but cannot verify the status of a certificate using a Certificate Revocation List (CRL) that is larger than 64 KB. This limit is due to the third party libraries that are used to parse CRLs.

SOA Security Manager compares certificates against CRLs stored in an LDAP directory. SOA Security Manager verifies the signature of the CRL by retrieving the CA public certificate from the LDAP directory. SOA Security Manager supports the following RSA algorithms for signature verification:

- MD5
- MD2
- SHA1

CA administrators must keep CRLs up to date. If SOA Security Manager retrieves an expired CRL, all certificates from the CA with the expired CRL will be denied access. If multiple CRLs exist, SOA Security Manager will search for and use the most recent CRL. If a CA's public certificate is not available or your CRL is signed with an unsupported algorithm, you can turn off signature checking during the CRL verification process.

Note: If signature checking is turned off, make sure that the LDAP directory is protected appropriately.

Configure Certificate Revocation List Checking

You configure CRL checking to ensure that a user with an invalid client certificate cannot access a protected resource.

To configure CRL checking

1. Open the certificate mapping.
2. Select Perform CRL Checks from the Certificate Revocation List Checking group box.

   CRL-specific fields and controls open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Select the LDAP user directory that contains the CRL from the CRL Directory list.

   Note: If the LDAP user directory is not in the list, click Create to add the new directory connection.
4. Complete the remaining settings, as necessary, and click Submit

   Certificate revocation list checking is enabled.

Online Certificate Status Protocol Checking

The Online Certificate Status Protocol (OCSP) lets OCSP-enabled applications determine the revocation state of an X.509 client certificate in a more timely manner than is possible with Certificate Revocation Lists. Certificate Revocation Lists can get large and their propagation can become slow. OCSP checking provides real-time status information and lessens network traffic significantly.

During certificate checking, the Policy Server looks for the existence of an Issuer DN in a configuration file: smocsp.conf. If the Issuer DN is found, a certificate status check is made using a certified OCSP 1.0 Responder, which is specified in the smocsp.conf file. If the Issuer DN is not found in the configuration file, the certificate is considered to have passed OCSP checking.

OCSP checking requires:

- An established CA environment
- An OCSP Responder
- One or more LDAP directories containing an OCSP Responder and Certificate Authority public certificates

Note: The Policy Server does not support the OCSP on a Linux platform.

Configure Online Certificate Status Protocol Checking

You configure OCSP checking to ensure that a user with an invalid client certificate cannot access a protected resource.

To configure OCSP certificate status checking

1. Ensure that CRL checking is not enabled for the certificate mapping.
2. Create an smocsp.conf file in the Policy Server config directory:

   siteminder_installation_dir/config

The smocsp.conf file must be an ASCII file containing one or more OCSPResponder records, each with the following format:

[
OCSPResponder
IssuerDN <*IssuerDN*>
[AlternateIssuerDN <*IssuerDN*>]
CACertDir <*Name of User Dir containing CA cert*>
CACertEP <*Entry point in CACertDir containing CA cert*>
ResponderCertDir <*Name of User Dir containing Responder cert*>
ResponderCertEP <*Entry point in ResponderCertDir containing Responder cert*>
ResponderCertAttr <*Directory attribute of Responder cert*>
ResponderLocation <*Server-name of Responder:port #*>
AIAExtension<YES|NO>

]

Consider the following when creating the smocsp.conf file:

- Each OCSPResponder record must start with the tag name OCSPResponder.
- Attribute names are not case sensitive.
- The user directory is the name of the user directory connection listed in the Administrative UI, not the IP address of the physical directory.
- The existence of a value for IssuerDN indicates the existence of an OCSP Responder record|object.
- CACertDir and ResponderCertDir directories should exist as user directories in the Policy Server database.
- All attributes except for AIAExtension and ResponderLocation are required.

  Note: ResponderLocation should exist or AIAExtension should be YES.
- AIAExtension is defaulted to NO.

The priority of AIAExtension and ResponderLocation is as follows:

If	Then
AIAExtension is YES	The AIAExtension is used for validations, if it is found in the certificate. Otherwise, the ResponderLocation is used.
AIAExtension is NO	The ResponderLocation is used, regardless of the value of AIAExtension in the userCertificate in the request.

Following is an example of a smocsp.conf file:

```
[
OCSPResponder IssuerDN C=US,O=U.S. Government,OU=DoD,OU=PKI,CN=DOD CLASS 3 CA-9
CACertDir localhost:389
CACertEP cn=DOD CLASS 3 CA-9,ou=PKI,ou=DoD,o=U.S. Government,c=US  ResponderCertDir localhost:389
ResponderCertEP cn=OCSP,ou=PKI,ou=DoD,o=U.S. Government,c=US ResponderCertAttr cacertificate
ResponderLocation aristotle.jfcom.mil:80
]
```

More information:

Configure Certificate Revocation List Checking (see page 272)

Custom Mapping Expressions

You can use custom mapping expressions for complex multiple attribute mapping. This allows you to specify multiple user attributes that should be extracted from a user DN to establish a certificate mapping.

Note: Custom mapping expressions are also useful when simulating certificate-based authentications through authentication the SOA Security Manager Test Tool.

The syntax for a custom mapping expression is a parsing specification designed to enable full mapping flexibility. It indicates which information to take from the certificate and where it should be applied to in the user directory. The basic syntax is as follows:

```
UserAttribute=%{CertificateAttribute},
UserAttribute2=%{CertificateAttribute}
```

Enable LegacyCertMapping Registry Key

Using LDAP syntax to create search filters that contain logic operators requires you to enable the LegacyCertMapping registry key. Enabling the registry key allows legacy behavior in certificate mapping, which ensures that users are authenticated using the specified LDAP search criteria.

LegacyCertMapping

KeyType: DWORD

Values: 0 (disabled) and 1 (enabled)

Default: 0

To enable the registry key on Windows

1. Navigate to HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\PolicyServer, and open LegacyCertMapping.
2. Edit the KeyType value to REG_DWORD.
3. Edit the Values value to 1.

   Note: If a value other than 0x1 is set, or the registry value does not exist, the registry key is disabled.
4. Save the registry key.

   LegacyCertMapping is enabled, and LDAP search filter syntax can be used with custom mapping.

To enable the registry key on UNIX

1. Open the sm.registry file.
2. Add the following lines to the file:

   HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\
   PolicyServer=XXXXX
   LegacyCertMapping=0X1 REG_DWORD
3. Save the file.

   LegacyCertMapping is enabled, and LDAP search filter syntax can be used with custom mapping.

Example 1

If a user's certificate contains:

SubjectDN: CN=John Smith, UID=JSMITH, OU=development, O=CompanyA

You can specify the following custom mapping:

CN=%{UID}, OU=%{OU}, O=%{O}

The resulting UserDN is:

CN=JSMITH, OU=development, O=CompanyA

Example 2

The custom mapping syntax also handles more complex mappings, as illustrated in the example:

If the user's certificate contains:

Subject DN: CN=John Smith + UID=jsmith
+EMAIL=jsmith@companyA.com, ou=development, o=companyA

You can specify the following custom mapping:

CN=%{CN.CN}+UID=%{CN.UID}, OU=%{O}

The resulting UserDN is:

CN=John Smith+UID=JSMITH, OU=companyA

In the above example, the CN contained multiple attributes. The syntax indicated which components of the CN to take and apply to the UserDN's CN. This was done by specifying "CN.CN or CN.UID" This syntax indicates that the custom expression uses both the CN and UID parts of the CN.

Note: You cannot use the "+" operator to disambiguate multiple attributes in a user directory. The "+" operator is used like any other character in the user DN for a user that is present in the user directory.

Example 3

Static text can be represented in a custom expression by leaving it outside of the bracket notation as shown below.

The user's certificate contains:

Subject DN: CN=John Smith, UID=JSMITH, OU-development

You can specify the following mapping:

CN=%{UID}, OU=%{OU}, O=companyA

The resulting UserDN is:

CN=JSMITH, OU=development, O=CompanyA

For more information, see the next section.

Template String Usage

The template string is composed of text and hash-bracketed expressions %{…}. All text outside the brackets is returned unchanged. The hash-bracketed expressions are evaluated based on the following rules:

- Undistinguished variable names (e.g. DN) are resolved before being returned.
- Distinguished variable names (for example, DN.UID) are resolved to the variable component before being returned.

Map to the Certificate Serial Number or IssuerDN

Certificate Mapping supports mapping of the CertSerialNumber and IssuerDN attributes, which are not part of the subjectDN. These attribute(s) in the subjectDN of user certificates can be mapped to default or custom user-attribute(s), such as UID or CN in the user directory.

To map these attributes, add the following in the Mapping Expressions field in the Certificate Mapping pane:

- CustomAttributeinLDAP1 = %{CertSerialNumber}

Custom Certificate Mapping for Multiple Attributes of the Same Type

Some certificates may have multiple attributes of the same type in their Subject DN. SOA Security Manager supports a simple method for using a custom certificate mapping to see attributes other than the first attribute of a particular type. The syntax is as follows:

%{attribute_name} for the first occurrence of attribute_name

%{attribute_nameN} for the Nth occurrence of attribute_name

If the Subject DN of the certificate contained CN=user,ou=dev,sn=1234,sn=2345,sn=3456,o=company,c=us, you could set up a custom certificate mapping to any of the sn attributes. For example, to map to the first sn, enter %{sn} as the custom mapping. To map to the second sn, you could enter %{sn2} as the custom mapping.

Map to Non-Required Attributes

Sometimes certificates for individuals may be slightly different. For example, some users may have two account numbers, while others have a single number. In these cases, you may want to map to the second of the numbers when a second attribute exists. You can do so using the following notation:

%{attribute_name2/attribute_name}

Using the example from above, you could enter %{SN2/SN} as a custom mapping to indicate that the second number in the Subject DN should be used if it exists, otherwise, the first occurrence of the account number attribute should be used.

This notation can also be used to specify two different attributes that are acceptable for a certificate mapping. For example, to indicate that the SN should be used, but a CN may be used if the SN does not exist, you could enter %{SN/CN}.

Chapter 8: Additional WS-Security Configuration Requirements

This section contains the following topics:

XML Signing and Validation Service (see page 281)
Configuration Requirements for Generating SAML Assertions (see page 296)

XML Signing and Validation Service

If your web service implementation requires SOA Security Manager to sign and validate signed messages with WS-Security tokens, or to produce or consume XML encrypted messages with WS-Security tokens, you must set up the XML Signing and Validation Service.

The XML Signing and Validation Service stores, manages, and retrieves keys and certificates required to sign and validate messages with WS-Security tokens. The service is also responsible for decrypting symmetric XML encryption keys that have been encrypted using the site's public key.

The XML Signing and Validation Service service comprises the following:

- **smkeydatabase**—A flat-file key and certificate database
- **smkeytool**—A command-line utility that lets you populate and manage the SMKeyDatabase

More information:

Smkeydatabase Overview (see page 281)

Smkeydatabase Overview

By default, an empty smkeydatabase is created by the SOA Security Manager installation program as part of the Policy Server install. If you do not create the database at installation, you can use smkeytool to create a new database at any time. You can also use smkeytool to delete and create a new database.

Aliases in the Smkeydatabase

Aliases enable you to easily reference any single certificate or certificate and private key pair in the smkeydatabase. Every certificate or certificate/private key pair in the smkeydatabase must have a unique alias.

Certificate Revocation Lists in the Smkeydatabase

A Certificate Revocation Lists (CRL) is issued by a Certificate Authority to its subscribers. The list contains the serial numbers of subscribers whose digital certificates have been revoked. When a user attempts to access a server, the server allows or denies access based on the CRL entry.

The smkeydatabase needs to point to a current CRL for each root CA certificate to help the Policy Server enforce secure access. To add and maintain a CRL in the smkeydatabase, a series of command options are available with smkeytool utility, which is used to modify the smkeydatabase.

If you are using CRLs, you need to specify the location of a CRL for the smkeydatabase. Updating a CRL differs depending on the CRL type. To update a certificate file, you have to point the smkeydatabase to the most updated file. For LDAP CRLs, once the location of the list is specified, the server administrator can configure the list to be updated automatically.

Note: The CRL feature for the smkeydatabase has no relationship to the SiteMinder client certificate authentication scheme. Federation CRL features must be configured on their own.

The CRL feature for the smkeydatabase supports the following:

- Addition of certificate files or LDAP CRLs
- PEM and DER encodings
- SAML 1.x, SAML 2.0, and WS-Federation protocols

The CRL feature does not support the Online Certificate Status Protocol (OCSP).

You can add a CRL to the smkeydatabase using smkeytool.

To add a CRL to the smkeydatabase

1. Add the CRL file or LDAP CRL to smkeydatabase with the addRevocationList command option.

   Example:

   ```
   smkeytool -addRevocationInfo -issueralias verisignca -type filecrl
   -location c:\crls\verisign_root_ca.crl
   ```

2. Restart the Policy Server.

Formats Supported by the Smkeydatabase

The smkeydatabase supports the following formats:

- Private Keys: Private keys must be in PKCS1, PKCS5, PKCS8 or PKCS12 format and DER or PEM encoded. Only RSA keys are supported.
- Public Certificates: V1, V2 and V3 of the X.509 certificate format are supported. DER, Base64, and PEM encoding formats are supported.

What to Store in Smkeydatabase

The XML Signing and Validation Service requires that you store the following in your Smkeydatabase:

- Your enterprise private key and certificate for signing WS-Security documents and generating X509v3 tokens
- Public key certificates of trusted message issuers for validation

The following table shows exactly which objects you will need to add to your Smkeydatabase to handle your particular WS-Security signing and validation requirements.

Function	WS-Security Token Type	Required Database Objects
Signing	All	Private key and certificate of web service host enterprise
Generating X509 Tokens	X509v3	Private key and certificate of web service host enterprise
Signature Validation	SAML Assertion; Sender-vouches	Certificate of issuing web service consumer application
	SAML Assertion; Holder-of-key	Certificates of XML request subject and issuing web service consumer application.
	X.509v3; Username (if signed)	Certificate of trusted issuer

Smkeydatabase Properties File

The smkeydatabase properties file, smkeydatabase.properties, supplies the XML Signing and Validation Service and smkeytool with essential configuration parameters required to access and manage the smkeydatabase.

If an smkeydatabase is created by the SOA Security Manager installation program as part of Policy Server installation, smkeydatabase.properties is configured automatically with all of the necessary information. If an smkeydatabase is not created by the installation, the file is created, but variable parameter values are not specified.

smkeydatabase.properties is located in the following files:

- *policy_server_install_dir*\Config\properties (Windows)
- *policy_server_install_dir*/Config/properties (UNIX)

Modify this file only to change the following options:

- NativeDBName--specifies name of the key database
- DBLocation--indicates the directory where the key database resides
- DBUpdateFrequencyMinutes--specifies the frequency at which the database is read from the file system.

The smkeydatabase.properties file contains the following settings:

- DBLocation (see page 284)
- NativeDBName (see page 285)
- XMLDocumentOpsImplementation (see page 285)
- AffiliateIXMLSignatureImplementation (see page 285)
- IXMLSignatureImplementation (see page 285)
- EncryptedPassword (see page 285)
- IXMLEncryptDecryptImplementation (see page 286)
- DBUpdateFrequencyMinutes (see page 286)

Descriptions of each setting follow.

DBLocation Setting

Specifies the path to the directory where the database resides.

Enter the location that smkeytool should use when you manually create the database.

Default: *policy_server_home*/smkeydatabase

NativeDBName Setting

Identifies the name of the database.

Specify the name you want smkeytool to use when you create the database.

Default: smkeydatabase

XMLDocumentOpsImplementation Setting

Specifies the Java class that implements the XML signing and validation.

Note: Do not change this value; it is static and preconfigured.

Default: com.ca.smkeydatabase.api.XMLDocumentOpsImpl

AffiliateIXMLSignatureImplementation Setting

Specifies the Java class that implements low-level cryptographic operations for signing and validation.

Note: Do not change this value; it is static and preconfigured.

Default: com.ca.smkeydatabase.api.XMLSignatureApacheImpl

IXMLSignatureImplementation Setting

Specifies the Java class for Transactionminder that implements low-level cryptographic operations for signing and validation.

Note: Do not change this value; it is static and preconfigured.

Default: com.ca.smkeydatabase.api.XMLSignatureApacheImpl

EncryptedPassword Setting

Indicates the smkeydatabase password.

(Encrypted using the policy store key at database creation.) Prior to creating a key database, this entry contains a dummy value.

Default: *encrypted_password_string*

IXMLEncryptDecryptImplementation Setting

Identifies the Java class that implements the encryption and decryption of assertions, Name IDs, and attributes.

Note: Do not change this value; it is static and preconfigured.

Default: com.ca.smkeydatabase.api.XMLEncryptDecryptApacheImpl

DBUpdateFrequencyMinutes Setting

Indicates the frequency at which the database is read from the file system. Specifically, it is the number of minutes after which the in-memory smkeydatabase expires and is reloaded.

Until this interval passes, certificates and keys added, removed, or changed in the database will not affect the Policy Server. If the value is 0, key database caching is disabled entirely. If the value is -1, the cache persists until the Policy Server is restarted.

Default: 60 minutes

Use the Smkeytool Utility to Modify the Key Database

Smkeytool is a command-line utility that lets you populate and manage the smkeydatabase file. The smkeytool utility is installed with the Policy Server.

Smkeytool requires values specified in the smkeydatabase.properties file. Ensure that the file has the proper settings before running smkeytool.

Smkeytool is located in the following directory:

- *policy_server_home*/bin (UNIX)
- *policy_server_home*\bin (Windows)

Use smkeytool to:

- Create and delete a key database

  You can only have one key database per Policy Server. After the database is created, you can add keys and certificates.
- Add and delete your private key
- Add and delete a partner certificate
- Import root certificates of CAs

- Add client certificate keys

  If you are using a root or chain Certificate Authority (CA) at the consuming authority that is not listed in the smkeydatabase file, you need to add it.

- List all certificates stored in the key database
- Export key data from smkeydatabase
- Add, list, validate, and delete a Certificate Revocation List

Note: If you are adding a private key or certificate, delete the certificate metadata from the certificate file before trying to import it into the smkeydatabase. Import only the data starting with the --BEGIN CERTIFICATE-- marker and ending with the --END CERTIFICATE-- marker. Be sure to include the markers.

Smkeytool Command Syntax and Options

Smkeytool is a command-line utility that provides many options to manage the smkeydatabase.

Run the smkeytool utility from a command line, using the following syntax:

UNIX

smkeytool.sh *-option* [*-argument(s)*]

Windows

smkeytool.bat *-option* [*-argument(s)*]

If you enter smkeytool from a command line without any options, you will see a list of all command line options.

The smkeytool utility uses the following command options and arguments:

- createDB
- addPrivateKey
- addCertOption
- addRevocationInfo
- changepassword
- deleteRevocationInfo
- deleteDB
- delete
- export
- findAlias

- importDefaultCACerts
- listCerts
- listRevocationInfo
- printCert
- renameAlias
- validateCert
- help

A description of each command option follows.

More Information:

Smkeytool Command Options (see page 288)

Smkeytool Command Options

The smkeytool utility lets you modify the smkeydatabase. Be aware of the following when making database changes:

- If you are adding a private key or certificate, delete the certificate metadata from the certificate file before trying to import it into the smkeydatabase. Import only the data starting with the --BEGIN CERTIFICATE-- marker and ending with the --END CERTIFICATE-- marker. Be sure to include the markers.
- If you add a new certificate to the key database or update an existing certificate, restart the Policy Server to see the change immediately. If you do not restart the Policy Server, it takes some time before the Policy Server and the key database synchronize. The amount of time for the key database to automatically update depends on the configured frequency of database updates. You can configure database updates by adjusting the DBUpdateFrequencyMinutes parameter in the smkeydatabase.properties file.

The smkeytool command options and arguments are as follows:

-createDB

Creates an new key database to store keys and certificates. By default, the directory is named smkeydatabase. Additionally, an empty alias store is created and named keyaliases.ser file in the smkeydatabase directory. You can change the key store location by modifying the smkeydatabase.properties file.

Important! To store multiple keys in the database, you must define the first key you add with the alias defaultenterpriseprivatekey before you can add subsequent keys.

Options for -createDB are as follows:

-password <*password*>

Required. The password is used to store all data in an encrypted format in the key database. It can be a value from 6 to 32 characters. It is encrypted using the policy store key and added to the smkeydatabase.properties file.

-importDefaultCACerts

Optional. Imports the default Certificate Authority (CA) certificates during the creation of the database. These certificates are imported from the cacerts.keystore file, which is installed with the [set to your product name] and contains all default CA certificates. This option is the same as executing the -importDefaultCACerts option.

-addPrivKey

Adds the specified private keys and corresponding certificates to the key database. You can have multiple private keys and certificates in the database. Only RSA keys are supported. When you use the -addPrivKey command, you can specify the key data by combining the -keyfile and -certfile options or by using the
-keycertfile option alone.

The Policy Server at the producing authority uses a single enterprise private key to sign SAML messages and to decrypt encrypted SAML messages received from the consuming authority. Typically, the enterprise key is the first private key found in the smkeydatabase.

Note: The entire smkeydatabase is encrypted; however, the individual private keys are not.

Options for -addPrivKey are as follows:

-alias <*alias*>

Required. Alias associated with a single certificate in the database. Must be a unique string and should contain only alphanumeric characters.

-certfile <*cert_file*>

Full path to the location of the certificate associated with this private key. Required for keys in PKCS1, PKCS5, and PKCS8 format.

-keyfile <*private_key_file*>

Full path to the location of the the private key file. Required for keys in PKCS1, PKCS5, and PKCS8 format.

-keycertfile <*key_cert_file*>

Full path to the location of the PKCS12 file that contains the private key and public certificate data. Required for keys in PKCS12 format.

-password <*password*>

Optional. Private key files are typically encrypted prior to be added to the smkeydatabase. When added to the smkeydatabase, the key needs to be decrypted. The password value represents the password used to decrypt the private key data. This password is not stored in the smkeydatabase.

-addCert

Adds a certificate to the key database.V1, V2, and V3 versions for X.509 certificate format are supported DER and PEM encoding formats are supported.

Note: For the Policy Server to recognize the new certificate immediately, restart the Policy Server. Otherwise, the database updates based on the frequency you configure in the smkeydatabase.properties file.

If you indicate that you want to trust the certificate as a Certificate Authority, this certificate is always treated as a CA certificate.

Options for -addCert are as follows:

-alias <*alias*>

Required. Alias to the certificate associated with this private key in the database. Must be a unique string and should contain only alphanumeric characters.

-infile <*cert_file*>

Required. Full path to the location of the newly added certificate.

-trustcacert

Optional. Checks that the user provider certificate being added is a CA certificate. Smkeytool checks that the certificate has a digital signature extension and that the certificate has the same IssuerDN and Subject DN values.

-noprompt

Optional. The user will not be prompted to confirm the addition of the certificate.

-addRevocationInfo

Specifies the location of a CRL so the smkeydatabase can locate the list during the SAML authentication process. The smkeydatabase does not store the contents of a CRL, but merely reads the CRL contents when the Policy Server starts and after a refresh interval has elapsed.

Important! If you add a CRL entry to the smkeydatabase, you must restart the Policy Server.

Options for -addRevocationInfo are as follows:

-issueralias <*issuer_alias*>

Required. Alias name of the Certificate Authority who issues the CRL.

Example: -issueralias verisignCA

-type (*ldapcrl* | *filecrl*)

Required. Specifies whether the list is a certificate file or an LDAP CRL. The options are ldapcrl or filecrl.

-location <*location*>

Required. Specifies the location of the CRL. For a file, specify the full path to the file. For an LDAP CRL, specify the full path to the LDAP server node.

Example of file location: -location c:\crls\siteminder_root_ca.crl

Example of LDAP CRL location: -location "http://localhost:880/sn=siteminderroot, dc=crls,dc=com"

-deleteRevocationInfo

Deletes a CRL from the database.

Options for -deleteRevocationInfo are as follows:

-issueralias <*issuer_alias*>

Required. Name of the Certificate Authority who issues the CRL.

-noprompt

Optional. The user will not be prompted to confirm the deletion of the CRL from the database.

-deleteDB

Deletes the smkeydatabase based on configuration data in the smkeydatabase.properties file. All the entries in the key database and the aliases data store file will be deleted.

Option for -deleteDB is as follows:

-noprompt

Optional. The user will not be prompted to confirm the deletion of the database.

-delete

Deletes an existing certificate from the smkeydatabase. If the certificate has an associated private key, the key is also deleted.

Option for -delete is as follows:

-alias <*alias*>

Required. Alias of the certificate to be removed.

-noprompt

Optional. The user will not be prompted to confirm the deletion of the database.

-export

Exports an existing certificate or a private key from the smkeydatabase. Certificate data is exported using PEM encoding. Private key data is exported using DER encoded PKCS8 format.

Options for -export are as follows:

-alias <*alias*>

Required. Identifies the certificate/key to be exported.

-outfile <*out_file*>

Required. Full path to the output certificate/key file.

-type (key|cert)

Optional. Indicates whether a certificate or key is being exported. If no option is specified, a certificate is the default.

-password <*password*>

Required when exporting a private key. Although the entire smkeydatabase is encrypted, individual private keys are stored in unencrypted form. This password encrypts the private key before it is exported.

-importDefaultCACerts

Imports all default trusted Certificate Authority certificates from the cacerts.keystore file, which is installed with the [set to your product name], into the smkeydatabase. Certificate Authority certificates are used to verify the server certificate associated with the producing authority's web server.

Note: For the Policy Server to recognize the updates to the smkeydatabase immediately, restart the Policy Server. Otherwise, the database updates based on the frequency you configure in the smkeydatabase.properties file.

-findAlias

Determines the alias associated with a certificate that is already in the smkeydatabase.

Option for -findAlias is as follows:

-infile <*cert_file*>

Required. Full path to the certificate file associated with the alias you want to find

-password <*password*>

Password required only when a password-protected P12 file is specified as the certificate file.

-listCerts

Lists some metadata of all the certificates stored in key database.

Option for -listCerts is as follows:

-alias <*alias*>

Optional. Lists the metadata details of the certificate and key associated with the alias specified. This option supports the asterisk (*) as a wildcard character. You can use this wildcard at the beginning and/or at the end of an alias value. Always enclose the asterisk in quotes to avoid a command shell from interpreting the wildcard character.

-listRevocationInfo

Displays a list of current CRLs in the smkeydatabase. The -listRevocationInfo option only prints the CRL name, type (file or ldap), and the location of all the CRLs in the database.

Option for -listRevocationInfo is as follows:

-issueralias <*issuer_alias*>

Optional. Name of the Certificate Authority who issues the CRL. This option supports the asterisk (*) as a wildcard character. You can use this wildcard at the beginning and/or at the end of an alias value. Always enclose the asterisk in quotes to avoid a command shell from interpreting the wildcard character.

-printCert

Displays some metadata of the specified certificate. This command is especially useful for UNIX systems, where it is difficult to see the certificate properties.

Options for -printCert are as follows:

-infile <*cert_file*>

Required. Location of the certificate file.

-password <*password*>

Password required only when a password-protected P12 file is specified as the certificate file.

-renameAlias

Renames an existing alias associated with a certificate.

Options for -renameAlias are as follows:

-alias <*current_alias*>

Required. Current alias associated with a certificate.

-newalias <*new_alias*>

Required. New alias name. Value must be a unique string and should contain only alphanumeric characters.

-validateCert

Optional. Indicates whether a certificate is revoked or not.

Option for -validateCert is as follows:

-alias <*alias*>

Required. Alias to the certificate associated with this private key in the database. Must be a unique string and should contain only alphanumeric characters.

-infile <*crl_file*>

Optional. Specifies the CRL file that you want smkeytool to look in for the certificate to validate it.

-help

Shows how to use the smkeytool utility.

Smkeytool Examples for UNIX Platforms

Example: Create a key database

This example shows the command for creating an smkeydatabase:

```
smkeytool.sh -createDB -password siteminderdb
```

Example: Add a private key and certificate

This example shows the command to add a private key and certificate to the smkeydatabase. The example assumes you are running the smkeytool from the directory where the certificates and keys are located, as follows:

```
smkeytool.sh -addPrivkey -password keypswd -alias idp1privkey -keyfile privkey.pkcs8 -certfile sample.crt
```

If you are not running smkeytool from the directory where the certificates and keys are located, you need to specify the full path to directory where these items are located, as follows:

```
smkeytool.sh -addPrivkey -alias privkey1 -keyfile "export/ca/siteminder/certs/
sampleprivkey.pkcs8" -certfile "export/ca/siteminder/certs/samplecert.crt"
```

Example: Add an trusted CA certificate

This example shows the commands required to add a trusted certificate authority certificate:

Important! Before adding a trusted certificate, obtain a CA certificate from a certificate authority.

To add a trusted CA certificate:

1. Check whether it already exists in the consuming authority database by entering:
   smkeytool.sh -listCerts
2. Add the CA certificate by entering:
   smkeytool.sh -addCert -alias -sp1cacert -infile /opt/netegrity/siteminder/certs/sampleCARoot.cer -trustcacert
3. (Optional) Restart the Policy Server to see the change to the key database immediately.

   If you do not restart the Policy Server, it takes some time before the Policy Server and key database synchronize. The amount of time for key database to update automatically depends on the configured frequency of database updates. You can configure database updates by adjusting the DBUpdateFrequencyMinutes parameter in the smkeydatabase.properties file.

Smkeytool Examples for Windows Platforms

Example: Create a key database

This example shows the command for creating an smkeydatabase:

```
smkeytool.bat -createDB -password smdb
```

Example: Add a private key and certificate

This example shows the command to add a private key and certificate to the smkeydatabase. The example assumes you are running the smkeytool from the directory where the certificates and keys are located, as follows:

```
smkeytool.bat -addPrivkey -password keypswd  -alias privkey1
-keyfile sampleprivkey.pkcs8" -certfile samplecert.crt"
```

If you are not running smkeytool from the directory where the certificates and keys are located, you need to specify the full path to directory where these items are located, as follows:

```
smkeytool.bat -addPrivkey -password keypswd -alias privkey1 -keyfile "c:\program
files\ca\siteminder\certs\sampleprivkey.pkcs8"
-certfile "c:\program files\ca\siteminder\certs\samplecert.crt"
```

Example: Add an trusted CA certificate

This example shows the commands required to add a trusted certificate authority certificate:

Important! Before adding a trusted certificate, obtain a CA certificate from a certificate authority.

To add a trusted CA certificate:

1. Check whether it already exists in the consuming authority database by entering:
   smkeytool.sh -listCerts
2. To add the CA certificate enter:
   smkeytool.bat -addCert "c:\program files\ca\siteminder\certs\sampleCARoot.crt" -trustcacert
3. (Optional) Restart the Policy Server to see the change to the key database immediately.

   If you do not restart the Policy Server, it takes some time before the Policy Server synchronize. The amount of time for key database to update automatically depends on the configured frequency of database updates. You can configure database updates by adjusting the DBUpdateFrequencyMinutes parameter in the smkeydatabase.properties file.

Configuration Requirements for Generating SAML Assertions

If your SOA security environment requires you to configure SOA Security Manager to add WS-Security tokens containing SAML assertions to SOAP documents for consumption by other web services, you must first perform some additional configuration using the Federation Security Services Administrative User Interface.

In all SAML token generation situations, you must create an Affiliate domain. Additionally, you must perform the following steps depending on the version of SAML tokens you need to generate:

- To create SAML 1.1 assertions, configure the SAML 1.x Assertion Generator properties file and add Affiliate objects to the Affiliate domain.
- To create SAML 2.0 assertions, add SAML 2.0 Service Provider objects to the Affiliate domain.

Note: Affiliate domains and related SAML token functionality are implemented as part of the CA SiteMinder Federation Security Services on the Policy Server. For more information, see the *CA SiteMinder Federation Security Services Guide*.

SAML 1.x Assertion Generator

If you are configuring SOA Security Manager to add WS-Security tokens containing SAML 1.x assertions to SOAP documents for consumption by other web services, you must configure the SAML Assertion Generator to produce the SAML 1.1 assertions that will be used in those tokens.

Note: The SAML Assertion Generator is a component of CA SiteMinder Federation Security Services on the Policy Server. For more information, see the *CA SiteMinder Federation Security Services Guide*.

The SAML Assertion Generator uses static configuration data from two sources to determine how to construct assertions:

SAML Assertion Generator Properties File

Specifies domain-wide SAML assertion generation parameters

Affiliate Objects

Define a set of parameters for the SAML Assertion Generator

Once configured, the SAML Assertion Generator is triggered to generate an assertion when a WS-Security SAML response (which specifies the affiliate to use to generate the assertion and dynamic information about how the assertion and message should be signed) is triggered by an authorizing policy.

More information:

Affiliate Objects (see page 298)

Configure the AMAssertionGenerator.properties File

The AMAssertionGenerator.properties file contains domain-wide configuration parameters required for generating SAML assertions.

To configure the AMAssertionGenerator.properties file for SOA Security Manager

1. Navigate to the following location: *policy_server_home*/config/properties
2. Open the AMAssertionGenerator.properties file in a text editor.
3. Modify the following parameters:

   AssertionIssuerID

   Specifies the URL of the authentication web service that is issuing the assertion. Must match the Issuer DN in the enterprise certificate. This value is used for unsigned assertions. For example:

   AssertionIssuerID = http://www.acmewidget.com/ordering

SecurityDomain

Specifies the domain name of the enterprise issuing the assertion. For example:

SecurityDomain = www.example.com

SourceID

Not used by SOA Security Manager.

4. Save the file and exit the text editor.
5. Restart the Policy Server. (Changes made to the AmAssertionGenerator.properties file will not be picked up by the Policy Server until it is restarted.)

Affiliate Objects

Affiliate objects define parameters used by the SAML Assertion Generator to produce SAML 1.x assertions for use in WS-Security SAML tokens.

To configure affiliate domains and affiliate objects, follow the associated procedures in the CA SiteMinder Federation Security Services Guide. However, because SOA Security Manager does not use the affiliate object to define an affiliate organization, you do not need to specify all the options.

Note: When you configure an affiliate object for use by SOA Security Manager, you are *not* defining an affiliate organization for which the assertion is intended. Assertions generated for SOA Security Manager can be sent to any web service protected by the WS-Security authentication scheme (or similarly capable third-party security application).

The following table summarizes all the affiliate configuration parameters. The table describes each parameter's function for generating SAML assertions for SOA Security Manager, where the parameter is required, or tells you if the parameter is not required.

Affiliate Dialog Element	Field Name	Purpose for SOA Security Manager SAML Assertion Production
Main panel	Name	Specifies the name of the affiliate object (must be unique across all affiliate domains). This name is referenced by WS-Security policy responses (by defining a txm_wssec_saml_affiliate attribute whose value matches the name of the affiliate object).
	Description	Not used by SOA Security Manager

Affiliate Dialog Element	Field Name	Purpose for SOA Security Manager SAML Assertion Production
	Password	Not used by SOA Security Manager
	Enabled	Sets the Enabled check box to activate the affiliate object. This option must be set for SOA Security Manager to produce SAML 1.x assertions.
	Allow Notification	Not used by SOA Security Manager.
	Authentication URL	Not used by SOA Security Manager.
Users tab	Select users	Specifies the users and groups (from the user directory or directories defined in the affiliate domain) for whom assertions should be generated.
Assertions tab (Optional)	Audience	Specifies the URI of a document that describes the terms and conditions of the agreement between the token issuer and consumer. This value is added to the assertion and can be used for authentication purposes. (If a request's assertion token contains an audience value, that value must match one specified in the WS-Security scheme for the request to be authenticated.) Additionally, the web service can parse the actual audience document to obtain additional information.
	Validity duration	Specifies the amount of time, in seconds, that the assertion will be valid.
	Skew time	Specifies the difference, in seconds, between the system clock time of the SAML assertion producer and the system clock time of the SAML assertion consumer.
Session tab	Shared sessioning	Not used by SOA Security Manager (leave option unset).
	Sync interval	Not used by SOA Security Manager (leave blank).

Affiliate Dialog Element	Field Name	Purpose for SOA Security Manager SAML Assertion Production
Attributes tab (Optional)	Affiliate Attribute dialog (Opened from Attributes tab by clicking Create button)	Not required for SOA Security Manager assertion production. However, if specified, an attribute statement will be included in the assertion that can be used for use in authentication and authorization decisions.
IP addresses tab (Optional)	Add an IP Address dialog (Opened from IP Addresses tab by clicking Add button)	Specifies the list of IP addresses that are allowed to generate SAML assertions.
Time restrictions tab (Optional)	N/A	Specifies times when assertion can be issued

More information:

Variables for Generating SAML Tokens (see page 368)

Configure SAML 2.0 Service Providers

SAML Service Provider objects define parameters used by the SAML Assertion Generator to produce SAML 2.0 assertions for use in WS-Security SAML tokens.

Note: When you configure a service provider object for use by SOA Security Manager, you are *not* defining a service provider organization for which the assertion is intended. Assertions generated for SOA Security Manager can be sent to any web service protected by the WS-Security authentication scheme (or similarly capable third-party security application).

To configure SAML Service Provider objects, generally follow the associated procedures in the CA SiteMinder Federation Security Services Guide. However, because SOA Security Manager does not use the affiliate object to define a service provider organization, you do not need to specify all the options. Fields whose use is different for use by SOA Security Manager are described below.

Name

Specifies the name of the service provider object (must be unique across all affiliate domains). This name is referenced by WS-Security policy responses (by defining a txm_wssec_saml_affiliate attribute whose value matches the name of the affiliate object).

Enabled

Sets the Enabled check box to activate the service provider object. This option must be set for SOA Security Manager to produce SAML 2.0 assertions.

Authentication URL

Not used by SOA Security Manager. However, a valid value *is* required. CA recommends using "http:\\localhost\"

Application URL

Not used by SOA Security Manager.

SSO Tab: Bindings

Choose the HTTP-Post option.

More information:

Variables for Generating SAML Tokens (see page 368)

Chapter 9: Configure Security Policies from WSDL Files Using Enterprise Policy Management

This section contains the following topics:

Application Overview (see page 303)
Create an Application (see page 306)
(Optional) Configure Application Responses (see page 307)
Secure Web Service Resources from WSDL Files (see page 308)
Modify the Default Role to Define User Access Rights (see page 310)
Create Additional Roles to Define User Access Rights (see page 311)
Modify Role Assignments in the Application Policy (see page 312)

Application Overview

SOA Security Manager enterprise policy management uses application objects to provide an intuitive method for creating and managing security policies for the web services in your SOA environment.

An application defines a complete security policy for one or more related web services. For example, an organization that divides its web service resources by business unit might create an application for marketing, a separate application for engineering, and so on. Applications associate resources with user *roles* to specify entitlement policies that determine what web service users can access what web service application resources. Roles identify the set of users who have access to a resource or group of resources in terms of a named or unnamed expression.

When using application objects, you are only required to provide data for configuration settings that do not have defaults; modifying other settings is optional. You can manipulate additional settings that allow you to define more fine-grained protection of an application; however, this is not required.

For the administrator already familiar with traditional policy management, there is a relationship between the application-oriented concepts and the underlying Policy Server objects, which is reflected in the Administrative UI. The following table shows this relationship.

Application Dialogs and Group Boxes	Equivalent Policy Server Objects
General application settings	Policy domain
Components	Realms
Resources	Rules
Application Roles	Replaces the function of user directory lookups in authorization polcies

More information:

Policy Domain Overview (see page 323)
Realms Overview (see page 329)
Rules Overview (see page 337)

Administrative Rights to Create Application Security Policies

SOA Security Manager uses an administrative model that lets you determine what different administrators can view and manipulate.

To implement application security policies, you need to have the necessary administrative rights. An administrator can be assigned the following rights related to applications:

- application administration

  The application administration privilege lets you create, modify and delete an application and its components.

- policy administration

  The policy administration privilege lets you define the resources, roles, and policies associated with an application.

To assign the application and policy administration rights

1. Click Administration, Administrator, Create Administrator.

   The Create Administrator pane opens.

2. Enter a name for the administrator in the Name field.

3. Click Create in the Rights group box.

   The Rights dialog opens.

4. In the table, choose one or more of the following:
   - application administration
   - policy administration
5. Click OK.

The administrator has been given application and policy administration privileges.

How to Create Application Security Policies

To protect applications in your organization, you create application security policies. These policies define the resources you want protected and specify who is allowed access to the protected application.

To create application security policies, use the following process:

1. Create an application object for the web service resources that you want to protect.
2. Create a new user directory or associate an existing user directory with the application.
3. (Optional) Configure any responses that you want to associate with web service resources.
4. Generate security policies from the web service definition contained in its WSDL file.

   The Administrative UI creates component and resource definitions corresponding to your settings for all specified web service ports and operations, a default application role (that defines *no* user access), and a security policy that binds that default role with resources.
5. Modify the default application role to define a group of users that can access a resource to which the role is assigned. Application roles are defined by expressions that search the user directories for users that meet the membership criteria of the application role.
6. (Optional) Create additional application roles that identify other groups of users that should have access to any of the protected resources.
7. Repeat Steps 4, 5, and 6 for any additional web services defined in other WSDL files that you want to protect in the same application.
8. Refine the generated policies by associating application roles with resources.

Create an Application

The application object you create for one or more related web services must specify the top-level location of the resources that you want to protect, and a directory of users who are authorized to use the resources.

To identify the application and select the directory server

1. Click Policies, Application, Create Application.

   The Create Application pane opens.

2. Enter values for the fields in the General group box. Choose distinctive values that help you remember its purpose or function, as shown in the following examples:

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

   Name

   Name of the application

   Description

   (Optional) A description of the application.

3. In the Components group box, specify values for a default component description.

   Note: These fields are mandatory, but the component they define is not used; component definitions for your web services will be created from their WSDL files.

   Agent Type

   Web Agent

   Agent

   Any SOA Agent.

   Resource Filter

   *

4. Accept the defaults for the remaining settings.

5. In the User Directories group box, click Add/Remove.

   The Choose User Directories dialog opens.

6. Select one or more directories that contain the the users that you want to be access the web service resources then click the right arrow to move the selected directory or directories from the Available members column to the Selected Members column.

7. Click OK.

   You return to the General tab.

8. Click Submit.

   The application is identified and the directory selected.

(Optional) Configure Application Responses

To include a response (for example, to generate WS-Security headers) in the application security policy you generate from a WSDL file, first configure the response.

To create a response:

1. Click on the Response tab from the Application dialog.
2. Click Create Response.

   The Create Application Response pane opens.

3. Type the name of the response in the General group box.
4. Click Create Response Attribute to create a response attribute and add it to the attribute list.
5. The Create Response Attribute pane opens. Select a response attribute from the Attribute drop-down list.

   a. Select an attribute type on the Attribute Kind (one of Static, User Attribute, DN Attribute, and Active Response) group box.

      The fields on the Attribute Fields group box are updated to match the specified attribute type.

   b. Complete the fields on the Attribute Fields group box.

      Note: For WebAgent-SAML-Session-Ticket-Variable and WebAgent-WS-Security-Token attributes, you can either enter values directly in the Variable Name and Variable Value fields or populate those fields with valid values from the Select a Name and Select a Value lists that appear.

c. Specify Cache Value or Recalculate value every ... seconds on the Attribute Caching group box.

d. Click Submit.

The Create Response Attribute Task is submitted for processing, and the response attribute is added to the Attribute List on the Create Response Attribute pane.

6. Create further response attributes as required.
7. Click OK.

The Create Response Task is submitted for processing and you are returned to the Responses tab.

More information:

SAML Session Ticket Response Attribute Variables (see page 360)
Variable for Specifying the Generated WS-Security Token Type (see page 366)
Variables for Generating Username and Password and X.509 Certificate Tokens (see page 367)
Variables for Generating SAML Tokens (see page 368)
Variables for Encrypting/Decrypting WS-Security Messages (see page 371)
Variables for Handling WS-Security Headers (see page 373)

Secure Web Service Resources from WSDL Files

After you have created the application object, you generate the following settings required to protect web service resources from their WSDL files:

- Component and resource definitions for the web service resources
- A default role that which denies all access, but which you can associate resources with groups of users
- A policy that binds the default role to resources to define how authorization decisions are made when web service consumers interact with those resources.

To create the web service resources and security policy

1. Click the SOA tab
2. Click Secure Web Services from WSDL.

The Secure Web Services from WSDL: Select Application pane appears.

3. Select the application to secure from the Choose an Existing Application list.
4. Click Next.

The Secure Web Services from WSDL: Input WSDL pane appears.

5. Specify whether you want to open a WSDL file that resides on your local system or at a specific URL by selecting the File or URL option, and identifying the file accordingly as follows:
   - If you chose FILE, click Browse and navigate to its location.
   - If you chose URL, type the URL.in the Enter the WSDL URL field. For example,

     http://example.com/WSDL/my-wsdl.wsdl
6. Click Next.

   The Secure Web Services from WSDL: Define Policies pane appears, displaying a selectable table of the web services (ports) defined in the WSDL file.
7. Define the web service(s) to protect in the Define Web Service Protection Policy table:
   - Check the web service or services that you want to protect in the Port Name column.
   - Assign the SOA Agent that will protect each protected web service from the Agent list.
   - Assign an authentication scheme to use to protect each web service from the Authentication Scheme list.
   - (Optional) Choose a response to bind to a web service from the Response list.
8. (Optional) Set the Propagate Authentication Scheme of Web Service to all its operations option to apply the authentication scheme you assigned to protect each web service to all of its constituent operations.
9. Click on a web service entry in the Port Name column to drill down to see its constituent operations in the Define Web Service Protection Policy table and select individual operations to protect, authentication schemes to use, and optionally, response bindings.

   (To return to the top-level WSDL view, click the All Web Services link at the top-left corner of the table.)
10. When your policy definitions are complete, click Next.

    The Secure Web Services from WSDL: Summary pane opens, displaying a summary of the components, sub-components, and resources that will be created according to your selections.
11. If the summary is correct, click Finish.

    The Administrative UI creates component and resource definitions corresponding to your settings for all specified web service ports and operations, a default application role (that defines *no* user access), and a security policy that binds that default role with resources.

However, if you assigned different authentication schemes to a web service port and any of its operations, you will need to manually create a resource definition for that web service port:

a. Click Policies, Application, Modify Application.

   The Modify Application pane opens

b. Specify search criteria, and click Search.

   A list of applications that match the search criteria opens.

c. Select your application from the list, and click Select.

   The Modify Object: *Name* pane opens.

d. Click on the Resources Tab.

e. Choose the appropriate entry for the web service port from the Select a context root pulldown. No resources should be listed.

f. Click Create.

   The Create Application Resource pane opens.

   Specifiy a name for the resource, accept the default resource filter (/*) and select the ProcessSOAP and ProcessXML Web Agent actions.

g. Click OK.

h. Click Submit.

The web services you chose to protect are now secure. No access requests will be authorized until you modify the default role to define access privileges or create more roles and bind them to resources in the authorization policy.

Note: You can repeat this procedure to add the resources from multiple WSDL files to the same application.

More information:

Create an Application (see page 306)
(Optional) Configure Application Responses (see page 307)
Modify the Default Role to Define User Access Rights (see page 310)
Create Additional Roles to Define User Access Rights (see page 311)
Modify Role Assignments in the Application Policy (see page 312)

Modify the Default Role to Define User Access Rights

Roles associate resources with groups of users must be created.The Administrative UI creates a default role that allows no access when it secures web services from a WSDL file. You must modify this role to define a group of users that can access a resource to which the role is assigned.

To create a new role

1. Click Policies, Application, Modify Application.

   The Modify Application pane opens

2. Specify search criteria, and click Search.

   A list of applications that match the search criteria opens.

3. Select your application from the list, and click Select.

   The Modify Object: *Name* pane opens.

4. Click the Roles tab.

5. Click the Edit button beside the default role.

6. Ensure the Create a new object of type Role button is selected, and then click OK.

   The Modify Role pane opens.

7. Type an expression that defines the group of users that can access a resource to which the role is assigned in the Expression field or use the Expression Editor to complete. To access the Expression Editor, click Edit.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

8. Click OK.

   The role is modified.

Create Additional Roles to Define User Access Rights

Roles associate resources with groups of users must be created. The Administrative UI creates a default role which is assigned to all resources in when it secures web services from a WSDL file. If required, you can create additional roles.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To create a new role

1. Click Policies, Application, Modify Application.

   The Modify Application pane opens

2. Specify search criteria, and click Search.

   A list of applications that match the search criteria opens.

3. Select your application from the list, and click Select.

   The Modify Object: *Name* pane opens.

4. Click the Roles tab.

5. Click Create.

6. Ensure the Create a new object of type Role button is selected, and then click OK.

   The Create Role pane opens.

7. Enter values for the fields in the General group box. Choose distinctive values that help you remember its purpose or function, as shown in the following examples:

   Name

   Name of the role.

   Description

   Description of the role.

   Expression

   The expression that defines the group of users that can access a resource to which a role is assigned.

   You can use the Expression Editor to complete this field or type in an expression. To access the Expression Editor, click Edit.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

8. Click OK.

   The role is created.

Modify Role Assignments in the Application Policy

The Administrative UI generates an application security policy that binds the web service resources specified in a WSDL to a default. You can modify this policy to change the roles assigned to resources to allow different groups of users to access different resources protected by the application.

To modify the role assignments in an application security policy

1. Click Policies, Application, Modify Application.

   The Modify Application pane opens

2. Specify search criteria, and click Search.

   A list of applications that match the search criteria opens.

3. Select your application from the list, and click Select.

   The Modify Object: *Name* pane opens.

4. Click the Policies tab.

   The Policies pane opens and displays a table listing the configured resources and available roles. This table lets you quickly see which roles can be granted access to which resources.

5. Place or remove checks in the role column to set the required role assignments for each web service resource.

   For example, if you had a human resources application that secures a web service for benefits management and another for performance appraisals and separate roles for employees and managers, you could:

   a. Check the Employees role beside the rows of resources that protect the benefits management operations to create a policy that allows employees to manage their benefits.

   b. Check the Managers beside the rows of resources that protect the performance appraisals to create a policy that allows only managers to access the performance appraisals web service.

6. Click Submit.

   Security policies are created for each role assigned.

Note: If you need to edit resources or roles, you must make the changes on the respective tabs and not on the Policies pane.

Chapter 10: Configure Security Policies Using Traditional Policy Management

This section contains the following topics:

Traditional Policy Management Overview (see page 315)
How to Identify a Web Service Resource by Agent, Realm, and Rule (see page 316)
Guided Example: Create Security Policies from a WSDL File (see page 320)

Traditional Policy Management Overview

Traditional policy management using policy domains and domain objects allows you to perform manual configuration of security policies for web service resources and *must* be used to modify policies migrated from a previous SOA Security Manager deployment or to create policies that implement content-based authorization using variables.

The remainder of this chapter gives advice about how to best define traditional policies to protect web service resources. The following chapters describe all the policy object types and how to configure them in detail.

Important! While traditional policy management provides all the same policy objects as in previous releases, the user interface is different — you must use the Administrative UI; even if available, you must not use the Policy Server User Interface to create new or manipulate existing policies.

More information:

Policy Domain Overview (see page 323)
Realms Overview (see page 329)
Rules Overview (see page 337)
Rule Group Overview (see page 347)
Responses Overview (see page 351)
Policy Overview (see page 385)
Variables Overview (see page 423)
Global Policies (see page 439)

How to Identify a Web Service Resource by Agent, Realm, and Rule

The Resource field in a SOA Security Manager rule specifies the resource that is the subject of the rule. The complete resource specification (shown by the Effective Resource field on the Rule dialog box) is a concatenation of the values of the Agent, the Resource Filter of the parent realm (or realms in a nested realm environment), and the Resource field of the rule itself:

[*agent*] [*realm_resource_filter*] [*rule_resource*]

agent

Specifies a SOA Agent that monitors a server or gateway that contains one or more realms of protected web service resources.

realm_resource_filter

Specifies a string that specifies the resources covered by the realm. If the realm is a top-level realm, specify the resources relative to the server that serves up the files or application. If the realm is nested, specify the resources relative to the parent realm.

rule_resource

Specifies a string or regular expression that specifies the resources to which the rule applies. Specify the resources relative to the realm containing the resource. You can use wildcards (for example, "*") to broaden the specification of a rule.

How SOA Agent for Web Servers Identifies Web Service Resources

By default, the SOA Agent for Web Servers identifies a web service being requested by extracting the binding URL and name of the web service and concatenating them as follows:

[*agent*] [/*web_service_URL*] [/*web_service_name*]

However, the SOA Agent for Web Servers can be configured to perform fine-grain resource identification, in which case it additionally identifies the web service operation being requested:

[*agent*] [/*web_service_URL*] [/*web_service_name*] [/*web_service_operation*]

How Other SOA Agent Types Identify Web Service Resources

This topic describes how the following SOA Agent types identify web service resources:

- SOA Agent for IBM WebSphere
- SOA Agent for BEA WebLogic
-
- SOA Security Gateway

If a request is received over HTTP(S) transport, these SOA Agent types identify the web services being requested by extracting the binding URL, the name of the web service, and the name of the web service operation and concatenating them as follows:

[*agent*] [*/web_service_URL*] [*/web_service_name*] [*/web_service_operation*]

If a request is received over JMS transport, these SOA Agent types identify the web services being requested by extracting the JMS queue or topic name and the name of the web service operation and concatenating them as follows

[*agent*] [*/queue_or_topic_name]* [*/web_service_operation*]

Resource Identification Policy Examples

Coarse-Grain Resource Identification Over HTTP Example

Say you want to protect a resource with the following properties.

- The resource is hosted on an IIS web server on host **soap** in domain **example.com** that is protected by a SOA agent called **MySoaAgent**.
- MyIISSoaAgent is configured to provide coarse-grain resource identification
- The resource is accessible over HTTP transport
- Web service URL is **services/soap2**.
- Web service name is **ExampleSearchService**.
- ExampleSearchService provides two operations:
  - **KeywordSearchRequest**
  - **PowerSearchRequest**

To protect ExampleSearchService, configure the following:

- A realm for ExampleSearchService with Resource Filter value "/services/soap2/ExampleSearchService"
- A single rule in the ExampleSearchService realm with Resource value "*"

Fine-Grain Resource Identification Over HTTP Example

Say you want to protect a resource with the following properties.

- The resource is hosted on an IBM WebSphere Application Server on host **soap** in domain **example.com** that is protected by a SOA agent called **MyWSSoaAgent**.
- MyWSSoaAgent provides fine-grain resource identification
- The resource is accessible over HTTP transport
- Web service URL is **services/soap2**.
- Web service name is **ExampleSearchService**.
- ExampleSearchService provides two operations:
  - **KeywordSearchRequest**
  - **PowerSearchRequest**

To protect ExampleSearchService, configure the following:

- A realm for ExampleSearchService with Resource Filter value "/services/soap2/ExampleSearchService"
- One rule in the ExampleSearchService realm for each operation:
  - A rule for the KeywordSearchRequest operation with Resource value "/KeywordSearchRequest"
  - A rule for the PowerSearchRequest operation with Resource value "/PowerSearchRequest"

Fine-Grain Resource Identification Over JMS Example

Say you want to protect a resource with the following properties.

- The resource is hosted on BEA WebLogic Server on host **soap** in domain **example.com** that is protected by a SOA agent called **MyWebLogicSoaAgent**.
- MyWebLogicSoaAgent provides fine-grain resource identification
- The resource is accessible over JMS transport
- JMS queue name is **ExampleQueue**

- Web service name is **ExampleSearchService**.
- ExampleSearchService provides two operations:
  - **KeywordSearchRequest**
  - **PowerSearchRequest**

To protect ExampleSearchService, configure the following:

- A realm for ExampleSearchService with Resource Filter value "/ExampleQueue"
- One rule in the ExampleSearchService realm for each operation:
  - A rule for the KeywordSearchRequest operation with Resource value "/KeywordSearchRequest"
  - A rule for the PowerSearchRequest operation with Resource value "/PowerSearchRequest"

Unprotected Realms, Rules, and Policies

By default a realm is created in a protected state. In most cases, you should use protected realms instead of changing a realm to an Unprotected state. In a protected realm, all resources are protected against access. To allow access, a rule must be defined, then included in a policy.

When you create a realm in an unprotected state, you must configure rules before SOA Security Manager protects the resources in the realm. If you create a rule for resources in the unprotected realm, only the specified resources are protected. Once the resource is protected, the rule must be added to a policy to allow users to access the resource. You may want to use an unprotected realm if only a subset of the resources in a realm need to be protected from unauthorized access.

The following is an example of the actions required when setting up an Unprotected realm:

Action	Protection State
Create unprotected realm called Realm1 with the Resource Filter: /dir.	Resources contained in /dir and subdirectories are not protected.
Create Rule1 in Realm1 for the resource: getCachedQuote.asp.	The /dir/getCachedQuote.asp resource is protected, but the rest of the contents of /dir are not protected.

Create Policy1 and bind Rule1 and User1 to the Policy.	User1 can access /dir/getCachedQuote.asp. All other users cannot access the protected file.

Guided Example: Create Security Policies from a WSDL File

Deployed web services are typically described in an associated Web Services Description Language (WSDL) file. One way of getting started creating security policies using tradtional policy management, especially in terms of creating realms, rules, and the resource mappings they define, is to work from the WSDL file associated with a deployed web service.

To create security policies from the WSDL file for a deployed web service

1. Parse the WSDL file for the web service you want to secure. Look for <service> elements. A <service> element contains the web service <port> elements which need to be secured. The name attribute of a <port> element identities the port type (and hence contains a reference to a <portType> element). A <port> element also contains the binding URL which refers to the URL where the web service is located. The web service port is protected by creating a realm whose Resource Filter is the combination of the binding URL and port name.

   In the following snippet from ExampleSearch.wsdl, the web service port to secure is ExampleSearchPort. This port is bound to the URL http://api.example.com/*search/beta2.*

   ```
   <service name="ExampleSearchService">
     <port name="ExampleSearchPort" binding="typens:ExampleSearchBinding">
       <soap:address location="http://api.example.com/search/beta2"/>
     </port>
   </service>
   ```

2. To protect the ExampleSearchPort web service, create a realm named ExampleSearchRealm whose Resource Filter is /search/beta2/. Choose a SOA agent and authentication scheme with which to secure this realm as appropriate.

3. Repeat Step 2 (create a realm) for every <port> element contained within the <service> element in the WSDL file.

4. Look for all the web service operations that are available under the above web service port by looking for the <portType> element whose name matches the name of the <port> element.

In the following snippet, the three web service operations to secure are doGetCachedPage, doSpellingSuggestion and doExampleSearch. All these three operations are children of ExampleSearchPort which has been secured by the realm named ExampleSearchRealm.

```
<portType name="ExampleSearchPort">
 <operation name="doGetCachedPage">
  <input message="typens:doGetCachedPage"/>
  <output message="typens:doGetCachedPageResponse"/>
 </operation>

 <operation name="doSpellingSuggestion">
  <input message="typens:doSpellingSuggestion"/>
  <output message="typens:doSpellingSuggestionResponse"/>
 </operation>

 <operation name="doExampleSearch">
  <input message="typens:doExampleSearch"/>
  <output message="typens:doExampleSearchResponse"/>
 </operation>

</portType>
```

5. To configure fine-grain authorization policies, you must secure every child <operation> element. Create a rule under the ExampleSearchRealm realm for each operation with the following properties:

   Resource Filter: "/*Web Service Operation Name*"

   Action: Post, ProcessSOAP and ProcessXML Web Agent actions

6. Create a policy containing the rules you created for every web service operation in the WSDL; assign users to the policy, as required.

Chapter 11: Domains

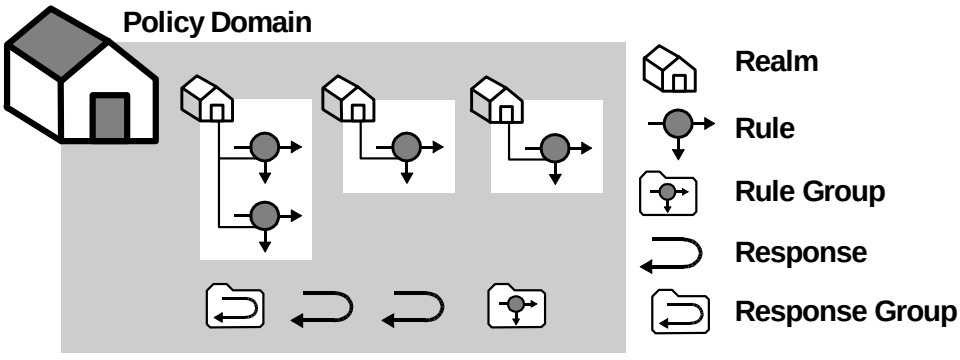
This section contains the following topics:

Policy Domain Overview (see page 323)
Domains and User Membership (see page 324)
How to Configure a Policy Domain (see page 324)
Disable Global Policy Processing for a Domain (see page 327)
Modify a Domain (see page 327)
Delete a Domain (see page 327)

Policy Domain Overview

A policy domain is a logical grouping of resources associated with one or more user directories. In addition, policy domains require one or more administrator accounts that can make changes to the objects within the policy domain. Policy domains contain realms, rules, responses, and policies (and optionally, rule groups and response groups). An administrator with the appropriate privileges assigns a policy domain to one or more administrators.

The resources in a policy domain can be grouped in one or more realms. A realm is a set of resources with a common security (authentication) requirement. Access to resources is controlled by rules, which are associated with the realm that contains the resource. The following diagram illustrates a small policy domain which contains realms and their associated rules, as well as a rule group, response group, and a pair of responses.



By grouping realms and rules in a policy domain, you can provide organizations with a secure domain for their resources. In the policies section, you learn how to create policies within a policy domain to control access to the policy domain's resources.

For example, a Marketing policy administrator who is specified in a Marketing policy domain can manage the Marketing Strategy and Marketing Projects realms. An Engineering administrator, who does not have administrative privileges to manage the Marketing policy domain, cannot control resources belonging to the Marketing department. However, the Marketing policy domain can be associated with a user directory that contains engineering users.

More information:

Policies (see page 385)

Domains and User Membership

Besides acting as a container for domain objects, policy domains also connect to user directories. The Policy Server authenticates users based on the requirements of the realm in which the target resource resides. In order to authenticate a user, the Policy Server must find the user directory where a user is defined. The Policy Server does this by locating the policy domain to which a realm belongs. From the policy domain, the Policy Server queries the user directories specified in the policy domain's search order.

The search order is defined when you add user directory connections to a policy domain. The order in which you add directory connections determines the order that the Policy Server uses to search for a user. For example, if you set up policy domain for a company migrating user data from a WinNT directory to an LDAP directory, and you want the Policy Server to search in the new LDAP directory first, then look in the WinNT user directory, add the LDAP directory connection to the policy domain first, then add the WinNT user directory connection.

How to Configure a Policy Domain

You configure a domain to create a logical grouping of resources with one or more user directories. Configuring a domain requires you to:

- Assign one or more user directories for user authentication
- Create one or more realms to group resources according to security policies

Note: You can edit a policy domain's properties if you need to add a realm in the future.

The following process lists the steps for configuring a new policy domain:

1. Configure the Policy Domain
2. Assign User Directories
3. Create a Realm

More information:

Realms (see page 329)
Start the Administrative UI (see page 42)

Configure a Policy Domain

You can create a policy domain that protects logical groupings of resources.

To create a policy domain

1. Click Policies, Domains.
2. Click Domain, Create Domain.

   The Create Domain pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Type the name and a description of the policy in the fields on the General group box.
4. Add User Directories and Realms.
5. Click Submit.

   The Create Domain Task is submitted for processing.

Assign User Directories

You can add one or more user directories to a policy domain. The Policy Server authenticates users by comparing the credentials that they enter to the credentials that are stored in the user directories. The Policy Server searches the user directories in the same order that they are listed in the policy domain.

To add user directories to a policy domain

1. Click Add/Remove on the User Directories group box on the Domain pane.

   The Choose user directories pane opens.
2. Select one or more user directories from the list of Available Members, and click the right-facing arrows.

   The user directories are removed from the list of Available Members and added to the list of Selected Members.

   Note: To select more than one member at one time, hold down the Ctrl key while you click on the additional members. To select a block of members, click on the first member and then hold down the Shift key while you click on the last member in the block.

3. Click OK.

   The selected user directories are added to the domain.

   Note: To create a new user directory and add it to the domain, click New... on the User Directories group box on the Domain pane.

Create a Realm

When you create a domain, you can create one or more realms in the domain and associate them with a SOA Agent or Agent group. Realms use resource filters to group resources that have similar security requirements and share a common authentication scheme.

To create a realm in a domain and associate it with a SOA Agent or Agent group

1. Click the Realms tab on the Domain pane, New Realm, OK.

   The Create Realm pane opens.

2. Type the name and a description of the realm in the fields on the General group box.
3. Select a SOA Agent or Agent group from the Agent drop-down list and specify the remaining resource properties on the Resource group box.
4. Create new rules or delete existing rules on the Rules group box.
5. Create new sub-realms or delete existing sub-realms on the Sub-Realms group box.
6. Specify the session properties on the Session group box.
7. Specify a registration scheme, directory mapping, events processing, or a combination on the Advanced group box.
8. Click Finish.

   The Create Realm Task is submitted for processing.

Disable Global Policy Processing for a Domain

Global policies let you associate responses with particular resources and events across all domains. By default, global policies apply to all of the resources in a policy domain.

To disable global policies for a specific domain

1. Open the domain.
2. Clear the Global Policies Apply check box, and then click Submit.

   Global policies no longer apply to the resources in this domain.

More information:

Global Policies, Rules, and Responses (see page 439)

Modify a Domain

You can change the name, description, user directory connections and administrators associated with a policy or affiliate domain. All other features of a domain are a result of peripheral configuration.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Delete a Domain

Important! Deleting a domain destroys all of the domain user directory and administrator connections and objects: rules, rule groups, realms, responses, response groups, and policies, or affiliates contained in the domain.

It may take a short amount of time for all deleted objects to be removed from caches.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Chapter 12: Realms

This section contains the following topics:

Realms Overview (see page 329)
Nested Realms (see page 330)
Realms in Request Processing (see page 331)
Configure a Realm (see page 331)
Modify a Realm (see page 333)
Delete a Realm (see page 333)
Configure a Nested Realm (see page 333)
Flush a Single Realm from the Resource Cache (see page 334)

Realms Overview

Complex sets of resources must be logically grouped so that security policies can be created. The basic SOA Security Manager groupings for resources are realms.

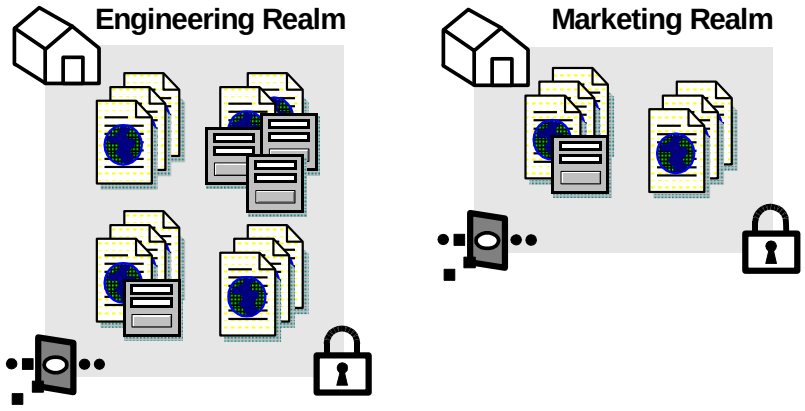
A realm is a cluster of resources within a policy domain grouped according to security requirements. A realm is usually defined for resources that reside in a common location on your network. For example, web services that process incoming purchase orders and are bound to URLs in an /Orders directory or accessible by an Orders JMS queue might be configured as a realm in a policy domain managed by an administrator in a company's order fulfillment organization. The realm also specifies the authentication scheme to be used. Typically, you should create a realm for each web service (as defined by a WSDL **port** element).

Web service resources are identified in the realm according to whether they will be accessed by HTTP(S) or JMS transports as follows:

- HTTP(S): Resources are identified by the URLs to which the web service is bound
- JMS: Resources are identified by the JMS queue or topic name serving the web service

The contents of a realm are protected by SOA Agents. When web service clients request resources within a realm, the associated SOA Agent handles authentication and authorization of the user associated with the request. The realm determines the method of authentication.

The following diagram shows the contents of two realms.



Each of the realms contains web service resources. In addition, each realm is associated with a SOA Agent and an authentication scheme.

A realm is usually defined for resources that reside in a common location on your network. For example, web services that process incoming purchase orders and are bound to URLs in an /Orders directory or accessible by an Orders JMS queue might be configured as a realm in a policy domain managed by an administrator in a company's order fulfillment organization. The realm also specifies the authentication scheme to be used. Typically, you should create a realm for each web service (as defined by a WSDL **port** element).

Web service resources are identified in the realm according to whether they will be accessed by HTTP(S) or JMS transports as follows:

- HTTP(S): Resources are identified by the URLs to which the web service is bound
- JMS: Resources are identified by the JMS queue or topic name serving the web service

Also, choose a SOA Agent to protect these resources and bind these resources to the same authentication scheme

Nested Realms

In SOA Security Manager, realms represent groups of resources in much the same way that directories of files and folders represent a file system's contents. Nested realms allow you to increase the protection level of resources that are lower in a directory tree. Below any existing realm, you can create a nested realm. You can then assign an authentication scheme with a higher protection level to the nested realm.

By default, to access resources in the child realm, a user must be authorized for resources in the parent realm *and* for resources in the child realm. You can globally change the default behavior of the Policy Server and always allow access to the resources in the child realm for users who are authorized either for the parent realm *or* the child realm. However, we do *not* recommend changing from the *and* logic to the *or* logic, which is less secure. To change to the *or* logic, remove the check from the Enable Nested Security check box.

When using nested realms, the realm structure should mimic the file structure of the resources. Each of the nested realms can have a different authentication scheme than its parent realm, with the authentication scheme for each child realm having a higher protection level than that of the parent realm. Web service clients will then need to re-authenticate when they try to access resources at lower levels of the tree. To implement this example, for each realm, you need to create a rule. Then, you need to create corresponding policies so that each policy contains a rule and users that need to access resources in a child realm can also access resources in the parent realm.

Note: Only administrators with the Manage System and Domain Objects privilege may create, edit, and delete realms. However, administrators with the Manage Domain Objects privilege may create, edit, and delete nested realms underneath existing realms in their policy domains.

More information:

Rules Overview (see page 337)
SiteMinder Administrators (see page 46)

Realms in Request Processing

When a user requests a resource, the Policy Server uses the longest matching realm to determine if a resource is protected, and if so, which authentication scheme must be used to establish the user's identity. The longest matching realm consists of the resource filter that can be located in the deepest level of any group of nested realms (or a single realm if nested realms are not used) that matches the requested path to a resource.

Configure a Realm

Realms are groupings of resources in a specific location on your network. The contents of a realm are protected by Agents. When web service clients request resources within a realm, the associated Agent handles authentication and authorization of the associated user account. The realm specifies the method of authentication.

You configure a realm to protect a group of resources that users access via a Web Server.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure the realm

1. Click Policies, Domains.
2. Click Realm, Create Realm.

   The Create Realm: Select Domain pane appears.
3. Select a domain from the Domain list, and click Next.

   The Create Realm: Define Realm pane appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Type the name and a description of the realm in the fields on the General group box.
5. Click the ellipsis button on the Resource group box.

   The Select an Agent group box opens.
6. Select a SOA Agent or Agent group, and click OK.
7. Specify the remaining resource properties on the Resource group box.
8. Create new rules or delete existing rules on the Rules group box.
9. Create new sub-realms or delete existing sub-realms on the Sub-Realms group box.
10. Specify the session properties on the Session group box.
11. Specify the authorization directory mappings and types of events the realm should process on the Advanced group box.
12. Click Finish.

    The Create Realm Task is submitted for processing.

Modify a Realm

When you modify an existing realm you cannot make changes to the following:

- Agent

  The Agent that protects a server where an existing realm is located cannot be changed. If you need to change the Agent, you must delete the realm and recreate it with the new Agent.

- Resource Filter

  The resource filter of an existing realm cannot be changed. If you need to change the resource filter, you must delete the realm and recreate it with the new resource filter.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Delete a Realm

When you delete a realm, all nested realms associated with the realm are also deleted. In addition, all rules associated with the deleted realm and its nested realms are also deleted.

It may take a short amount of time for all deleted objects to be removed from caches.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Configure a Nested Realm

Administrators who have privileges to Manage Domain Objects can create a nested realm within a parent realm, as long as the parent realm is associated with a domain within the administrator's scope.

To create a nested realm

1. Click Policies, Domains.
2. Click Realm, Modify Realm.

   The Modify Realm pane opens.
3. Specify search criteria, and click Search.

   A list of realms that match the search criteria opens.

4. Select a realm, and click Select.

   The Modify Realm: *Name* pane opens.

5. In the Sub-Realms group box, click Create Sub-Realm.

   The Create Realm pane opens.

6. Verify that Create a new object is selected, and click OK.

   The Create Realm: *Name* pane opens.

7. Type the name and a description of the realm in the fields on the General group box.

8. Type the path of the resource filter in the Resource Filter field on the Resource group box.

   Note: The resource filter of the nested realm is added to the resource filter of the parent realm. For example, if the parent realm's filter is /marketing, and the nested realm's filter is /data, the entire filter is: <agent_of_the_parent_realm>/marketing/data.

   Note: Asterisk (*) and question mark (?) characters are treated as literal characters in resource filters, not wildcards.

9. Specify session properties on the Session group box.

10. In the Advanced group box, specify the following settings:

    - Authorization directory mapping
    - Events processing

11. Click Submit.

    The Create Realm task is submitted for processing.

Flush a Single Realm from the Resource Cache

SOA Security Manager caches realm information when users access protected resources. This allows SOA Security Manager to improve network performance by keeping track of recently used resources. However, if you make a change to the security requirements or contents of a realm, you may want to flush the realm from the SOA Security Manager resource cache.

Note: If you have the Manage System and Domain Objects administrative privilege, you can flush all realms from the resource cache using the Cache Management dialog. More information exists in the *Policy Server Administration Guide*.

To flush a single realm from the resource cache

1. Login to the Administrative UI.
2. Click the Policies tab.
3. Click Domains, Realm, Modify Realm.

   The search window appears.
4. (Optional) Fill out the search form to narrow your search criteria.
5. Click Search.

   A list of realms appears.
6. Click the radio button on the left of the Realm that you want, and then click Select.

   The Modify Realm:*Realm Name* pane appears.
7. Click Flush in the Advanced group box.

   SOA Security Manager flushes the realm from the resource cache.

More information:

Start the Administrative UI (see page 42)

Chapter 13: Rules

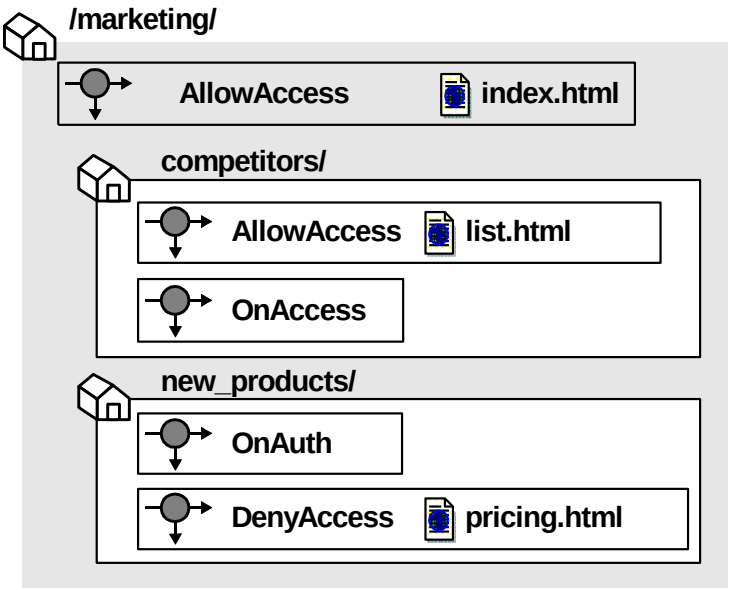
This section contains the following topics:

Rules Overview (see page 337)
Configure a Rule for Web Agent Actions (see page 341)
Resource Matching and Regular Expressions (see page 342)
Enable and Disable Rules (see page 344)
Advanced Rule Options (see page 344)
Delete a Rule (see page 346)

Rules Overview

Rules identify specific resources and either allow or deny access to the resources. Rules can also be used to trigger responses when authentication or authorization events take place. When you create rules, you must associate rules with specific realms.

The following diagram illustrates a number of realms and nested realms and their associated rules.



In the diagram above, different realms and nested realms have specific rules associated with the resources in the realm. It is also possible to have a single rule associated with all of the resources in a realm, or a subset of resources in the realm. This is done by using resource matching or regular expressions to specify resources.

More information:

Resource Matching and Regular Expressions (see page 342)

How Rules Work as Part of a Policy

Policies protect resources by binding together rules, users, and responses. Rules are the parts of policies that determine precisely which resources are protected, and which types of actions cause a rule to fire.

For example, a rule can specify all HTML files in a realm are protected for a GET action, which a Web server uses to respond to a request for an HTML page. When a user's browser attempts to access the resource, the rule fires and the policy containing the rule determines whether or not the user can view the selected resource.

How the Policy Server Processes Rules

The Policy Server evaluates rules according to the relationships between users, rules, and responses defined in policies. When a user accesses a protected resource, the Policy Server must process rules included in policies to determine whether or not the user is authorized for the resource, if any authentication and authorization events must be processed, and if any responses should be generated and returned to SOA Security Manager Agents.

When the Policy Server processes an authorization event, it looks for the realm with the longest resource filter matching the protected resource. Then, the Policy Server fires only those rules associated with that realm. In this example, the user is a manager, who wants to access the following protected resource:

/company/employees/managers/performance/

The following realms have resource filters that match the protected resource:

Realm Name	Realm Description	Resource Filter
Company	Customers, employees, vendors	/company/
Company Employees	All employees	/company/employees/

Company Managers	All managers	/company/employees/managers/
Performance Management	Managers with team members	/company/employees/managers/performance/

The realm with the longest matching resource filter is Performance Management. In response to the authorization event, the Policy Server fires all rules associated with the Performance Management realm.

In a deployment of nested realms, the Policy Server keeps a ranked list of matching realms for use during processing. If any matching rules deny access to a resource, processing stops, and the Policy Server returns any responses associated with the deny access rule to the SOA Security Manager Agent.

The Policy Server collects responses from all matching rules that fire. When the Policy Server finishes collecting responses based on rules, it deletes any duplicate responses.

In a deployment that uses nested realms, the Policy Server collects the entire list of accumulated responses for all matching rules. For OnAuthAccept rules, the Policy Server returns the entire list of responses to the SOA Security Manager Agent. For OnAuthReject rules, the Policy Server only returns the responses associated with the rule in the deepest nested realm to the SOA Security Manager Agent. OnAuthReject rules fire only for users bound to the policy.

Rules and Nested Realms

Nested realms are realms created within an existing realm. A nested realm has a parent, or top level realm, and is considered a child of the parent realm. When you create nested realms, you can also create separate rules to protect the resources in the child realms. You may also copy an existing rule, attach the rule to another realm, and rename the rule.

Rule Actions

A rule's action determines what must take place for the rule to fire. A rule fires when the Policy Server determines that an action specified in a rule occurs. The rule must be contained in an existing, enabled policy. For example, if a policy contains a rule that allows access to a web service operation, and the policy specifies users who exist in a particular directory, when one of the users listed in the directory attempts to access a resource, the Policy Server determines that the rule must fire in order to process the request.

When a rule that specifies Allow Access fires, if a user authenticates successfully, SOA Security Manager allows the user to access the specified resource. If a rule specifies Deny Access, SOA Security Manager denies access to the successfully authenticated user. Deny access rules may be added to policies to provide an additional layer of security by rejecting specific individuals or groups who should not have access to a resource. Allow Access is the default.

Deny access rules take precedence over allow access rules. If a deny access rule and an allow access rule fire when a user attempts to access a resource, the presence of the deny access rule overrides all allow access rules.

SOA Security Manager rules should specify one or more of the following *Web Agent* actions:

Post

Incoming web service request sent to the URL to which your web service is bound using the Post HTTP request action. Rules that specify the Post action will fire for any web service request posted over HTTP.

Process SOAP

Incoming web service request is a SOAP message. Rules that specify the Process SOAP action will fire for any web service request sent over HTTP or JMS and wrapped in a SOAP envelope.

Process XML

Incoming web service request is raw XML (not wrapped with a SOAP envelope). Rules that specify the Process XML action will fire for any web service request sent in raw XML format.

Note: For ProcessSOAP and ProcessXML actions to be identified, the XMLSDKResourceIdentification Agent parameter must be set true for the target SOA Agent.

Web service requests sent over HTTP must be sent to the URL to which your web service is bound using the POST HTTP request action, so the standard action that you should define for rules protecting web services accessed over HTTP is Post.

Web service request sent over JMS must be sent to the JMS queue or topic serving your web service in a SOAP message, so the standard action that you should define for rules protecting web services accessed over JMS is Process SOAP.

By default, rules created by the SOA Security Manager Administrative UI specify Post, ProcessSOAP, and ProcessXML Agent actions.

Advanced Rule Options

Advanced options allow you to define additional rule settings:

Time restrictions

Specify when a rule should and should not fire.

Active rules

Allow dynamic authorization based on external business logic.

Configure a Rule for Web Agent Actions

You create a rule that fires in response to specified actions and that allows or denies access to the resource that the rule is designed to protect.

To create a rule

1. Click Policies, Domains.
2. Click Rule, Create Rule.

   The Create Rule: Select Domain pane opens.
3. Select a domain from the Domain list, and click Next.

   The Create Rule: Select Realm pane opens.
4. Select the realm that includes the resources that you want the rule to protect, and click Next.

   The Create Rule: Define Rule pane opens.

   Note: If a realm does not exist for the resources that you want to protect, a rule cannot be created to protect those resources.
5. Type the name and a description of the rule in the fields on the General group box.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
6. Type the resource that you want the rule to protect in the Resource field.

   The Effective Resource updates to include the resource.
7. Specify whether the rules should allow or deny access to the protected resource in the Allow/Deny and Enable/Disable group box.
8. Select the Web Agent actions radio button on the Action group box.

   The Action List is populated with related actions.
9. Select one or more of the Post, Process SOAP, and Process XML actions from the Action list.

10. (Optional) Specify time restrictions, an active rule, or both on the Advanced group box.
11. Click Finish.

    The Create Rule task is submitted for processing.

Resource Matching and Regular Expressions

Rules may use resource matching and regular expression matching to specify resources in a realm.

Standard Resource Matching

By default, resource matching for a rule is done with wildcards.

The following table describes the characters that are supported for resource matching.

Character	Use
*	The wildcard (*) is used to match all characters in the string. The expression *.html will match all files with a .html file extension, such as index.html, out.html, and so forth.
?	The question mark (?) will match a single character of the string. The expression lmn?p will match the sub-string lmnop, lmnep, and so forth.

Regular Expressions for Resource Matching

Regular expressions allow for greater flexibility in resource matching. To enable regular expression matching, in the SOA Security Manager Rule dialog, select the Regular Expression check box.

Regular expressions are text patterns used for string matching. Examples of the syntax used in regular expressions are shown in the following table:

Characters	Results
\	Used to quote a meta-character (like '*')
\\	Matches a single '\' character

Characters	Results
(A)	Groups subexpressions (affects order of pattern evaluation)
[abc]	Simple character class (any character within brackets matches the target character)
[a-zA-Z]	Character class with ranges (any character range within the brackets matches the target character)
[^abc]	Negated character class
.	Matches any character other than newline
^	Matches only at the beginning of a line
$	Matches only at the end of a line
A*	Matches A 0 or more times (greedy)
A+	Matches A 1 or more times (greedy)
A?	Matches A 1 or 0 times (greedy)
A{*n*}	Matches A exactly *n* times (greedy)
A{*n*,}	Matches A at least *n* times (greedy)
A{*n*,*m*}	Matches A at least *n* but not more than *m* times (greedy)
A*?	Matches A 0 or more times (reluctant)
A+?	Matches A 1 or more times (reluctant)
A??	Matches A 0 or 1 times (reluctant)
AB	Matches A followed by B
A|B	Matches either A or B
\1	Backreference to 1st parenthesized subexpression
n	Backreference to *n*th parenthesized subexpression

Limit: Each regular expression can contain no more than 10 subexpressions, including the expression itself. The number of subexpressions equals the number of left or opening parentheses in the regular expression plus one more left parenthesis for the expression itself.

Enable and Disable Rules

You enable a rule to ensure SOA Security Manager protects the specified resources. You disable a rule to prevent SOA Security Manager from protecting the specified resources.

If a rule is enabled, no one may access the protected resource(s) unless a policy that contains the rule has been created, and the user attempting to access the rule is part of a group specified in the policy. To allow access to resources before a policy is put into place, you can disable the rule.

To enable or disable a rule

1. Open the rule.
2. Select the Enabled check box to enable the rule; clear the Enabled check box to disable the rule.
3. Click Submit.

   The rule is saved.

More information:

Start the Administrative UI (see page 42)
Legacy Administrator Privileges (see page 54)

Advanced Rule Options

The Advanced group box on the Rule pane is where you define additional rule settings. This group box lets you set time restrictions and active rules. Time restrictions and active rules are discussed in the following sections.

Add Time Restrictions to a Rule

You configure time restrictions to specify when SOA Security Manager should fire the rule.

Configuring a time restriction from 9am - 5 pm, Monday - Friday, for example, specifies that SOA Security Manager should only fire the rule during the specified time. Users have access to the resource when the rule is set to fire. The resource is not available outside of the specified time.

Note: More information about how SOA Security Manager handles time across multiple time zones exists in How a SOA Agent and Policy Server Calculate Time (see page 57).

To configure a time restriction

1. Click Set in the Time Restrictions group box.

   The Time Restrictions pane appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

2. Specify starting and expiration dates.

3. Specify time restrictions in the Hourly Restrictions table.

   Note: Each check box represents one hour. When a check box is selected, the rule fires during that hour, and the rule applies to the specified resources. When a a check box is cleared, the rule does not fire during that hour, and the rule will not apply to the specified resources.

4. Click OK.

   The time restrictions are saved, and the rule settings appear.

Configure an Active Rule

You configure an active rule for dynamic authorization based on external business logic. The Policy Server invokes a function in a customer-supplied shared library. This shared library must conform to the interface specified by the Authorization API, which is available separately via the Software Development Kit

Note: More information on shared libraries exists in the *API Reference Guide for C*.

To configure an Active Rule

1. Select the Edit Active Rule check box in the Active Rule group box.

   Active rule fields appear.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

2. Specify the library name, function name, and function parameters in the respective fields.

   The active rule string appears in the Active Rule field.

3. Click Submit.

   The active rule is saved.

Delete a Rule

If you delete a rule, the rule is automatically removed from the policies that included the rule. However, the policies remain on your system. Verify that the policies function without the deleted rule.

Note: Policies must contain at least one rule.

When you delete a rule that is included in a rule group, it may take several seconds before the deleted rule is removed from the rule group. It may also take a short amount of time for all deleted objects to be removed from caches.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Chapter 14: Rule Groups

This section contains the following topics:

Rule Group Overview (see page 347)
Create a Rule Group (see page 347)
Add Rules to a Rule Group (see page 348)
Modify a Rule Group (see page 349)
Modify a Rule Group (see page 349)
Delete a Rule Group (see page 350)

Rule Group Overview

A rule group is a set of rules that can be bound to SOA Security Manager policies. You can use a rule group to combine groups of rules you will be applying to the same policy. For example, if you have a number of rules that allow Post and Process SOAP action for different related web service resources, you could then create a rule group that contains all of the resources. When you configure the policy that will include the rules, you can add a single rule group to the policy, rather than add all of the rules individually.

When you include a rule group in a policy, each rule in the group is evaluated and applied independently of other rules in the group.

Create a Rule Group

You can create a rule group and add it to a domain.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To create a rule group

1. Click Policies, Domains.
2. Click Rule Group, Create Rule Group.

   The Create Rule Group pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create Rule Group: Select Domain pane opens.
4. Select a domain name from the drop-down list, and click Next.

   The Create Rule Group: Define Rule Group pane opens.

5. Type the name and a description of the rule group in the fields on the General group box.
6. Select SiteMinder and an Agent Type on the Attributes group box.
7. Click Add/Remove on the Group Members group box.

   The Choose rules pane opens.

   The Available Members column lists all rules that are defined in the specified domain and in the realms associated with the specified Agent type.
8. Select one or more rules from the list of Available Members and click the right-facing arrows.

   The rules are removed from the list of Available Members and added to the list of Selected Members.

   To select more than one member at one time, hold down the Ctrl key while you click on the additional members. To select a block of members, click on the first member and then hold down the Shift key while you click on the last member in the block.
9. Click OK.

   The selected rules are added to the rule group.
10. Click Finish.

    The Create Rule Group Task is submitted for processing.

Add Rules to a Rule Group

.You can add rules to a rule group in the same domain and of the same Agent type.

To add rules to a rule group

1. Click Policies, Domains.
2. Click Rule Group, Modify Rule Group.

   The Modify Rule Group pane opens.
3. Specify search criteria, and click Search.

   A list of rule groups opens.
4. Select a rule group, and click Select.

   The Modify Rule Group: *Name* pane opens.
5. Click Add/Remove on the Group Members group box.

   The Choose rules pane opens.

Note: The Available Members column lists all rules that are defined in the specified domain and in the realms associated with the specified Agent type.

6. Select one or more rules from the list of Available Members and click the right-facing arrows.

   The rules are removed from the list of Available Members and added to the list of Selected Members.

   Note: To select more than one member at one time, hold down the Ctrl key while you click on the additional members. To select a block of members, click on the first member and then hold down the Shift key while you click on the last member in the block.

7. Click OK.

   The selected rules are added to the rule group.

8. Click Submit.

   The Modify Rule Group Task is submitted for processing.

Modify a Rule Group

You can modify all of the properties of a rule group, except the Agent Type for SOA Security Manager Agents and the vendor type for RADIUS Agents. To change the Agent type or vendor type, delete the rule group and create a new one.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Modify a Rule Group

You can modify all of the properties of a rule group except the Agent Type. To change the Agent type or vendor type, delete the rule group and create a new one.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Delete a Rule Group

Deleting a rule group only deletes the grouping. The rules contained in the grouping are not deleted.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Chapter 15: Responses and Response Groups

This section contains the following topics:

Responses Overview (see page 351)
Response Attribute Types (see page 352)
Web Agent Response Attributes for SOA Agents (see page 353)
Responses and Directory Mappings (see page 356)
Configure a Response (see page 356)
Configure a Response Attribute for SOA Agents (see page 357)
Configure Responses to Generate SAML Session Tickets for Outgoing Messages (see page 357)
Configure Responses for WS-Security Header Production (see page 364)
Configure Response Attribute Caching (see page 377)
Variable Objects in Responses (see page 377)
Edit a Response (see page 380)
Delete a Response (see page 380)
Response Groups (see page 380)

Responses Overview

A response is a container for one or more response attributes which the Policy Server sends to a SOA Agent after processing the response. The Policy Server supports several types of response corresponding to different Agent types, each of which provides a different set of attributes, which take the form of name/value pairs.

SOA Agents accept only accept *Web Agent* responses, which provide attributes (name/value pairs) that can be used by SOA Agents and SiteMinder Web Agents.

Policies contains rules and responses which are bound to users and user groups. In a policy, responses are bound to specific rules or rule groups. When a rule fires, the associated response returns information to a SOA Agent, such as user attributes, DN attributes, static text, or customized active responses. Responses can also be used to instruct a SOA Agent to generate WS-Security headers and SAML Session Tickets.

Response Attribute Types

SOA Security Manager supports different types of response attributes. The types of response attributes determine where the Policy Server finds the proper values for the response attributes.

You can specify the following types of response attributes when you add response attributes to a SOA Security Manager response:

Static

Returns data that remains constant.

Use a static attribute to return a string as part of a SOA Security Manager response. This type of response can be used to provide information to a Web application. For example, if a group of users has specific customized content on a Web site, the static response attribute, show_button = yes could be passed to the application.

User Attribute

Returns profile information from a user's entry in a user directory.

This type of response attribute returns information associated with a user in a directory. A user attribute can be retrieved from an LDAP, WinNT, Microsoft SQL Server or Oracle user directory.

Note: In order for the Policy Server to return values from user directory attributes as response attributes, the user directories must be configured on the SOA Security Manager User Directory pane.

DN Attribute

Returns profile information from a directory object in an LDAP, Microsoft SQL Server or Oracle user directory.

This type of response attribute is used to return information associated with directory objects to which the user is related. Groups to which a user belongs, and Organizational Units (OUs) that are part of a user DN, are examples of directory objects whose attributes can be treated as DN attributes.

For example, you can use a DN attribute to return a company division for a user, based on the user's membership in a division.

Note: In order for the Policy Server to return values from DN attributes as response attributes, the user directories must be configured on the SOA Security Manager User Directory pane.

Active Response

Returns values from a customer supplied library that is based on the SOA Security Manager Authorization API.

An Active Response is used to return information from an external source. An Active Response is generated by having the Policy Server invoke a function in a customer-supplied shared library. This shared library must conform to the interface specified by the Authorization API (available separately with the Software Development Kit; if installed, see the API Reference *Guide for C* for more information).

Note: It is up to you to make sure the value returned by an active response is valid. For example, if an active response returns a numeric type, the library and function must return a string whose value is a number. When you configure a response attribute, the correct Value Type for the response attribute is displayed on the Response Attribute pane.

Variable Definition

Returns the value of the specified variable at runtime.

Select Variable Definition when you want to select and use a variable from a list of already-defined variables.

Web Agent Response Attributes for SOA Agents

Web Agent response attributes are response attributes that SOA Agents and SOA Security Manager Web Agents can interpret and pass on to other applications. The following is a list of Web Agent response attributes available in SOA Security Manager:

WebAgent-SAML-Session-Ticket-Variable

Provides data from the Policy Server that the SOA Agent uses to generate a SAML assertion to insert into an XML message's HTTP or SOAP envelope header or a cookie (as specified by associated response attributes).

When you configure a SAML Session Ticket response, the Policy Server generates the response data that instructs the SOA Agent how to build the assertion. The SOA Agent encrypts a session ticket (and optionally, a web service consumer's public key) together with the response data and generates the actual assertion. The Agent then delivers the assertion to the web service. The token can only be encrypted and decrypted by the SOA Agent using its Agent key.

WebAgent-WS-Security-Token

Provides data from the Policy Server that the SOA Agent uses to generate WS-Security Username, X509v3, or SAML tokens (as specified by associated response attributes) to add to a SOAP message header.

When you configure a WS-Security response, the Policy Server generates the response data that instructs the SOA Agent how to build the token. The Agent then generates and adds the token to the SOAP request and delivers it to the web service.

WebAgent-HTTP-Authorization-Variable

Indicates an attribute defined and reserved for future SOA Security Manager use.

WebAgent-HTTP-Cookie-Variable

Generates a SetCookie header, which then sets a non-persistent cookie in a Web browser. The cookies only exist in the cookie domain where the Web Agent is configured. You can enter multiple WebAgent-HTTP-Cookie-Variables.

Limits: Use in accept or reject responses. Multiple instances of this attribute are allowed per response.

WebAgent-HTTP-Header Variable

Specifies an arbitrary dynamic name/value pair for use by a Web application. You can enter multiple WebAgent-HTTP-Header-Variables.

The Web Agent does not include header variables in the responses that it sends back to a Web browser. Instead, these responses, generated by the Policy Server, reside in the request headers of the Web server.

Consequently, the header variables will not be visible in the debug logs that you can enable from the Policy Server Management Console.

Limits: Use in accept or reject responses. Multiple instances of this attribute are allowed per response.

WebAgent-OnAccept-Redirect

Defines *one* of the following, depending on the type of response in which it is used:

- In an authorization response, this defines a URL to redirect the user to if the user is allowed access to a resource.
- In an authentication response, this defines a URL to redirect the user to if the user was authenticated for a security realm.

To determine whether or not this is an authorization or authentication response, include it in a policy with a rule that specifies an OnAuthAccept or OnAccessAccept event action.

Limits: Use in accept responses. Only one instance of this attribute is allowed per response.

WebAgent-OnAccept-Text

Specifies text that the Web Agent puts in the HTTP_ONACCEPT_TEXT environment variable when it redirects the user after a successful authorization or authentication attempt.

Limits: Use in accept responses. Only one instance of this attribute is allowed per response.

WebAgent-OnAuthAccept-Session-Idle-Timeout

Overrides the number of seconds a user session can be idle. Once this limit is reached, the user is forced to re-authenticate. Associate this response with a rule configured with an OnAuthAccept authentication event.

Limits: Use in accept responses. Only one instance of this attribute is allowed per response.

WebAgent-OnAuthAccept-Session-Max-Timeout

Overrides the total number of seconds a user session can be active. Once this limit is reached, the user session is terminated and the user is forced to re-authenticate. Associate this response with a rule configured with an OnAuthAccept authentication event.

Limits: Use in accept responses. Only one instance of this attribute is allowed per response.

WebAgent-OnAuthAccept-Session-AuthContext

Specifies an AuthContext response attribute for an authentication scheme. The value of this response attribute is added to the SOA Security Manager session ticket as the value of the SM_AUTHENTICATIONCONTEXT user attribute. It is not returned to the client as a user response.

Note: The response attribute value is truncated to 80 bytes in length.

Limits: Used in accept responses. Only one instance of this attribute is allowed per response.

WebAgent-OnReject-Redirect

Defines *one* of the following, depending on the type of response in which it is used:

- In an authorization response, this defines a URL to redirect the user to if the user is denied access to a resource.
- In an authentication response, this defines a URL to redirect the user to if the user has failed to authenticate for a security realm.

To determine whether or not this is an authorization or authentication response, include it in a policy with a rule that specifies an OnAuthReject or OnAccessReject event action.

Limits: Use in reject responses. Only one instance of this attribute is allowed per response.

WebAgent-OnReject-Text

Specifies text that the Web Agent puts in the HTTP_ONREJECT_TEXT environment variable when it redirects the user after a failed authorization or authentication attempt.

Limits: Use in reject responses. Only one instance of this attribute is allowed per response.

Responses and Directory Mappings

Directory mappings let you specify a separate authorization user directory for a realm. When you define a separate authorization directory, a user is authenticated based on the information contained in one directory, but authorized based on the information contained in another directory.

When you create a response and associate it with a authentication (OnAuth) event, any information retrieved from a user directory is retrieved from the authentication directory. If you create an authorization (OnAccess) event, any information retrieved from a user directory is retrieved from the authorization directory.

More information:

Directory Mapping Overview (see page 207)

Configure a Response

You create a response for SOA Security Manager by specifying the Web Agent agent type and an attribute list. A response contains the specified attributes and is sent to the specified agent.

To create a response

1. Click Policies, Domains.
2. Click Response, Create Response.

   The Create Response: Select Domain pane opens.
3. Select a domain, and click Next.

   The Create Response: Define Response pane opens.
4. Type the name and a description of the response in the fields on the General box.
5. Select SiteMinder and the Web Agent Agent Type on the Attributes group box.

6. Click Create Response Attribute to create a response attribute and add it to the attribute list.

   Configure the response attribute (see page 357) on the Create Response Attribute pane which opens.

7. Create further response attributes as required.
8. Click Finish.

   The Create Response Task is submitted for processing.

Configure a Response Attribute for SOA Agents

You create a response attribute for a SOA Agent by selecting SiteMinder and Web Agent on the Attributes group box on the Response pane.

To create a response attribute

1. Click Create Response Attribute on the Attribute List group box on the Response pane.

   The Create Response Attribute pane opens.

2. Select a response attribute from the Attribute drop-down list.
3. Select an attribute type on the Attribute Kind (one of Static, User Attribute, DN Attribute, and Active Response) group box.

   The fields on the Attribute Fields group box are updated to match the specified attribute type.

4. Complete the fields on the Attribute Fields group box.

   Note: For WebAgent-SAML-Session-Ticket-Variable and WebAgent-WS-Security-Token attributes, you can either enter values directly in the Variable Name and Variable Value fields or populate those fields with valid values from the Select a Name and Select a Value lists that appear.

5. Specify Cache Value or Recalculate value every ... seconds on the Attribute Caching group box.
6. Click Submit.

   The Create Response Attribute Task is submitted for processing, and the response attribute is added to the Attribute List on the Response pane.

Configure Responses to Generate SAML Session Tickets for Outgoing Messages

This section describes how to generate SAML Session Tickets.

More information:

Authentication Service Models (see page 27)
SAML Session Ticket Authentication (see page 233)

How the SAML Session Ticket Response is Used

The SAML Session Ticket response provides the data that the SOA Agent uses to create an assertion. The only authentication scheme that can evaluate the assertion is the SAML Session Ticket authentication scheme.

When an XML assertion document arrives at a web service protected by the SAML Session Ticket authentication scheme, the SOA Agent does the following:

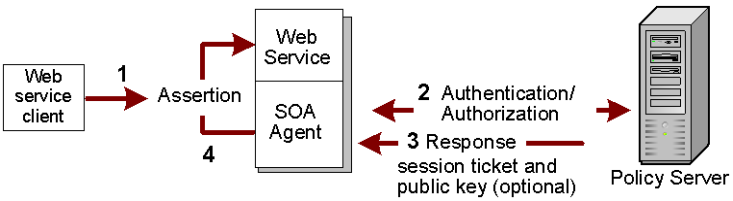
- Extracts the encrypted session ticket from the assertion
- Decrypts the session ticket using the agent key
- Signs the document using the public key from the assertion

  The binding of the session ticket and the public key ensures that the XML document is signed by a client that is authenticated by SOA Security Manager and that has a valid session.

Then, if the Agent does not have the session ticket in its cache, the Policy Server validates the client with the session ticket from the assertion. If the Agent does have the session ticket in its cache, the Policy Server is not invoked.

Note: The web service that returns the assertion is not protected by the SAML Session Ticket authentication scheme. Only subsequent services in the single sign-on environment require this authentication scheme.

The following illustration shows the response process.



1. Client sends a request.
2. SOA Agent passes credentials to Policy Server. Authentication handled by any CA SiteMinder-supported authentication scheme.
3. After the client is authenticated, the client is authorized. The policy that authorizes the client has a SAML response configured with it, which generates a session ticket and, optionally, a public key.
4. SOA Agent generates the assertion and delivers it to the web service.

Configure a SAML Session Ticket Response

SAML Session Ticket responses are actions that tell SOA Agents to send SAML Session Tickets to web service implementations. These responses are bound to rules and triggered when those rules fire.

Use variable types, if needed, to pass data back to the web service. Variables are resolved by the Policy Server at run time, when it generates the response.

Before you configure a SAML Session Ticket response, do the following:

- If the public keys used in assertions are going to be stored in the user directory, define an attribute in your directory to store these public keys, and make sure it is available to the Policy Server.

  Note: This is not required if the public key is included in the client's submitted XML document or obtained from a certificate over the SSL link.

- In an authentication service environment, create an application to protect the authentication service. This policy includes the following:
  - A realm that protects the authentication service URL
  - A rule that fires when a user posts an XML document to the authentication service realm

SAML Session Ticket Response Attribute Variables

The following table lists the response attribute variable name/value pairs specific to the WebAgent-SAML-Session-Ticket-Variable attribute. You can use these variables to build assertions.

Note: You can configure other response variables with the SAML Session Ticket attribute; however, they are ignored by SOA Security Manager for the assertion and are handled as standard response attributes by CA SiteMinder.

Variable Name	Variable Value	Attribute Kind	Meaning
TXM_SAML_Location (required)	■ Envelope_Header (default) ■ HTTP_Header ■ Cookie_Header	Static	Instructs the SOA Agent to insert the assertion into the SOAP envelope message header, an HTTP header, or a cookie header. If Envelope_Header is the value, the client must provide an XML message for the assertion. If HTTP_Header is the value, an HTTP header named tmsamlsessionticket is added to the HTTP headers delivered to the web service If Cookie_Header is the value, the assertion is inserted into a cookie named tmsamlsession and returned to the caller in an HTTP Set-Cookie header. The cookie can also be read by the web service application at the URI protected by SOA Security Manager. **Note:** Do not attempt to place more than one signature in a cookie—a 4 KB limit on the size of cookies that can be returned by the SOA Agent results in a dummy cookie being returned to the caller if the generated cookie is greater than 4KB.

Variable Name	Variable Value	Attribute Kind	Meaning
TXM_Force_Logon (optional)	Yes or No	Static	Forces the client to authenticate using the authentication scheme for the target realm. This variable is useful if a client tries to get an assertion when logging on with only a cookie. The client is allowed access to the web service, but does not receive an assertion because the client has only a cookie. To inform the user that they have to logoff and then get rechallenged to obtain the assertion, the web service can be set up to redirect the client to a log-off URI. The user can then come back to the web service and be challenged again to obtain the assertion. **Note:** To find out how to set up a log-off URI, see the *CA SiteMinder Agent Guide*.
TXM_Issuer (optional)	URI	Static	Indicates the issuer of the assertion. Value is placed in the issuer URI field in the generated assertion. If the assertion is sent to a third party, the third party can use this variable to validate the assertion by sending it back to the specified URI.
TXM_Namequalifier (optional)	Domain name	Static	Indicates the domain name of the subject of the assertion.
TXM_Sign (optional)	Yes or No	Static	Tells the SOA Agent to sign the SOAP document payload with the private key dynamically generated by the Policy Server. **NOTE**: If you use this variable, do not use the TXM_Public_Key variable.
TXM_Sign_Assertion (optional)	Yes or No	Static	Tells the SOA Agent to sign the assertion that is part of the SOAP document. This ensures that no one can alter the assertion.

Variable Name	Variable Value	Attribute Kind	Meaning
TXM_Public_Key (optional)	■ XMLDSIG ■ Client_Cert ■ User_Store	Static	Tells the SOA Agent where to get the public key that it binds to the session ticket. **XMLDSIG**—Tells SOA Agent to get the key from the document with the digital certificate. (Web service must be protected by the XML Digital Signature authentication scheme.) **Client_Cert**—Indicates the client certificate sent over the SSL connection **User_Store**—Tells SOA Agent to get the key from the user store. **Note:** Do not use this variable with TXM_Sign.
TXM_User_Cert LDAP user directories only (optional)	usercertificate This value is the most common for LDAP user directories. If you have used a custom naming scheme for your LDAP directory, the value will be different.	User Attribute	Specifies the LDAP query string that the SOA Agent uses to retrieve the public key from the user store. This variable is required when TXM_Public_Key is set to User_Store.

Note: Do not use the SAML assertion, XML Body, XML Agent, and XML Envelope Header variables that you can choose from the Variables policy object in a policy domain. These variables are for use exclusively in policy expressions, not with the SAML Session Ticket response.

Enter these variables by typing the name and value in the appropriate fields in the Response Attribute dialog.

Use SAML Session Ticket Assertion Variables for a Session Ticket Response

You can use assertion variables to help the SOA Agent build the assertion.

Example 1

If the web service is protected by the XML-DSIG authentication scheme, create an attribute that extracts the client's public key from the certificate and adds it to the SAML assertion. To instruct the SOA Agent to get the public key from the digital certificate, enter the variable TXM_Public_Key with the value XMLDSIG.

The following table shows how to complete the Response Attribute fields:

Field	Value
Attribute	WebAgent-SAML-Session-Ticket-Variable
Attribute Kind	Static
Variable Name	TXM_Public_Key
Variable Value	XMLDSIG

If the public key is coming from the user directory, two response attributes are required. The fields for the first attribute in the Response Attribute dialog would be as follows:

Field	Value
Attribute	WebAgent-SAML-Session-Ticket-Variable
Attribute Kind	User Attribute
Variable Name	TXM_User_Cert
Variable Value	usercertificate

The fields for the second attribute in the Response Attribute dialog would be as follows:

Field	Value
Attribute	WebAgent-SAML-Session-Ticket-Variable
Attribute Kind	Static
Variable Name	TXM_Public_Key
Variable Value	User_Store

Example 2

To ensure that the assertion is placed in the SOAP envelope message header, the fields in the Response Attribute dialog would be as follows:

Field	Value
Attribute	WebAgent-SAML-Session-Ticket-Variable
Attribute Kind	Static

Field	Value
Variable Name	TXM_SAML_Location
Variable Value	Envelope_Header

Configure Responses for WS-Security Header Production

This section describes how to configure responses that produce WS-Security headers.

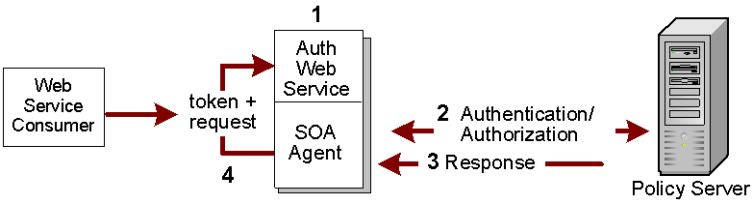
More information:

Authentication Service Models (see page 27)
WS-Security Authentication (see page 243)

How the WS-Security Response is Used

WS-Security responses are typically used to instruct the SOA Agent protecting an authentication web service to create WS-Security headers and, optionally, to perform XML encryption on those headers and the message content.

The following illustration shows the response process in such an environment.



1. A web service consumer sends a request (in the form of an XML message) to the authentication web service.
2. The SOA Agent obtains credentials and passes them to the Policy Server. Authentication is handled by any *supported* authentication scheme.

   Note: Although any authentication scheme can be configured to obtain credentials from a request, not every authentication scheme is suitable for creating every type of WS-Security token.
3. After the web service consumer is authenticated, the client is authorized. The policy that authorizes the consumer has a WS-Security response configured with it, which instructs the SOA Agent to generate WS-Security headers.
4. The SOA Agent generates the WS-Security headers and delivers them, together with the request message, to the authentication web service.

However, for a web service that receives requests with XML-encrypted elements, but that does not have the logic to decrypt those requests internally, WS-Security responses can be used to instruct the SOA Agent to pass the web service decrypted versions of those requests (see TXM_WSSEC_ENCRYPT_PUB_KEY_ROLE).

More information:

Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

Configure a WS-Security Response

WS-Security responses are actions that tell SOA Agents to send WS-Security tokens to web service implementations. These responses are associated with and triggered when those rules fire.

Use variable types, if needed, to pass data back to the web service. Variables are resolved by the Policy Server at run time, when it generates the response.

Before you configure a WS-Security response, do the following:

- If you are signing the WS-Security tokens you generate, store your enterprise private key and certificate chain in the SMKeyDatabase.
- If you are using WS-Security SAML tokens, configure an affiliate object (within an affiliate domain) that will create the SAML assertions to be inserted in the tokens.
- In an authentication service environment, create a policy to protect the authentication service and trigger response generation. This policy should include the following:
  - A realm that protects the authentication service URL.
  - A rule that fires when a user posts an XML document to the authentication service realm

Use response attributes specific to the WebAgent-WS-Security-Token attribute to build WS-Security headers and tokens.

Note: You can configure other response attributes with the WS-Security attribute; however, they are ignored by SOA Security Manager for the purpose of generating the WS-Security token and are handled as standard SiteMinder response attributes.

More information:

Configure a Response (see page 356)
Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

Variable for Specifying the Generated WS-Security Token Type

The following table describes the mandatory TXM_WSSEC_TOKEN_TYPE response variable name/value pair that determines the WS-Security token type to generate.

Variable Name	Variable Value	Attribute Kind	Meaning
TXM_WSSEC_TOKEN_TYPE	■ password ■ password_nodigest ■ X509 ■ SAML	Static	Specifies the type of WS-Security token the SOA Agent should create and add to the WS-Security header for message authentication: ■ password—Creates a Username and Password Digest token, providing for password digest message authentication. For this token type, the TXM_WSSEC_USER_PASSWORD response variable must also be configured. ■ password_nodigest—Creates a Username and Password token, providing clear text password message authentication. For this token type, the TXM_WSSEC_USER_PASSWORD response variable must also be configured. ■ X509—Creates an X509v3 token. ■ SAML—Creates a SAML token, providing for SAML message authentication. For this token type, the TXM_WSSEC_SAML_AFFILIATE response variable must also be configured.

More information:

Choose a WS-Security Token Type (see page 248)
Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

Variables for Generating Username and Password and X.509 Certificate Tokens

The following table describes the response variable name/value pairs associated with generating username and password (digest or clear text) and X.509 certificate tokens.

Variable Name	Variable Value	Attribute Type	Meaning
TXM_WSSEC_USER_PASSWORD	userpassword (Value most common for LDAP user directories -- if you have used a custom naming scheme for your LDAP directory, the value will be different.)	User Attribute	Specifies the LDAP query string that the SOA Agent uses to retrieve the web service consumer's password from the user store. This value is then placed in the token.
	Or *password*	Static	Specifies a static password value to be used in the token.
TXM_WSSEC_ROLE (Optional)	*token_role_name*	Static	Specifies the value of a SOAP role attribute that identifies the WS-Security header element containing the Username and Password or X.509 token.
TXM_WSSEC_TIMESTAMP (optional)	■ True ■ False	Static	If True, tells the agent to add a wsu:Timestamp element to the WS-Security SOAP header that specifies the time that the message was created
TXM_WSSEC_TIMESTAMP _EXPIRY (Only set if TXM_WSSEC_TIMESTAMP is True)	*message lifespan in seconds*	Static	Tells the agent to add a wsu:Expires element to the wsu:Timestamp element in the WS-Security SOAP header. The value of the wsu:Expires element is an absolute time based on the time of message creation and the specified message lifespan.

Variable Name	Variable Value	Attribute Type	Meaning
TXM_WSSEC_SIGNATURE (optional)	■ all ■ body_ts ■ body ■ headers	Static	For WS-Security tokens of type password or X509, tells the agent to retrieve the enterprise private key from the Smkeydatabase and use it to digitally sign the SOAP document: ■ all—the generated signature will cover the entire SOAP envelope. ■ body_ts—the generated signature will cover the SOAP body and the generated <wsu:Timestamp> element. If a timestamp response attribute is not configured, a message will be logged and the signature will cover only the SOAP body. ■ body—the generated signature will cover the SOAP body. ■ headers—the generated signature will cover the SOAP header containing the generated/modified WS-Security element.

More information:

Username and Password Digest Token (see page 249)
X509v3 Certificate Token (see page 250)
Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

Variables for Generating SAML Tokens

The following table describes response variable name/value pairs associated with generating SAML tokens for use in WS-Security tokens.

Variable Name	Variable Value	Attribute Type	Meaning
TXM_WSSEC_SAML20_ASSERTION	■ Yes ■ No (default)	Static	Specifies whether the generated SAML assertion token is SAML 2.0 compliant.

Variable Name	Variable Value	Attribute Type	Meaning
TXM_WSSEC_SAML20_SPID (required for SAML 2.0)	*SAML_20_audience_value*	Static	Specifies the value of the <saml:Audience> element in a generated SAML 2.0 assertion token.
TXM_WSSEC_SAML_AFFILIATE	*affiliate_or_service_provider_object_name*	Static	Identifies the affiliate (SAML 1.x) or service provider (SAML 2.0) object that configures how SAML assertions will be produced for inclusion in SAML tokens.
TXM_WSSEC_SAML_ROLE (optional)	*SAML_assertion_token_role_name*	Static	Specifies the value of a SOAP role attribute that identifies the WS-Security header element containing the SAML assertion token.
TXM_WSSEC_SAML_SIG_REQUIRED	■ HK ■ SV ■ SVS	Static	Specifies how the assertion and document should be signed: ■ HK (for holder-of-key)—Only the assertion will be signed (enveloped). ■ SV (for sender-vouches with SSL-based issuer confirmation)—Both assertion and document will be signed (external). ■ SVS (for sender-vouches with signature-based issuer validation)—Assertion is explicitly signed (enveloped) in addition to SV signing. (This option is only supported for SAML 1.x assertions.) Any other value or no value results in the default—no signing.

Variable Name	Variable Value	Attribute Type	Meaning
TXM_WSSEC_ SAML_USER_CERT_SRC	■ XMLDSIG ■ Client_Cert ■ User_Store	Static	If TXM_WSSEC_SAML_SIG _REQUIRED is set to HK, this value specifies where SOA Security Manager should obtain the web service consumer's public key: ■ XMLDSIG—The public key will be retrieved from a signed request sent to a web service protected by the XML DSIG authentication scheme. ■ Client_Cert—The public key will be retrieved from SSL. ■ User_Store—the public key should be retrieved from an associated user store. If this value is set, the TXM_WSSEC_SAML_USER_CERT response variable must also be configured. **Note:** If TXM_WSSEC_SAML_SIG _REQUIRED is set to SV, this option is ignored because no user public key is required.
TXM_WSSEC_ SAML_USER_CERT	usercertificate This value is the most common for LDAP user directories. If you have used a custom naming scheme for your LDAP directory, the value will be different.	User Attribute	If TXM_WSSEC_SAML_USER_ CERT_SRC is set to User_Store, specifies the LDAP query string that the SOA Agent uses to retrieve the web service consumer's public key from the user store for signing SAML assertion tokens. **Note:** SOA Security Manager automatically completes the query string using the value you specify.
TXM_WSSEC_ SAML_TIMESTAMP (optional)	■ True ■ False (default)	Static	A value of True causes a timestamp to be generated for use in SAML assertions. **Note:** If TXM_WSSEC_SAML_SIG _REQUIRED is set to SV or SVS, the timestamp is signed.

Variable Name	Variable Value	Attribute Type	Meaning
TXM_WSSEC_ SAML_TIMESTAMP _EXPIRY (optional)	*message _lifespan_in_seco nds*	Static	Tells the agent to add an expiry element to the timestamp used in SAML assertions. The value of this expiry element is an absolute time based on the time of assertion creation and the specified message lifespan.

More information:

SAML Assertion Token (see page 251)
Configuration Requirements for Generating SAML Assertions (see page 296)
Supported Authentication Schemes for Producing Each WS-Security Header Type (see page 244)

Variables for Encrypting/Decrypting WS-Security Messages

The following table describes response variable name/value pairs that can be configured to tell the SOA Agent to encrypt message elements or to pass a decrypted version of a message to the recipient web service.

Note: There are two versions of each XML encryption-related name/value pair—use the former for use with messages with username/password or X.509 tokens, use the latter for use with messages with SAML tokens.

Variable Name	Variable Value	Attribute Type	Meaning
TXM_WSSEC_ENCRYPT_PUB_KEY_ROLE or TXM_WSSEC_SAML_ENCRYPT_PUB_KEY_ROLE (required)	*name_of_WS-Security_token_consumer*	Static	Specifies the value of a SOAP role attribute that identifies the WS-Security header element containing the recipient's X.509 certificate. The public key in this certificate is used to encrypt the symmetric key. The corresponding private key must be held by the intended message recipient. This element is *required*. If no role is specified, the variable must be declared with a null value; SOA Security Manager will then obtain the key in the WS-Security header with no role, of which only one is allowed.

Variable Name	Variable Value	Attribute Type	Meaning
TXM_WSSEC_ENCRYPT_DECRYPT or TXM_WSSEC_SAML_ENCRYPT_DECRYPT	■ True ■ False (default)	Static	Specifies whether the SOA Agent should pass an incoming encrypted message to the web service in its encrypted or decrypted form. If True, the SOA Agent will replace the current message with the decrypted version of the message, if available.
TXM_WSSEC_ENCRYPT_ELEMENT or TXM_WSSEC_SAML_ENCRYPT_ELEMENT	■ UsernameToken ■ Assertion ■ Body	Static	Identifies the message element to be encrypted. You should add one such name value/pair for *each* element you want encrypted. For example, configure one name/value pair for the message body and one name/value pair for the token. For TXM_WSSEC_ENCRYPT_ELEMENT: If **UsernameToken**, Username and Password and Username and Password Digest tokens will be encrypted. If **Body**, the message body will be encrypted. For TXM_WSSEC_SAML_ENCRYPT_ELEMENT: If **Assertion**, SAML assertion token will be encrypted. If **Body**, the message body will be encrypted.
TXM_WSSEC_ENCRYPT_OR_SIGN_FIRST or TXM_WSSEC_SAML_ENCRYPT_OR_SIGN_FIRST	■ Sign (default) ■ Encrypt	Static	Indicates whether encryption or signing should be performed first.
TXM_WSSEC_ENCRYPT_ALG_KEY or TXM_WSSEC_SAML_ENCRYPT_ALG_KEY	■ rsa-1_5 (default) ■ rsa_oaep	Static	Indicates the encryption algorithm to use to encrypt the symmetric encryption key.

Variable Name	Variable Value	Attribute Type	Meaning
TXM_WSSEC_ENCRYPT_ALG_DATA or TXM_WSSEC_SAML_ENCRYPT_ALG_DATA	■ tripledes-cbc (default) ■ aes128-cbc ■ aes256-cbc ■ aes192-cbc	Static	Indicates the encryption algorithm to use to encrypt the data element or elements that have been specified using TXM_WSSEC_ENCRYPT[_SAML]_ELEMENT variables.

Variables for Handling WS-Security Headers

The following table describes response variable name/value pairs that can be configured to tell the SOA Agent how to handle the mustUnderstand attribute in WS-Security request , or to remove the mustUnderstand or the entire security token from messages passed to the recipient web service.

Variable Name	Variable Value	Attribute Type	Meaning
TXM_WSSEC_MUST_UNDERSTAND (optional)	■ True ■ False ■ Not set (default)	Static	Determines how the SOA Agent behaves with respect to the mustUnderstand attribute when consuming and producing messages. For specifics of the behavior of this variable when consuming and producing headers, see TXM_WSSEC_MUST_UNDERSTAND Response Variable Effect Detail (see page 374).
TXM_WSSEC_REMOVE (optional)	■ True ■ False (default)	Static	Determines whether the SOA Agent removes WS-Security headers from messages. ■ If True, the WS-Security header for a specified actor/role (or the default header if no actor/role is defined) is removed from the inbound document after authentication but before any response is applied. This setting effectively scrubs credentials. ■ (Default) If False, no change to is made to the request document.

More information:

TXM_WSSEC_MUST_UNDERSTAND Response Variable Effect Detail (see page 374)

TXM_WSSEC_MUST_UNDERSTAND Response Variable Effect Detail

The following table describes the exact behavior of the SOA Agent when consuming and producing WS-Security tokens with different values of the TXM_WSSEC_MUST_UNDERSTAND response variable name/value pair.

TXM_WSSEC_MUST_UNDERSTAND Value	Behavior when Consuming WS-Security Tokens	Behavior when Producing WS-Security Tokens
Not set	If present in the WS-Security header, leaves mustUnderstand="1".	Places mustUnderstand="1" in the generated WS-Security header
False	If present in the WS-Security header, removes mustUnderstand="1"	Does not place mustUnderstand="1" in the generated WS-Security header
True	If present in the WS-Security header, leaves mustUnderstand="1".	Places mustUnderstand="1" in the generated WS-Security header

WS-Security Response Examples

The following examples show how you can use WS-Security responses.

Example 1

This example shows how to create a response that generates a Username and Password Digest token and uses the enterprise private key to digitally sign the message's SOAP envelope.

The following table shows the response attributes you must add to the response (all attributes are of type WebAgent-WS-Security-Token):

Variable Name	Variable Value	Attribute Type
TXM_WSSEC_TOKEN_TYPE	password	Static
TXM_WSSEC_USER_PASSWORD	userpassword	User Attribute
TXM_WSSEC_SIGNATURE	all	Static

Example 2

This example shows how to create a response that generates an X509v3 token and uses the enterprise private key to digitally sign the message's SOAP envelope.

The following table shows the response attributes you must add to the response (all attributes are of type WebAgent-WS-Security-Token):

Variable Name	Variable Value	Attribute Type
TXM_WSSEC_TOKEN_TYPE	X509	Static

Example 3

This example shows how to create a response that generates a SAML assertion token using the holder-of-key subject confirmation method, retrieving the subject's public key from an associated user store.

The following table shows the response attributes you must add to the response (all attributes are of type WebAgent-WS-Security-Token):

Variable Name	Variable Value	Attribute Type
TXM_WSSEC_TOKEN_TYPE	SAML	Static
TXM_WSSEC_SAML_AFFILIATE	affiliate1	Static
TXM_WSSEC_SAML_SIG _REQUIRED	hk	Static
TXM_WSSEC_SAML_USER_CERT_ SRC	User_Store	Static
TXM_WSSEC_SAML_USER_CERT	usercertificate	User attribute

Example 4

This example shows how to create a response that encrypts an incoming document and deliver the encrypted document to the web service.

The response generates a SAML assertion token using the sender vouches subject confirmation method and encrypts the SAML assertion and message body. The token and other related information are placed in a WS-Security header identified by the SOAP actor/role samlrole.

The SAML assertion and the message body are encrypted using the public key certificate found in the WS-Security header with the role pubkeyrole. The rsa-1_5 algorithm should be used to encrypt the symmetric encryption key; the tripledes-cbc algorithm should be used to encrypt the assertion and body data.

The document should be signed before encryption; the document and assertion should also be signed with a sender-vouches signature.

The following table shows the response attributes you must add to the response (all attributes are of type WebAgent-WS-Security-Token):

Variable Name	Variable Value	Attribute Type
TXM_WSSEC_TOKEN_TYPE	SAML	Static
TXM_WSSEC_SAML_AFFILIATE	affiliate2	Static
TXM_WSSEC_SAML_ROLE	samlrole	Static
TXM_WSSEC_SAML_SIG _REQUIRED	sv	Static
TXM_WSSEC_SAML_ENCRYPT_PUB_ KEY_ROLE	pubkeyrole	Static
TXM_WSSEC_SAML_ENCRYPT_ALG_ KEY	rsa-1_5	Static
TXM_WSSEC_SAML_ENCRYPT_ALG_ DATA	tripledes-cbc	Static
TXM_WSSEC_SAML_ENCRYPT_ELEM ENT	Assertion	Static
TXM_WSSEC_SAML_ENCRYPT_ELEM ENT	Body	Static
TXM_WSSEC_SAML_ENCRYPT_OR_S IGN_FIRST	sign	Static

Example 5

This example shows how to create a response that decrypts an incoming encrypted message and passes it to the associated web service in a message with a SAML assertion token.

Variable Name	Variable Value	Attribute Type
TXM_WSSEC_TOKEN_TYPE	SAML	Static
TXM_WSSEC_SAML_AFFILIATE	affiliate2	Static

Variable Name	Variable Value	Attribute Type
TXM_WSSEC_SAML_ENCRYPT_DECRYPT	yes	Static

Configure Response Attribute Caching

Responses return values to a requesting Agent. The data returned to the Agent can be a fixed value, or it may change over time. When you use a SOA Security Manager Agent to protect a resource, Agents can cache a value for fixed data, so that the value does not need to be recalculated each time the associated policy fires.

For example, a customer's account number is a fixed value, while the customer's account balance changes after each transaction. It would be more efficient to retrieve the account number once and then cache it. However, you probably want the balance to be recalculated at a regular interval to make sure the information is current.

Note: SOA Security Manager does not cache RADIUS response attributes.

To configure response attribute caching

1. Open the response.

   The associated response attributes are listed in the Attribute List group box.

2. Click the edit icon to the left of the response attribute you want.

   The Modify Response Attribute pane opens.

3. Specify the cache settings in the Attribute Caching group box.
   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

4. Click Submit.

   The cache settings are saved.

Variable Objects in Responses

You can create responses that include variable objects by incorporating them in response attributes. Variable objects can be used in response attributes to include dynamic information evaluated during the authorization of a request.

Note: Variable objects included in responses are only evaluated during the authorization of a request and not during the authentication process. Responses that include variables are limited to authorization events.

Responses can contain any number of response attributes. Each response attribute contains one variable object. Like HTTP header and cookie variables, a SOA Security Manager variable object is a name-value pair. SOA Security Manager variable objects are different from HTTP header and cookie variables, however, in that the variable object name is used to look up the variable object value at runtime. Then, in the case of response attributes, the resulting name-value pair can be returned in an HTTP header or cookie variable.

Select a Variable Using Variable Lookup

The Select Variable pane allows you to select one variable object from a list of existing variable objects.

To select a variable using variable lookup

1. Select Variable Definition as the Attribute Kind on the Attribute Setup group box.
2. Click Lookup on the Attribute Fields group box.

   The Select Variable pane opens.
3. Select one variable object from the list, and click OK.

   The Create Response Attribute pane reopens, and the name of the variable object is displayed in the Variable field on the Attribute Fields group box.

Configure a Response Attribute that Contains a Variable

A response can contain one or more response attributes whose values are determined by variable objects. Each response attribute contains one variable object. Each variable object is a name-value pair. The name of the variable object is used to look up the value of the variable object at runtime. SOA Security Manager passes the resulting name-value pair to the Web Agent.

To configure a response attribute that contains a variable

1. Follow the instructions in Configure a Response to create a response.
2. Select SiteMinder and Web Agent as the Agent Type on the Attributes group box.
3. Click Create Response Attribute on the Attribute List group box.

   The Create Response Attribute pane opens.
4. Select a response attribute from the drop-down list on the Attribute Type group box.
5. Select the type of response attribute on the Attribute Kind group box.

6. Type the name of the variable object in the Variable Name field on the Attribute Fields group box.

   Note: When this field is required, SOA Security Manager passes this name to the Web Agent in the form of a name-value pair.

7. For the selected response attribute type, complete the following fields on the Attribute Fields group box:

   Static

   Specify the value of the static variable in the Variable Value field.

   User Attribute

   Specify the name of the user attribute in the Attribute Name field.

   DN Attribute

   Specify the DN of the user or user group in the DN Spec field and the name of the user attribute in the Attribute Name field.

   (Optional) Click Lookup to search for and select one set of users or user group in a specified user directory.

   (Optional) Select the Allow Nested Groups checkbox.

   Active Response

   Specify the name of your library, the name of a library function, and optionally the names of parameters in the Library Name, Function Name, and Parameters fields.

   Note: Your library must be based on the SiteMinder Authorization API.

   Variable Definition

   Click Lookup to select an existing variable object for the Variable field.

   Note: SOA Security Manager uses the information that you provide in the fields on the Attribute Fields group box to determine the value that it passes to the Web Agent in the form of a name-value pair.

8. Click OK.

   The response attribute is saved.

More information:

Select a Variable Using Variable Lookup (see page 378)

Edit a Response

You can edit all of the properties of a response, except the Agent Type. If you want to change the Agent Type, you must delete the response and create a new one.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Delete a Response

Deleting a response removes the response from any policies with which it is associated.

It may take a short amount of time for all deleted objects to be removed from caches.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Response Groups

A response group is a collection of responses that are logically grouped so they can be applied to a single rule within a policy. All relevant responses in a response group will fire when a rule paired with the response group fires.

Response groups allow you to combine multiple responses in a single object. When you create policies, you can more easily associate multiple responses with a single rule within those policies.

Configure a Response Group

You can create a response group that applies a set of responses to one rule in a policy.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure a response group

1. Click Policies, Domains.
2. Click Response Group, Create Response Group.

   The Create Response Group pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create Response Group: Select Domain pane opens.
4. Select a domain name from the drop-down list, and click Next.

   The Create Response Group: Define Response Group pane opens.
5. Type the name and a description of the response group in the fields on the General group box.
6. Select SiteMinder and the Web Agent Agent Type on the Attributes group box.

   Note: The specified Agent type must correspond to the Agent type of the responses in the group. Only responses with the specified Agent type are available for inclusion in the group.
7. Click Add/Remove on the Group Members group box.

   The Choose responses pane opens.

   Note: The Available Members column lists all responses that are defined in the specified domain for the specified Agent type. When the Agent type is Generic Radius, the Available Members column lists all responses that are supported by Radius agents.
8. Select one or more responses from the list of Available Members, and click the right-facing arrows.

   The responses are removed from the list of Available Members and added to the list of Selected Members.

   Note: To select more than one member at a time, hold down the Ctrl key while you click on the additional members. To select a block of members, click on the first member and then hold down the Shift key while you click on the last member in the block.
9. Click OK.

   The selected responses are added to the response group.
10. Click Submit.

    The Create Response Group Task is submitted for processing.

Add Responses to a Response Group

You can add responses of the same Agent type to a response group. All of the responses must exist in the same domain.

To add responses to a response group

1. Open the response group.
2. Click Add/Remove in the Group Members group box.

   The Choose responses group box opens. The Available Members column contains responses available from the selected domain and with the specified Agent type or RADIUS vendor type.

   Note: The Available Members column lists all of the responses supported by RADIUS agents if you specified Generic RADIUS.
3. Move responses to the Selected Members column to include them in the group, and click OK.

   The Response Group pane opens. The selected rules open in the Group Members group box.
4. Click Submit.

   The response group is saved.

Add Responses to a Response Group

You can add responses of the same Agent type to a response group. All of the responses must exist in the same domain.

To add responses to a response group

1. Open the response group.
2. Click Add/Remove in the Group Members group box.

   The Choose responses group box opens. The Available Members column contains responses available from the selected domain and with the specified Agent type.
3. Move responses to the Selected Members column to include them in the group, and click OK.

   The Response Group pane opens. The selected rules open in the Group Members group box.
4. Click Submit.

   The response group is saved.

Modify a Response Group

You can modify all of the properties of a response group, except Agent type.

To change the Agent type, delete the response group and create a new one.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Delete a Response Group

Deleting a response group only deletes the grouping, not the individual responses contained in the group.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Chapter 16: Policies

This section contains the following topics:

Policy Overview (see page 385)
How to Configure a Policy (see page 388)
Exclude a User or Group from a Policy (see page 392)
Allow Nested Groups in Policies (see page 393)
AND Users/Groups Check Box (see page 393)
Specify AND/OR Relationships between Users/Groups (see page 395)
Add Users by Manual Entry (see page 396)
Enhance Policy Server's LDAP Authorization Performance (see page 397)
Add an LDAP Expression to a Policy (see page 398)
Enable and Disable Policies (see page 399)
Advanced Policy Options (see page 400)
Policy Binding Establishment (see page 404)
Delete a Policy (see page 416)
Bind Policies to SQL Queries (see page 416)
Expressions in Policies (see page 417)

Policy Overview

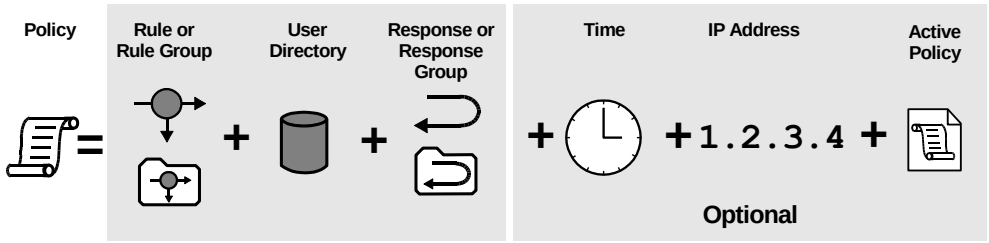
Policies define how authorization decisions are made when web service consumers interact with your web service resources. When you create policies in the Administrative UI, you link together (bind) objects that identify users, resources, and actions associated with the resources.

Policies are stored in policy domains. When you configure a policy, you can select users and groups from the user directories available in the policy domain.

SOA Security Manager identifies resources through rules. When you create a policy, you can select rules that specify the resources you want to include in a policy.

Once you identify users and resources in a policy, you can specify actions that should take place when those users access the specified resources. These actions take the form of responses. Policies can include responses that allow or deny access to a resource, customize a user's session time, redirect the user to other resources, or customize the content the user receives based on attributes contained in a user directory.

The following diagram illustrates all of the possible parts of a policy. These parts are described briefly following the diagram, and in more detail throughout the rest of this chapter.



Rules/Rule Groups

A policy must contain at least one rule or rule group. A rule identifies a specific resource or resources that are included in the policy.

Users

A policy must specify the users or groups of users that are affected by the policy. Connections to these users or groups of users must be configured on the SOA Security Manager User Directory pane. Only users or user groups for directories that are included in the policy domain in which the policy is located may be associated with a policy.

Responses

A response defines the action that is triggered when a user accesses a resource specified in a rule. Responses can return attributes from a user directory for use by other applications or to the customize the appearance of a resource. Responses can also trigger actions based on authentication and authorization events.

(Optional) IP Addresses

A policy may be limited to specific user IP addresses. Once you add an IP address restriction to a policy, if a user attempts to access a resource from an IP address not specified in the policy, the policy will not fire for the user, and therefore will not allow/deny access or process any responses.

(Optional) Time Restrictions

A policy may be limited to specific days or ranges of hours. A policy with a time restriction will not fire outside specified times, and therefore will not allow/deny access to protected resources or process any responses.

(Optional) Active Policies

An Active policy allows business logic external to SOA Security Manager to be included in a policy definition. Active policies allow SOA Security Manager to interact with custom software created using the SOA Security Manager APIs.

Policies Explanation

Policies bind other Policy Server objects together into a logical group that determines how the objects should interact. By linking together users that are accessible through directory connections, rules that point to specific resources, and responses that define actions, policies define who is authorized to access resources. Responses included in policies can also provide personalization by retrieving directory attributes when a user accesses a resource.

When one of the users specified in a policy attempts to access a resource identified in one of the policy's rules, the Policy Server uses the information contained in the policy to resolve whether or not the user can access the resource, and if any personalization should take place.

More advanced policies can be restricted to certain time periods or certain user IP addresses. This allows administrators of a group of resources a finer control over their resources.

Policy Bindings

A policy binding is the method used to link a user with a policy. The Policy Server only resolves policies for users who are part of a policy binding created by the users or groups contained in a policy.

Before the Policy Server can resolve a user's attempt to access a protected resource, the user must be authenticated. When SOA Security Manager authenticates a user, it establishes a context for the user. The user context provides information about who the user is and what privileges the user has when accessing resources.

For example, if a user is part of the group in a user directory called Employees, when the user authenticates, the Policy Server creates a policy binding for the user's membership in the group Employees. When the user attempts to access a resource protected by a rule in a policy that allows access for Employees group members, the user's policy binding allows SOA Security Manager to authorize the user.

More information:

Policy Binding Establishment (see page 404)

How to Configure a Policy

The following process lists the steps for configuring a policy.

Note: You can also create policies using the Scripting Interface for Perl. For more information, see the *Programming Guide for Perl*.

To configure a policy

1. Create the policy (see page 388).
2. Add users to the policy (see page 389).
3. Add one or more rules to the policy (see page 390).
4. (Optional) Associate responses or response groups with rules (see page 390).
5. (Optional) Associate global responses with rules (see page 391).
6. (Optional). Configure advanced policy options (see page 400).

More information:

Start the Administrative UI (see page 42)
Add Users to a Policy (see page 389)
Add Rules to a Policy (see page 390)
Associate a Rule with a Response or Response Group (see page 390)
Associate a Rule with a Global Response (see page 391)
Advanced Policy Options (see page 400)

Create the Policy

You can create a policy by adding it to a new or existing domain. Policies define relationships between users and resources.

To create a policy and add it to an existing domain

1. Click the Policies, Domains.
2. Click Domain, Modify Domain.

   The Modify Domain pane opens.
3. Specify search criteria, and click Search.

   A list of domains that match the search criteria opens.
4. Select a domain, and click Select.

   The Modify Domain: *Name* pane opens.

5. Click the Policies tab on the Domain pane.

   The Policies group box opens.

6. Click Create.

   The Create Policy pane opens.

7. Verify that Create a new object is selected, and click OK.

   The Create Policy: *Name* pane opens.

8. Type the name and a description of the policy in the fields on the General group box.

9. Click the Users tab.

   The User Directories group box opens.

10. Add users, user groups, or both to the policy, and click Submit.

    The Modify Domain: *Name* pane reopens.

11. Click Submit.

    The Modify Domain Task is submitted for processing.

Add Users to a Policy

You can add individual users, user groups, or both to a policy and create a policy binding between the added users and the policy. When a user tries to access a protected resource, the policy verifies that the user is part of its policy binding and then fires the rules included in the policy to see if the user is allowed to access the resource.

To add users to a policy

1. Click the Users tab on the Policy pane.

   The User Directories pane opens and contains group boxes for each user directory associated with the policy domain.

2. Add users or groups from the user directory to the policy.

   From within each user directory group box, you can choose Add Members, Add Entry, Add All. Depending on which method you use to add users to the policy, a dialog box will open enabling you to add users.

   Note: If you select Add Members, the User/Groups pane opens. Individual users are not displayed automatically. Use the search utility to find a specific user within one of the directories.

   You can edit or delete a user or group by clicking the right arrow (>) or minus sign (-), respectively.

3. Select individual users, user groups, or both using whatever method and click OK.

   The User Directories pane reopens and lists the policy's new users on the user directory's group box.

The task of binding users to the policy is complete.

More information:

View User Directory Contents (see page 167)
Policy Binding Establishment (see page 404)

Add Rules to a Policy

Rules indicate the specific resources included in a policy and whether to allow or deny access to the resources when the rule fires. Responses indicate the actions that should take place when the rule fires.

Note: You must add at least one rule or rule group to a policy.

To add rules or rule groups to a policy

1. Click the Rules tab on the Policy pane.

   The Rules group box opens.

2. Click Add Rule.

   The Available Rules pane opens.

3. Select the individual rules, rule groups, or both that you want to add to the policy, and click OK.

   The Rules group box lists the added rules and groups.

4. (Optional) Associate the rule with a response or response group.

   Note: To remove a rule or rule group from a policy, click the minus sign (-) to the right of the rule on the Rules group box. To create a new rule, click New Rule on the Available Rules pane.

Associate a Rule with a Response or Response Group

You can associate a response or response group with a rule in a policy. When the rule fires, the associated response also fires.

To associate a rule with a response or response group

1. Click Add Response for the rule or rule group for which you want to associate a response.

   The Available Responses pane opens and lists the responses and response groups that have been configured for the policy domain.

2. Select a response or response group, and click OK.

   The response opens in the Rules group box, and is associated with the respective rule.

   Note: If the response you require does not exist, click New Response to create the response.

Associate a Rule with a Global Response

You can associate a rule with an existing global response.

To associate a rule with a global response

1. Click the Rules tab on the Policy pane.

   The Rules group box opens.

2. Click the Add Response button next to the rule that you want to modify.

   The Available Responses pane opens.

   Note: Global responses, responses, and group responses are listed in that order on the Available Responses pane.

3. Select a global response, and click OK.

   The Rules group box reopens, and the selected response is added to the rule.

4. Click Submit.

   The Modify Policy Task is submitted for processing.

More information:

Global Policies, Rules, and Responses (see page 439)

Exclude a User or Group from a Policy

The Administrative UI allows you to exclude a user or group of users from a policy. This feature is very useful if you have a large user group that should be included in a policy, but you want to exclude a small subset of the group from the policy.

To exclude a user or group from a policy

1. Click the Users tab on the Policy pane.

   The User Directories pane opens.

2. On the User Directory group box, click *one* of the following:

   - Add Members
   - Add Entry
   - Add All

3. Choose the task from the following list that corresponds to the item you clicked in Step 2:

   - If you clicked Add Members, search from the Current Members list shown in the Users/Groups pane and select the check box of the user or group that you want.
   - If you clicked Add Entry, use the User Directory Search Expression Editor to create a search expression that locates the item you want to exclude.
   - If you clicked Add All, the entire group appears in the User Directory group box. Go to Step 5.

4. Click OK.

   The User Directories pane re-opens showing the user or group you chose, along with an Exclude button.

5. To exclude the selected user or group, click Exclude.

   A check mark appears to the right of the user or group in the Current Members list to indicate that the user or group is excluded from the policy. An Include button replaces the Exclude button.

   When you exclude a group from a policy, the exclusion indicates that anyone included in the policy who is a member of the excluded group (or the specifically excluded user), is not included in the policy. For example, if a policy contained the group Employees, and the excluded group Marketing, anyone who is a member of the Employees group, and not part of the Marketing group is included in the policy.

6. Click Submit.

   Your changes are submitted. The user or group will be excluded from the policy.

Allow Nested Groups in Policies

LDAP user directories can contain groups that contain other groups. In very complex directories, a hierarchy of nested groups is one way to organize tremendous amounts of user information.

For each LDAP user directory, you can specify that the policy allow nested groups. When nested groups are allowed in an LDAP directory, each user group in the directory and all sub-groups are searched when the policy is processed. When nested groups are not allowed, each user group in the directory is searched, but no sub-groups can be searched, when the policy is processed.

To allow nested groups in a policy that contains an LDAP user directory

1. Click the Users tab on the Policy pane.

   The User Directories pane opens and contains group boxes that correspond to the user directories associated with the policy domain.

2. Select the Allow Nested Groups check box for each user directory that contains nested groups, and click Submit.

   The Modify Policy Task is submitted for processing, and nested groups are allowed for the specified LDAP user directories.

AND Users/Groups Check Box

The AND Users/Groups check box lets you restrict authorization to users who are members of more than one user group or to a particular user who is a member of one or more user groups. When adding individual users and user groups in a user directory to a policy, you can specify AND relationships between them by selecting the check box. Alternately, you can specify OR relationships between them by clearing the check box.

When you specify AND relationships and apply the resulting policy to a user, the user must meet the following requirements to be authorized:

- The user must be a member of each user group that is bound to the policy.
- If an individual user is bound to the policy, the user must be that individual.

Note: A user who is excluded from the policy or is a member of a group that is excluded from the policy cannot be authorized.

Example: Assume that User1, Group1, and Group2 are all bound to a policy and that AND relationships are specified. In this case, test_user must be User1 and a member of Group1 and Group2 to be authorized.

Example: Assume that User1, User2, and Group1 are all bound to a policy and that AND relationships are specified. In this case, test_user cannot be both User1 and User2. Therefore, test_user cannot be authorized.

Important! Do not add two or more individual users to a policy and specify AND relationships. Because no single user can be more than one individual, the policy always fails.

To specify both AND and OR relationships, choose one of the following configurations:

- A Single Policy and Multiple User Directories

  In this configuration, two or more user directories are available to a single policy. The relationship between individual users and user groups in a single directory can be AND or OR. The relationship between individual users and user groups in different directories is always OR.

  Example: There are two user directories and a single policy. In each directory, there are two user groups, and an AND relationship is specified. Assume that Directory1 contains Group1 and Group2 and that Directory2 contains Group3 and Group4. In this case, test_user must be a member of Group1 and Group2 or a member of Group3 and Group4 to be authorized.

  This can be expressed logically as follows:

  Directory1(Group1 AND Group2) OR Directory2(Group3 and Group4)

  Use Case: There are two user directories and a single policy. Directory1 contains the user groups Facilities and Human_Resources, and an AND relationship is specified. Directory2 contains the user groups Marketing and Sales, and an OR relationship is specified. In this case, the user must be a member of Facilities and Human_Resources or a member of Marketing or a member of Sales to be authorized. This can be expressed logically as follows:

  Directory1(Facilities AND Human_Resources) OR Directory2(Marketing OR Sales)

- Multiple Policies and a Single User Directory

  In this configuration, two or more policies in a shared domain have access to a single user directory. The relationship between individual users and user groups in the user directory can be AND in one policy and OR in another policy. The relationship between different policies in a shared domain is always OR.

  Example: There are two policies and one user directory. The user directory contains four user groups. Assume that Group1 and Group2 are bound to Policy1 and that Group3 and Group4 are bound to Policy2. AND relationships are specified between the user groups in both policies. In this case, test_user can be authorized by the application of Policy1 or Policy2. This can be expressed logically as follows:

  Policy1(Group1 AND Group2) OR Policy2(Group3 AND Group4)

Use Case: There are two policies and one user directory. The user groups Human_Resources, Marketing, and Sales are bound to Policy1, and an OR relationship is specified. The user groups Facilities and Human_Resources are bound to Policy2, and an AND relationship is specified. In this case, the user must be a member of Human_Resources, Marketing, or Sales or a member of Facilities and Human_Resources to be authorized. The second policy only authorizes members of Facilities who are also members of Human_Resources.

This can be expressed logically as follows:

Policy1(Human_Resources OR Marketing OR Sales) OR Policy2(Facilities AND Human_Resources)

Specify AND/OR Relationships between Users/Groups

The AND Users/Groups check box lets you restrict authorization to users who are members of more than one user group or to a particular user who is a member of one or more user groups. When adding individual users and user groups in a user directory to a policy, you can specify AND relationships between them by selecting the check box. Alternately, you can specify OR relationships by clearing the check box.

When you specify AND relationships and apply the resulting policy to a user, the user must meet the following requirements to be authorized:

- The user must be a member of each user group that is bound to the policy.
- If an individual user is bound to the policy, the user must be that individual.

Important! Do not add two or more individual users to a policy and specify AND relationships. Because no single user can be more than one individual, the policy always fails.

To specify AND relationships between a user and one or more user groups or between multiple user groups in one user directory

1. Click the Users tab on the Policy pane.

   The User Directories pane opens, and each user directory is displayed in a separate group box.

2. Select the AND Users/Groups check box corresponding to each user directory for which you want to specify AND relationships.

3. Click Submit.

   The task is submitted for processing.

Add Users by Manual Entry

In addition to using the Available Members list in the Policy Users/Groups Dialog to specify the users and groups to include in a policy, you can specify a user or search string in the Manual Entry group box.

To add a user or group by manual entry

1. Click the Policies tab, and then click Domains, Modify Policy.

   The search window appears.

2. (Optional) Fill out the search form to narrow your search criteria.

3. Click Search.

   A list of policies appears.

4. Click the option button on the left of the policy you want, and then click Select.

   The Modify Policy: *Name* pane appears.

5. Click the Users tab.

   The user directories associated with the domain appear in the User Directories group box.

6. In the Policy Users/Groups Dialog, do one of the following:

   - For Active Directory user directories, specify the following settings:

     Manual Entry Field

     Specifies a search filter for the Active Directory user directory.

     Validate Entry Check Box

     Specifies whether the search filter is validated before the entry is added to the Active Directory user directory.

     Note: If validation of the Active Directory search filter fails, clear this check box.

     Default: Selected

   - For LDAP directories, select *one* of the following search types from the Where to Search drop-down list:

     Validate DN

     Locates the DN in the directory.

     Search Users

     Limits search to matches in user entries.

     Search Groups

     Limits search to matches in group entries.

Search Organizations

Limits search to matches in organization entries.

Search Any Entry

Limits searches to matches in user, group, and organization entries.WinNT directories

- For Microsoft SQL Server, Oracle, and WinNT directories, type a user name in the Manual Entry field.

  Note: For Microsoft SQL Server and Oracle, you can type a SQL query in the Manual Entry field instead of a user name.

  Example: SELECT NAME FROM EMPLOYEE WHERE JOB ='MGR';

The Policy Server executes the query as the database user specified in the Username field of the Credentials and Connection tab for the user directory. Before constructing the SQL statement for the Manual Entry field, become familiar with the database schema for the user directory. For example, if you are using the SmSampleUsers schema and want to add specific users, you could select from the SmUser table.

Note: For an LDAP directory, you can enter all in the Manual Entry field to bind the policy to the entire LDAP directory.

7. Click Add to Current Members.

   The Administrative UI adds the user or query to the Current Members list.

8. Click OK to save your changes and return to the Modify Policy: *Name* pane.

Enhance Policy Server's LDAP Authorization Performance

You can enhance the Policy Server's authorization performance for users stored in LDAP user directories by limiting the role-based authorization to a specific user record rather than the user's role, as follows:

To enhance the policy server's performance

1. Click the Users tab on the Modify Policy pane.

   The User Directories pane opens and contains the group boxes that correspond to the user directories associated with the policy domain.

2. If the directory on which you want to enhance the authorization performance already appears in a group box, go to Step 8.

3. If the directory you want does not appear, click Add Members on the directory's group box.

   The Users/Groups pane opens and lists the users and groups in the selected user directory.

4. Select a Search type from the drop-down list:

   Attribute-value

   Specifies a user attribute name and value pair.

   Expression

   Specifies a SOA Security Manager expression.

5. Type the user attribute name and value required for authorization in the Attribute and Value fields on the Users/Groups group box.
6. Click GO to search the directory.

   A list of directories appears.

7. Select the check box of the directory you want to add, and then click OK.

   The Users/Groups pane closes and the User Directories pane appears. The directory you selected appears in the group box.

8. Click the Edit (arrow) icon to the left of the directory.

   The User Directory Search Expression Editor appears.

9. Ensure that Validate DN appears in the Where to Search drop-down list, and then click OK.

   The User Directory Search Expression Editor closes. The Policy Server's LDAP search is done within the context of the current user and not in the LDAP server's base DN. This optimization decreases the load on the LDAP server and Policy Server, which allows quicker authorization responses.

Add an LDAP Expression to a Policy

If you create a policy in a policy domain that contains connections to an LDAP user directory, you can use the User Directory Search Expression Editor to bind an LDAP search expression to a policy. Search expressions can bind users to a policy based on attributes that appear in user, group, and organization profiles.

To add an LDAP expression to a policy

1. Click the Policies tab, and then click Domains, Modify Policy.

   The search window appears.

2. (Optional) Fill out the search form to narrow your search criteria.
3. Click Search.

   A list of policies appears.

4. Click the radio button on the left of the policy you want, and then click Select.

   The Modify Policy: *Name* pane appears.

5. Click the Users tab.

   The user directories associated with the domain appear in the User Directories group box.

6. Click Add Entry for the user directory on which the LDAP search expression is to apply.

   The User Directory Search Expression Editor appears.

7. Build an LDAP expression that binds a particular user, group, or organization attribute to your policy.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

8. Click OK.

   The expression appears in the user directory table.

Enable and Disable Policies

The Administrative UI allows you to enable and disable policies. By default, when you create a policy, the policy is enabled. When a policy is enabled, rules contained in the policy fire when users attempt to access the resources specified in the rules.

If you disable a policy, the rules contained in the policy still fire, but no user will be authorized by the policy. Any resources specified in rules contained in the policy are still protected. Until you enable the policy, no users may access resources associated with the rules specified in the policy. However, if another enabled policy allows access to a resource in the disabled policy, users associated with the enabled policy may access the resource.

To enable or disable a policy

1. Open the policy.
2. Select or clear the Enabled check box.

   If the check box is selected, the policy is enabled. If the check box is cleared, the policy is disabled. A disabled policy does not fire.

3. Click Submit.

   The policy is saved.

Advanced Policy Options

There are a number of advanced features you can use when setting up policies in the Administrative UI. These features include the following:

- IP Addresses

  This feature lets you specify certain IP addresses that a user must be using in order for a policy to fire.

- Time Restrictions

  This feature lets you specify times during which the policy fires. If you add a time restriction to a policy, the policy is effectively disabled outside of the specified times.

- Active Policies

  This feature lets you have a function call invoked in a shared library created with the SOA Security Manager API. The function call determines whether or not the policy fires.

More information:

Enable and Disable Policies (see page 399)

Allowable IP Addresses for Policies

You specify that a policy should only fire for users who access the policy's resources from a specific:

- IP address
- host name
- subnet mask
- range of IP addresses

For example, if you include a rule that allows access to resources in the policy, and then you specify a range of IP addresses, only those users who login in from one of the specified IP addresses will be allowed access to the protected resources.

Specify a Single IP Address

You specify a single IP address to ensure that the policy only fires for users who access the policy's resources from the specified IP address.

To specify single IP address

1. Open the policy.
2. Click Add in the IP Address group box.

   Settings for IP addresses appear.
3. Select the Single Host radio button.

   Settings specific to a single host appear.
4. Enter the IP Address, and click OK.

   The IP address appears in the IP Address group box.

   Note: If you do not know the IP address, but have the domain name for the address, click DNS Lookup. Enter a fully qualified host name to look up the IP address.
5. Click Submit.

   The policy is saved.

Specify a Host Name

You specify a host name to ensure the policy only fires for users who access the policy's resources from the specified host.

To specify a host name

1. Open the policy.
2. Click Add in the IP Address group box.

   Settings for IP Addresses appear.
3. Select the Host Name radio button.

   Settings specific to a host name appear.
4. Enter the host name, and Click OK

   The host name appears in the IP Address group box.
5. Click Submit.

   The policy is saved.

Add a Subnet Mask

You specify a subnet mask to ensure the policy only fires for users who access the policy's resources from the specified subnet mask.

To add a subnet mask

1. Open the policy.
2. Click Add in the IP Address group box.

   Settings for IP Addresses appear.
3. Select the Subnet Mask radio button.

   Settings specific to the subnet mask appear.
4. Enter an IP address in the IP Address field.

   Note: If you do not know the IP address, but have the domain name for the address, click DNS Lookup. Enter a fully qualified host name to look up the IP address.
5. Enter a subnet mask in the Subnet Mask field.
6. Click OK.

   The subnet mask appears in the IP Address group box.
7. Click Submit.

   The policy is saved.

Add a Range of IP Addresses

You specify a range of IP addresses to ensure that the policy only fires for users who access the policy's resources from one of the IP addresses included in the range of addresses.

To add a range of IP addresses

1. Open the policy
2. Click Add in the IP Address group box.

   Settings IP Addresses appear.
3. Select the Range radio button.

   Settings specific to a range of IP addresses appear.
4. Enter a starting IP Address in the From field.

   Note: If you do not know an IP address, but have the domain name for the address, click DNS Lookup. Enter a fully qualified host name to look up the IP address.
5. Enter an ending IP address in the To field.

6. Click OK.

   The range of IP addresses appears in the IP Address group box.

7. Click Submit.

   The policy is saved.

Time Restrictions for Policies

The Administrative UI lets you add time restrictions to a policy. When you add a time restriction, the policy only fires during the period specified by the time restriction. If a user attempts to access a resource outside of the period specified by the time restriction, the policy does not fire.

For example, if you create a time restriction for a policy that secures access to a resource, and specifies that the policy will only fire from 9am - 5 pm, Monday - Friday. A user will only be authenticated and authorized during the times indicated in the time restriction. The resources protected by the policy will not be available outside the times indicated.

Note: Time restrictions are based on the system clock of the server on which the Policy Server is installed.

How Rule and Policy Time Restrictions Interact

If you specify a time restriction for a policy, and that policy contains a rule with a time restriction, the policy fires during the times that are intersection of the two restrictions.

For example, if a policy has a time restriction of 9AM to 5PM, and a rule has a time restriction of Monday through Friday, then the policy only fires between 9AM and 5PM, Monday through Friday.

Add Time Restrictions to a Policy

You add time restrictions to a policy to ensure that the policy only fires at specific times.

To add a time restriction to a policy

1. Open the policy.
2. Click Set in the Time group box.

   The Time Restrictions pane appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

3. Specify starting and expiration dates.

4. Specify time restrictions in the Hourly Restrictions table.

   Note: Each check box represents one hour. When a check box is selected, the rule fires during that hour, and the rule applies to the specified resources. When a check box is cleared, the rule does not fire during that hour, and the rule will not apply to the specified resources.

5. Click OK.

   The time restrictions are saved.

Configure an Active Policy

An active policy is used for dynamic authorization based on external business logic. An active policy is included in the authorization decision by having the Policy Server invoke a function in a customer-supplied shared library.

This shared library must conform to the interface specified by the Authorization API, which is available separately with the Software Development Kit.

Note: More information exists in *API Reference Guide for C.*

To configure an Active Policy

1. Open the global policy.
2. Select the Edit Active Policy check box in the Advanced Group box.

   Active policy settings appear.

3. Enter the name of the shared library in the Library Name field.
4. Enter the name of the function in the shared library that is to implement the active policy.
5. Click Submit.

   The policy is saved.

Policy Binding Establishment

The following sections describe the methods for establishing different types of policy bindings. Supported policy binding types differ based on the type of user directory in which user information is located.

Policy Bindings for LDAP Directories

When SOA Security Manager authenticates a user, it establishes a user context. Subsequently, access control policy decisions are based on the user context matching one of the criteria shown in the table below.

User Namespace	Description
User	The user's Distinguished Name (DN) must match the DN specified in the policy.
User Attribute	The search expression specifying conditions related to user attributes must be true.
User Group	The user's DN must be a member of the user group specified in the policy.
Group Attribute	The search expression specifying conditions related to the group attribute must be true.
Organizational Role	The user must occupy the organizational role specified in the policy.
Organization Unit	The user must be a member of the organizational unit specified in the policy. The Organizational Unit must be a part of a user's DN, group, or role (group and role are not used by default).
Organization	The user must be a member of the organization specified in the policy. The Organization must be a part of a user's DN, group, or role (group and role are not used by default).
Organization Attribute	The search expression specifying conditions related to the organization attribute must be true.
Custom Object Classes	SOA Security Manager can be configured to associate Policies with custom directory objects.

Generally, you bind users or user attributes to policies on the SOA Security Manager Policy pane by selecting an entry from the list of available directory entries. Individual users are not visible in the list of available directory entries. However, you can search for specific users within a directory and add the users directly to the policy.

More information:

Add Users to a Policy (see page 389)

Bind Policies to Users with the Manual Entry Field

There are two ways to bind individual users to a policy. The first is by using the Manual Entry field in the SOA Security Manager Policy Users/Groups dialog. The second is by using the Search feature in the SOA Security Manager Policy Users/Groups dialog.

To bind users to a policy with the Manual Entry Field

1. From the Users tab on the SOA Security Manager Policy dialog, locate the group box with the user directory that you want to search, and then click Add Entry.

   The User Directory Search Expression Editor opens.

2. Click the Where to Search drop-down list, and then select Search Users.

3. In the Manual Entry field, specify a user DN.

   For example: uid=JSmith, ou=people, o=myorg.org

   Note: This user DN must match exactly the user's distinguished name. This feature will not match a subset of information contained in the user's DN.

4. Click OK.

   The User Directory Search Expression Editor closes and the user DN you entered appears in the group box of the directory.

5. Click Submit to save your changes.

More information:

Add Users by Manual Entry (see page 396)
Add Users to a Policy (see page 389)

Bind Policies to Users with the Search Feature

On the User Directories pane, there are two ways to bind individual users to a policy. You can click Add Members on a user directory group box and use the attribute-value feature on the Users/Groups pane. Or you can click Add Entry on a user directory group box and use the User Directory Search Expression Editor.

To bind individual users to a policy by using the search feature

1. Click the Users tab on the Policy pane.

   A list of the user directories that are associated with the domain opens on the User Directories pane.

2. Click Add Members on a user directory group box.

   The Users/Groups pane opens.

3. Specify search criteria, and click Go

   A list of users that match the search criteria opens.

4. Select the users that you want, and click OK.

   The User Directories pane reopens, and the selected users are added to the user directory group box.

5. Click Submit.

   The Create or Modify Policy task is submitted for processing.

More information:

Add Users by Manual Entry (see page 396)
Add Users to a Policy (see page 389)
Search User Directories (see page 167)

Bind Policies to User Attributes

To bind a policy to user attributes, specify an LDAP search expression that defines conditions related to user attributes that must be true. For example, to bind a policy to all people whose location *(l)* is *westcoast or* whose mail address *(mail)* ends with *string.com*, insert the following search expression (using a pipe (|) at the beginning) in the Manual Entry field:

(|(l=westcoast)(mail=*string.com))

More information:

Add an LDAP Expression to a Policy (see page 398)

Bind Policies to User Groups

User Groups open in Users/Groups pane.

To bind a policy to a User Group

1. Click the Users tab.

   The user directories associated with the domain open in the User Directories group box.

2. Click Add Members.

   The Users/Groups pane opens.

3. Select the user group.

4. Click OK.

   The User Directories group box opens. The respective user directory table lists the user group to which the policy should apply.

Bind Policies to Organizational Roles

SOA Security Manager allows you to bind policies to an Organizational Role. When you bind a policy to an organizational role, users must be a member of the role in order for the policy to fire.

To bind organizational roles to a policy with the Manual Entry Field

1. From the Users tab on the SiteMinder Policy dialog, locate the group box with the user directory that you want to search, and then click Add Entry.

   The User Directory Search Expression Editor opens.

2. Click the Where to Search drop-down list, and then select Search Users.

3. In the Manual Entry field, specify an organizational role.

4. Click OK.

   The User Directory Search Expression Editor closes and the organizational role you entered appears in the group box of the directory.

5. Click Submit to save your changes.

   The organizational roles are bound to the policy.

Bind Policies to Group Attributes

To bind a policy to group attributes, specify an LDAP search expression that defines conditions related to group attributes that must be true.

To bind policies to group attributes

1. From the Users tab on the SOA Security Manager Policy dialog, locate the group box with the user directory that you want to search, and then click Add Entry.

   The User Directory Search Expression Editor opens.

2. Click the Where to Search drop-down list, and then select Search Users.

3. In the Manual Entry field, specify a group. For example, to bind a policy to all groups located in the state of Massachusetts in the USA, insert the following search expression in the Manual Entry field:

   (&(c=USA)(s=Massachusetts))

4. Click OK.

   The User Directory Search Expression Editor closes and the group you entered appears in the group box of the directory.

5. Click Submit to save your changes.

More information:

Add an LDAP Expression to a Policy (see page 398)

Bind Policies to Organization Units

To bind a policy to an organizational unit, specify an LDAP search expression that defines an organizational unit.

To bind organization units to a policy with the Manual Entry Field

1. From the Users tab on the SOA Security Manager Policy dialog, locate the group box with the user directory that you want to search, and then click Add Entry.

   The User Directory Search Expression Editor opens.

2. Click the Where to Search drop-down list, and then select Search Organizations.

3. In the Manual Entry field, specify an organization unit. For example, to bind a policy to all people whose organization unit (ou) is marketing, insert the following search expression in the Manual Entry field:

   ou=Marketing

4. Click OK.

   The User Directory Search Expression Editor closes and the user DN you entered appears in the group box of the directory.

5. Click Submit to save your changes.

   The organization unit is bound to the policy.

Bind Policies to Organizations

To bind a policy to an organization, specify an LDAP search expression that defines an organization.

To bind organizations to a policy with the Manual Entry Field

1. From the Users tab on the SOA Security Manager Policy dialog, locate the group box with the user directory that you want to search, and then click Add Entry.

   The User Directory Search Expression Editor opens.

2. Click the Where to Search drop-down list, and then select Search Organizations.

3. In the Manual Entry field, specify an organization.

   For example, to bind a policy to all people whose organization *(o)* is *myorg.org*, insert the following search expression in the Manual Entry field:

   o=myorg.org

4. Click OK.

   The User Directory Search Expression Editor closes and the organization you entered appears in the group box of the directory.

5. Click Submit to save your changes.

   The organization is bound to the policy.

Bind Policies to Organization Attributes

To bind a policy to organization attributes, specify an LDAP search expression that defines conditions related to organization attributes that must be true.

To bind users to a policy with the Manual Entry Field

1. From the Users tab on the SOA Security Manager Policy dialog, locate the group box with the user directory that you want to search, and then click Add Entry.

   The User Directory Search Expression Editor opens.

2. Click the Where to Search drop-down list, and then select Search Organizations.

3. In the Manual Entry field, specify an organization attribute. For example, to bind a policy to all organizations located in the state of Massachusetts in the USA, insert the following search expression in the Manual Entry field:

   (&(c=USA)(s=Massachusetts))

4. Click OK.

   The User Directory Search Expression Editor closes and the organization attribute you entered appears in the group box of the directory.

5. Click Submit to save your changes.

   The policy is bound to the organization attributes.

More information:

Add an LDAP Expression to a Policy (see page 398)

Binding Policies to Custom Object Classes

SOA Security Manager can be configured to bind policies to custom object classes. If you have the Software Development Kit installed, see the *API Reference Guide for C* for more information.

Policy Bindings for WinNT User Directories

When SOA Security Manager authenticates a user, it establishes a user context. Subsequently, access control policy decisions are based on the user context matching one of the criteria shown below.

User Namespace	Description
User	The user's user name must match the user name specified in the policy.
User Group	The user must be a member of the user group specified in the policy.

Generally, you bind users to policies on the Policy pane by selecting an entry from the list of available directory entries. However, individual users are not visible in the list of available directory entries.

More information:

Add Users to a Policy (see page 389)

Bind Policies to Users with the Manual Entry Field

On the User Directories pane, there are two ways to bind individual users to a policy. You can click Add Members on a user directory group box and use the attribute-value search feature on the Users/Groups pane. Or you can click Add Entry on a user directory group box and use the User Directory Search Expression Editor.

To bind individual users to a policy by using the Manual Entry field

1. Click the Users tab on the Policy pane.

   A list of the user directories that are associated with the domain opens on the User Directories pane.

2. Click Add Entry on a user directory group box.

   The User Directory Search Expression Editor pane opens.

3. Specify a user DN on the Condition and Infix Notation group boxes.

4. Click OK.

   The User Directories pane reopens, and the specified users are added to the user directory group box.

5. Click Submit.

   The Create or Modify Policy task is submitted for processing.

More information:

Add Users by Manual Entry (see page 396)
Add Users to a Policy (see page 389)

Bind Policies to Users with the Search Feature

On the User Directories pane, there are two ways to bind individual users to a policy. You can click Add Members on a user directory group box and use the attribute-value feature on the Users/Groups pane. Or you can click Add Entry on a user directory group box and use the User Directory Search Expression Editor.

To bind individual users to a policy by using the search feature

1. Click the Users tab on the Policy pane.

   A list of the user directories that are associated with the domain opens on the User Directories pane.

2. Click Add Members on a user directory group box.

   The Users/Groups pane opens.

3. Specify search criteria, and click Go

   A list of users that match the search criteria opens.

4. Select the users that you want, and click OK.

   The User Directories pane reopens, and the selected users are added to the user directory group box.

5. Click Submit.

   The Create or Modify Policy task is submitted for processing.

More information:

Add Users by Manual Entry (see page 396)
Add Users to a Policy (see page 389)
Search User Directories (see page 167)

Bind Policies to User Groups

You can bind a policy to a user group.

To bind a policy to a user group

1. Click the Users tab on the Policy pane.

   A list of the user directories that are associated with the domain opens on the User Directories pane.

2. Click Add Members on a user directory group box.

   The Users/Groups pane opens.

3. Select a user group.
4. Click OK.

   The User Directories pane reopens, and the selected user group is added to the user directory group box.

More information:

Add Users to a Policy (see page 389)

Policy Bindings for Microsoft SQL Server and Oracle User Directories

When SOA Security Manager authenticates a user, it establishes a user context. Subsequently, access control policy decisions are based on the user context matching one of the criteria shown in below.

User Namespace	Description
User	The user's name must match the user name specified in the policy.
User Group	The user must be a member of the user group specified in the policy.
User Attribute	The search expression specifying conditions related to user attributes must be true.
SQL query	The SQL query specifying conditions related to the user must be true.

Generally, you would bind users or user attributes to policies on the Policy Users/Groups pane by selecting an entry from the list of available directory entries. However, individual users may not be visible in the list of available directory entries (depending on the setup of Query Enumerate in the SQL query scheme for the user directory).

More information:

Add Users to a Policy (see page 389)

Bind Policies to Users with the Manual Entry Field

On the User Directories pane, there are two ways to bind individual users to a policy. You can click Add Members on a user directory group box and use the attribute-value search feature on the Users/Groups pane. Or you can click Add Entry on a user directory group box and use the User Directory Search Expression Editor.

To bind individual users to a policy by using the Manual Entry field

1. Click the Users tab on the Policy pane.

   A list of the user directories that are associated with the domain opens on the User Directories pane.

2. Click Add Entry on a user directory group box.

   The User Directory Search Expression Editor pane opens.

3. Specify a user DN on the Condition and Infix Notation group boxes.

4. Click OK.

   The User Directories pane reopens, and the specified users are added to the user directory group box.

5. Click Submit.

   The Create or Modify Policy task is submitted for processing.

More information:

Add Users by Manual Entry (see page 396)
Add Users to a Policy (see page 389)

Bind Policies to Users with the Search Feature

On the User Directories pane, there are two ways to bind individual users to a policy. You can click Add Members on a user directory group box and use the attribute-value feature on the Users/Groups pane. Or you can click Add Entry on a user directory group box and use the User Directory Search Expression Editor.

To bind individual users to a policy by using the search feature

1. Click the Users tab on the Policy pane.

   A list of the user directories that are associated with the domain opens on the User Directories pane.

2. Click Add Members on a user directory group box.

   The Users/Groups pane opens.

3. Specify search criteria, and click Go

   A list of users that match the search criteria opens.

4. Select the users that you want, and click OK.

   The User Directories pane reopens, and the selected users are added to the user directory group box.

5. Click Submit.

   The Create or Modify Policy task is submitted for processing.

More information:

Add Users by Manual Entry (see page 396)
Add Users to a Policy (see page 389)
Search User Directories (see page 167)

Bind Policies to User Groups

You can bind a policy to a user group.

To bind a policy to a user group

1. Click the Users tab on the Policy pane.

   A list of the user directories that are associated with the domain opens on the User Directories pane.

2. Click Add Members on a user directory group box.

   The Users/Groups pane opens.

3. Select a user group.

4. Click OK.

   The User Directories pane reopens, and the selected user group is added to the user directory group box.

More information:

Add Users to a Policy (see page 389)

Bind Policies to User Attributes

To bind policies to user attributes, specify a search expression that defines conditions related to user attributes that must be true.

For example, to bind a policy to all people whose area code is 555, insert the following expression in the Manual Entry field: (areacode='555').

Delete a Policy

When you are deleting a policy, remember that all of the elements in the policy still exist, it is only the grouping (or binding) of those elements that you remove when you delete the policy.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Bind Policies to SQL Queries

SOA Security Manager allows you to use the Manual Entry field to specify a SQL query to bind policies to users.

To use the Manual Entry Field to Bind Policies to a SQL Query

1. From the Users tab on the SOA Security Manager Policy dialog, locate the group box with the user directory that you want to search, and then click Add Entry.

   The User Directory Search Expression Editor opens.

2. Click the Where to Search drop-down list, and then select Search Users.

3. In the Manual Entry field, specify a user DN.

   For example: uid=JSmith, ou=people, o=myorg.org

   Note: This user DN must match exactly the user's distinguished name. This feature will not match a subset of information contained in the user's DN.

4. Click OK.

   The User Directory Search Expression Editor closes and the user DN you entered appears in the group box of the directory.

5. Click Submit to save your changes.

   For example, to bind a policy to all people whose account balance is greater than $1000, a query might look like this:

   SELECT name FROM customers WHERE balance, 1000

   The contents of the SQL query depend on your database schema.

More information:

How to Configure a Policy (see page 388)

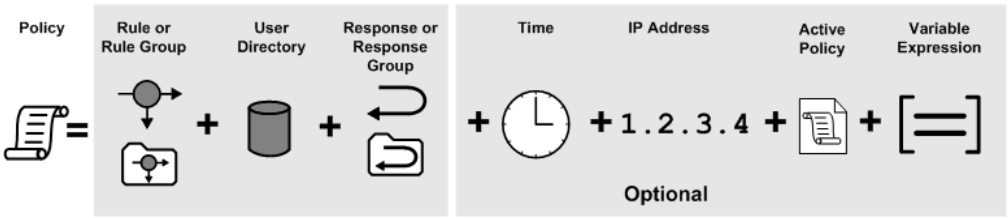
Expressions in Policies

eTelligent Rules makes available a set of variables for use in policy expressions.

Expressions extend policies to include dynamic information evaluated at runtime. Variable objects may be used in expressions to create a boolean set of conditions that determines entitlements for the resources protected by the policy.

To use variable objects in an active policy expression, you must configure a policy object and build the appropriate boolean expression using the Expression dialog. The interface is similar to the LDAP Search Expression editor described in Add LDAP Expressions to Policies (see page 398).

Note: Expressions may be added to other data supported by policy objects as shown in the following figure.



Note: Active expressions and named expressions are not the same. While both types of expressions are evaluated at run-time, they differ in the following ways:

- While active expressions are Boolean expressions, named expressions can return a string, number, or Boolean value.
- While active expressions are referenced as is and must be reentered each time that they are used, named expressions are referenced by name and can be referenced from anywhere and reused.

More information:

Variables (see page 419)

SOA Security Manager provides the following predefined variable objects that can be used in policy expressions to dynamically extend your policies to include the following information about an incoming web service request:

- Transport over which the message was sent (HTTP or HTTPS).
- SOAP message envelope (including WS-Security headers).

- Message *payload* (the body of the message, which represents the details of the transaction).
- SAML Session Ticket assertions.

Add an Expression to a Policy

You can create a Boolean expression and add it to a policy. Boolean expressions operate on variables, and the values of the variables at the time that the policy is processed affect the outcome of the processing. Thus, Boolean expressions influence policy decisions.

To add an expression to a policy

1. Click the Expression tab on the Policy pane.

   The Expression group box opens.

2. Click Edit.

   The Policy Expression pane opens.

3. Type variable names in the fields on the Condition group box, or click Variable Lookup, select an operator from the drop-down list, and click Add.

   The condition is added to the Infix Notation group box.

   Note: To create multiple conditions, repeat this step.

4. Select the conditions and click the buttons on the Infix Notation group box to create an expression.

5. Click OK.

   The Expression group box reopens, and the expression is displayed in the field on the group box.

6. Click Submit.

   The Modify Policy task is submitted for processing.

Chapter 17: Variables

This section contains the following topics:

eTelligent Rules (see page 419)
Variables Overview (see page 423)
Create a Variable (see page 427)
Configure Message-based Authorization Using an XPath Query in XmlToolkit.properties (see page 436)

eTelligent Rules

You can use eTelligent Rules to define variables that enable fine-grained access-control criteria known as policy expressions.

Policy expressions are implemented as policy attributes. They include operators and customer-defined variables that are evaluated at runtime, when a user actually needs to access a protected resource on a Web site.

Variables can store local information that is within the enterprise or remote information that is provided by various Web Services.

The variables provided by eTelligent Rules are available in the Administrative UI. You can define variable objects and incorporate them into policy logic through policy expressions. You can also include variables in SOA Security Manager response objects.

SOA Security Manager eTelligent Rules Benefits

- Reduce complexity and eliminate the need for custom code.

  Authorization access is defined by the SOA Security Manager administrator in policy expressions, using graphical tools rather than application code. There is no need to integrate and reconcile backend business applications' access control information, because that information is centralized in the SOA Security Manager Policy Server.

- Use business data dynamically in security policies.

  Defining access control to secure resources is based on local user information and incoming information, such as the amount of a purchase order placed by the user.

- Combine various types of information for authorization decisions.

  Web browser forms data, user-context data (stored locally in the Policy Server), and remote data (obtained through a service bureau) can be flexibly combined in policy expressions.

- Make transactional decisions online.

  There is no need to go back to a backend business application each time authorization is needed to access a protected resource.

- Rely on XML-based third-party security data.

  eTelligent Rules use a standard XML protocol to communicate with trusted service bureaus, thus increasing the choice of web services providers.

- Use Boolean logic.

  Policy expressions are defined by SOA Security Manager security administrators, using variables together with logical operators.

- Minimize the number of policies required.

  Due to the use of policy expressions based on logic, fewer policies are necessary, thus keeping policy administration to a minimum.

eTelligent Rules Configuration

The tasks require to configure eTelligent Rules are as follows:

- Configure variables
- Configure policy expressions that use the eTelligent Rules variables

  Variables and policy expressions are configured using the Administrative UI.

- Modify the eTelligent Rules properties files, which are:
  - JVMOptions.txt
  - LoggerConfig.properties

  You can modify only the LoggerConfig.properties file.

More information:

Variables Overview (see page 423)
Policies (see page 385)
eTelligent Rules Properties Files (see page 421)

eTelligent Rules Properties Files

The following properties files are for eTelligent Rules:

- JVMOptions.txt

  This is a required file for eTelligent Rules. The installed location of this file is: *policy_server_home*/config/

- LoggerConfig.properties

  This file is required to configure logging for eTelligent Rules. The installed location of this file is:

  policy_server_home/config/properties

More information:

JVMOptions.txt File (see page 421)
Modify the LoggerConfig.properties File (see page 421)

JVMOptions.txt File

The JVMOptions.txt file contains the settings that the Policy Server uses when creating the Java Virtual Machine that is used to support eTelligent Rules.

If you encounter errors related to missing classes, you may need to modify the classpath directive in the JVMOptions.txt file. For complete information about the settings contained in the JVMOptions.txt file, see your Java documentation.

Modify the LoggerConfig.properties File

On the Policy Server, the LoggerConfig.properties file allows you to specify logging features that are used when you start the SiteMinder service from a command line. The properties contained in this file are not used when the service is started from the Policy Server Management Console. The settings in this file are generally only used for debugging purposes.

You may want to modify this file to obtain more output for debugging purposes.

The following shows an example of a LoggerConfig.properties file.

```
// LoggingOn can be Y, N
LoggingOn=Y

// LogLevel can be one of LOG_LEVEL_NONE, LOG_LEVEL_ERROR,
LOG_LEVEL_INFO, LOG_LEVEL_TRACE
LogLevel=LOG_LEVEL_TRACE

// If LogFileName is set Log output will go to the file named
LogFileName=affwebserv.log

// AppendLog can be Y, N.  Y means append output to LogFileName if
specified
AppendLog=Y

// AlwaysWriteToSystemStreams can be Y, N.
// Y means log messages are written to System.out
// or System.err regardless of what the logger streams are
// set to.  If the logger streams are set to System.out
// or System.err log messages will be written multiple times.
// This facilitates logging messages to System.out/System.err
// and a file simultaneously.
AlwaysWriteToSystemStreams=N

// DateFormatPattern can be any valid input to java.text.DateFormat
constructor.
// See the Java documentation for java.text.DateFormat for details
// If not specified, the default format for the default locale is used
DateFormatPattern=MMMM d, yyyy h:mm:ss.S a
```

The settings in this file are:

LoggingOn

Enables or disables logging. Set this parameter to Y to enable logging. Set this parameter to N to disable logging.

LogLevel

Indicates the level of detail contained in logs. The LogLevel can be one of the following:

LOG_LEVEL_NONE

No messages will be logged.

LOG_LEVEL_ERROR

Only records error messages.

LOG_LEVEL_INFO

Records error messages and warnings.

LOG_LEVEL_TRACE

Records error messages, warnings, and general processing information that may be useful for tracking problems.

LogFileName

If LogFileName is set, all log output will go to the file named in this parameter.

AppendLog

Indicates whether log information should be appended to an existing file at startup or a new file should be created at startup. Set this parameter to Y to append output to the file specified in the LogFileName parameter. Set this file to N if a new file should be created at startup.

AlwaysWriteToSystemStreams

Set this parameter to Y to log messages to System.out or System.err regardless of what the logger streams are set to. If the logger streams are set to System.out or System.err, log messages will be written multiple times. This facilitates logging messages to System.out/System.err and a file simultaneously.

DateFormatPattern

DateFormatPattern can be any valid input to java.text.DateFormat constructor. See the Java documentation for java.text.DateFormat for details.

If not specified, the default format for the default locale is used.

Variables Overview

In the context of Policy Server, variables are objects that can be resolved to a value which you can incorporate into the authorization phase of a request. The value of a variable object is the result of dynamic data and is evaluated at runtime. Variables provide a flexible tool for expanding the capabilities of policies and responses.

Variable Types

The following types of variables are available:

- Static Variables (see page 424)
- Request Context Variables (see page 424)
- User Context Variables (see page 425)
- Form Post Variables (see page 425)
- Web Services Variables

Static Variables

Static variables consist of a simple name/value pair of a particular type, such as string, boolean, and others. The key benefit of a static variable is to implement good programming practices. Instead of repeating the value of a constant each time it's used in a policy, a static variable provides a single piece of data that can be used throughout multiple policies.

Request Context Variables

Each request processed by SOA Security Manager establishes a request context. This context identifies the following:

Action

Indicates the type of action specified in the request, such as GET or POST.

Resource

Indicates the requested resource, such as /directory_name/.

Server

Indicates the full server name specified in the request, such as server.example.com.

A request context variable may capture any of this information and make it available for inclusion in a policy expression or response. The key benefit of this type of variable is to provide fine-grained request context information without any programming logic.

User Context Variables

When the Policy Server authenticates a user against an entry in a directory, a user context is created. The user context consists of information about the user directory and the contents of the directory that pertain to the authenticated user.

User context variables can be based on an attribute of a directory connection, or based on the contents of the directory. The key benefit of this type of variable is to provide flexibility in defining rules based on particular user context without any programming logic.

Form Post Variables

HTML forms are often used to collect information required by back-end applications. Form Post variables can be used to capture any information entered in an HTML form and POSTed. For example, if the business logic associated with an application requires a purchase order amount specified on a HTML form used for logging into the application, you can create a Form Post variable object that collects the value of the purchase order supplied by a user. The variable can then be used in policies.

Important: Form Post variables are not supported by EJB or Servlet Agents. Do not use Form Post variables in policies enforced by EJB or Servlet Agents.

The key benefit of this type of variable is that it allows the Policy Server to use POST data as a part of a policy expression rather than forcing enterprises to build security logic into back end server applications. Using HTTP POST variables results in efficient network usage between Agents and Policy Servers. The Agent only needs to extract the HTTP variable information from the HTTP stream so that the information can be used during authorization processing by the Policy Server.

Variable Use in Policies

Variables allow you to include business logic in policies by capturing a wide range of dynamic data that can be built into policy expressions. When you define variable objects in the Administrative UI, you may use those variables in expressions in the Policy dialog on the Expression tab. You can build expressions that use multiple variable objects and boolean operators to capture very complex business logic in your policies.

For example, a policy may contain an expression that requires the value of a user's account type and a credit score in order to allow access to an application. An expression can be defined in the policy so that only users whose account type is "gold", and whose credit score is greater than a specific value may have access to a resource. This example requires two variables, which must be combined in an expression on the Expression tab of the Policy dialog.

More information:

Expressions in Policies (see page 417)

Note: Variables can only be used in policy expressions when using traditional (policy domain-based) policy management. They are not available when using enterprise (application-based) policy management.

Message-based Authorization Using Variables

Variables are objects that can be resolved to a value, which you can incorporate into the authorization phase of a request. The value of a variable object is the result of dynamic data and is evaluated at run time.

To make authorization decisions based on the transport header, SOAP envelope header, XML payload, or SAML assertions, you can define specific SOA Security Manager variables and add them to policies in the form of policy expressions. The Policy Server can use a policy expression as an additional criterion when determining if a client should be permitted access to a web service.

SOA Security Manager provides five variables types that represent dynamic, context-sensitive data from any layer (transport, message envelope, or message body) of an XML message. All of these variables can be used in policy expressions.

- **SAML Assertion**—Lets you obtain information from SAML assertions.
- **Transport**—Lets you to obtain HTTP header values from the web service request.
- **XML Agent**—Lets you obtain information about the web server whose resources the SOA Agent is protecting.
- **XML Body**—Lets you obtain information from any element in the body (or payload) of an incoming XML message.
- **XML Envelope Header**—Lets you obtain information from any element in the SOAP envelope (including WS-Security headers) of an incoming XML message.

Once defined, these variables can be used in policy expressions to make authorization decisions. For example, you could define an XML body variable called ShipToZipCode that corresponds to an XML query that obtains the ship-to ZIP code from a purchase order XML document.

Variable Use in Responses

Variables may be used in responses. When you define variable objects in the Administrative UI, you can use those variables in responses. The value of the response is created at runtime by the Policy Server as it resolves the value of a variable object.

Create a Variable

You create a variable to make it available for use in policies or responses. Variables are domain objects. You create them within a specific policy domain, or import them into a domain using the smobjimport tool.

More information about importing objects into policy domains exists in the *Policy Server Administration* guide.

More information:

Domains (see page 323)

Create a SAML Assertion Variable

SAML Assertion variables let you obtain information from any SAML assertion and use this information in policy expressions to authorize a client. The assertion may be included in a SOAP envelope or HTTP header of an incoming XML message. For example, you can create a variable that enables the Policy Server to check who issued the assertion before permitting access to a web service.

SAML assertion variables are resolved to the value of an XPath string. The string identifies an element (and optionally, an operation to perform on that element) of a SAML assertion.

Note: For more information about XPATH, see the XPATH specification available at http://www.w3.org/TR/xpath.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.
3. Click Create Variable.

   The Create Variable pane opens.

4. Verify that Create a new object is selected, and click OK.

   Variable settings open.

Type the variable name in the Name field.

1. Select SAML Assertion from the Variable Type list.

   SAML Assertion variable settings open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

2. Specify the data type in which the value of the specified XPATH query should be returned by choosing one of the following options from the Return Type list:
   - Boolean
   - Number
   - String (the default)
3. Type in an XPath query that you want to resolve to the variable value in the Query box.
4. Optionally, set the SAML Authentication Scheme Required box if the web service is protected by the SAML Session Ticket authentication scheme.
5. If the web service is not protected by the SAML Session Ticket authentication scheme, specify whether the SOA Agent should look for the SAML assertion in the Envelope Header or HTTP Header by selecting the appropriate SAML Assertion Location option.
6. Click Finish.

   The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Create a Transport Variable

Transport variables let you obtain HTTP header values from the web service request.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.
3. Click Create Variable.

   The Create Variable pane opens.

4. Verify that Create a new object is selected, and click OK.

   Variable settings open.

5. Type the variable name in the Name field.
6. Select Transport from the Variable Type list.

   Transport variable settings open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

7. Enter information in the following fields:

   Description

   (Optional) Specifies a brief description of the variable.

   Limits: No more than 1KB.

   Return Type

   Specifies the data type in which the value of the transport header data should be returned:

   - Boolean
   - Date
   - Number
   - String (the default)

   Transport Data Name

   Specifes the name of the HTTP header (for example, SOAPAction) that will provide the value of the variable.

8. Click Finish.

   The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Create an XML Agent Variable

SOA Agent variables let you obtain information about the web server whose resources the SOA Agent is protecting for use in policy expressions or responses.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.

3. Click Create Variable.

   The Create Variable pane opens.

4. Verify that Create a new object is selected, and click OK.

   Variable settings open.

5. Type the variable name in the Name field.
6. Select XML Agent from the Variable Type list

   XML Agent variable settings open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

7. Enter information in the following fields:

   Description

   (Optional) Specifies a brief description of the variable.

   Limits: No more than 1KB.

   Property

   Specifies the XML Agent property that will provide the value of the variable:

   - Server Product Name—String representation of the web server product name—for example, iPlanet Web Server. Value is obtained from the ServerProductName Agent Configuration parameter.
   - Server Vendor—String representation of the web server vendor—for example, Sun. Value is obtained from the ServerVendor Agent Configuration parameter.
   - Server Version—String representation of the web server product version—for example, 6.0 SP2.

8. Click Finish.

   The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Create an XML Body Variable

XML Body variables let you obtain information from any element in the body (or payload) of an incoming XML message for use in policy expressions and responses.

Specifically, XML Body variables are resolved to the value of an XPath string that identifies an element (and optionally, an operation to perform on that element) of an XML document.

Note: For more information about XPATH, see the XPATH specification available at http://www.w3.org/TR/xpath.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.
3. Click Create Variable.

   The Create Variable pane opens.
4. Verify that Create a new object is selected, and click OK.

   Variable settings open.
5. Type the variable name in the Name field.
6. Select XML Body from the Variable Type list.
7. XML Body variable settings open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
8. Enter information in the following fields:

   Description

   (Optional) Specifies a brief description of the variable.

   Limits: No more than 1KB.

   Return Type

   Specifies the data type in which the value of the specified XPATH query should be returned:

   - Boolean
   - Date
   - Number
   - String (the default)

9. Do one of the following:
   - Load a schema (.xsd) file and select the element to map to the specified field name by browsing using the following procedure:
     a. Click Browse and navigate to the schema file.
     b. Click Upload XSD File.

        The schema is uploaded.
     c. Select the schema element that you want to map to the specified field name in the Select a node group box.

        The Select a node group box displays the selected schema using a standard tree-style hierarchical view. Click the plus sign (+) next to an element to expand it. Click the minus sign (-) beside an expanded element to contract it. Elements marked with an asterisk (*) are repeatable within the XML document (that is, incoming XML documents may contain multiple instances of that element).
   - Unset the Advance XPath query option and type an XPATH query defining the mapping in the XPath field.
10. Optionally, if you are working from a loaded schema in the Select a node group box, specify an XPath function (count, div, index, mod, sum) that you want to apply to a repeatable schema element, by choosing it from the Function drop-down list.

    The Function option lets you create more complex mappings by processing functions that further evaluate the XML document.

    Note: For more information about these functions, go to the XPATH specification at http://www.w3.org/TR/xpath.
11. Click Finish.

    The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Create an XML Envelope Header Variable

XML Envelope Header Variables let you obtain information from any element in the SOAP envelope header (including WS-Security headers) of an incoming XML message, for use in policy expressions or responses.

Specifically, XML Envelope Header variables are resolved to the value of an XPath string that identifies a SOAP envelope header element (and optionally, an operation to perform on that element) of an XML document.

Note: For more information about XPATH, see the XPATH specification available at http://www.w3.org/TR/xpath.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.
3. Click Create Variable.

   The Create Variable pane opens.
4. Verify that Create a new object is selected, and click OK.

   Variable settings open.
5. Type the variable name in the Name field.
6. Select XML Header from the Variable Type list.

   XML Header variable settings open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
7. Specify the data type in which the value of the specified XPATH query should be returned by choosing one of the following options from the Return Type list:
   - Boolean
   - Number
   - String (the default)
8. Type in an XPath query that you want to resolve to the variable value in the Query box.
9. Click Finish.

   The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Create a Static Variable

You create a static variable to make it available for use in policies or responses.

Note: The value of the resolved variable may be no larger than 1K.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.

3. Click Create Variable.

   The Create Variable pane opens.

4. Verify that Create a new object is selected, and click OK.

   Variable settings open.

5. Type the variable name in the Name field.

6. Select Static from the Variable Type list.

   Static variable settings open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

7. Specify the data type and value of the variable in the Variable Information group box.

8. Click Submit.

   The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Create a Request Context Variable

You create a request context variable to make it available for use in policies or responses.

Note: The value of the resolved variable may be no larger than 1K.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.

3. Click Create Variable.

   The Create Variable pane opens.

4. Verify that Create a new object is selected, and click OK.

   Variable settings open.

5. Type the variable name in the Name field.

   Note: Request Context variable names must begin with the percent character (%).

   Example: %REQUEST_ACTION

6. Select Request Context from the Variable Type list.

   Request context settings open.

7. Select the variable value from the Property list.
8. Click OK.

   The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Create a User Context Variable

You create a user context variable to make it available for use in policies or responses.

Note: The value of the resolved variable may be no larger than 1K.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.
3. Click Create Variable.

   The Create Variable pane opens.
4. Verify that Create a new object is selected, and click OK.

   Variable settings open.
5. Type the variable name in the Name field.

   Note: User Context variable names must begin with the percent character (%).

   Example: %SM_USERPATH
6. Select User Context from the Variable Type list.

   User context settings open.
7. Select the portion of the user context that provides the value of the variable from the Property list.

   The return type value appears as either string or boolean depending on the value you selected from the Property list.
8. (Required for User Property and Directory Entry) Enter the name of the directory or user attribute that provides the variable value in the Property field.
9. (Required for User Property and Directory Entry) Enter the size of the buffer (in bytes) that is to store the variable in the Buffer field.
10. (Required for Directory Entry) Enter the distinguished name of the directory entry in the DN field.

11. Click Submit.
12. The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Create a Form Post Variable

You create a Form Post variable to make it available for use in policies or responses.

Note: The value of the resolved variable may be no larger than 1K.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.
3. Click Create Variable.

   The Create Variable pane opens.
4. Verify that Create a new object is selected, and click OK.

   Variable settings open.
5. Type the variable name in the Name field.
6. Select Post from the Variable Type list.

   Form post settings open.
7. Enter the name of the POST variable contained in the form in the Form Field Name field.
8. Click OK.

   The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Configure Message-based Authorization Using an XPath Query in XmlToolkit.properties

You can configure message content-based authorization based on information in incoming XML messages by configuring variables and policy expressions to extract the required information and trigger authorization decisions based on the obtained values.

Alternatively, you can configure an XPath query in the target SOA Agent's XMLToolkit.properties file that extracts a value from each incoming message and incorporates it into your policy's resource value.

Note: To find out the location of the XMLToolkit.properties file for each SOA Agent type, see the *CA SOA Agent Configuration Guide* or the CA SOA Security Gateway documentation.

To configure content-based resource processing

1. Open the XmlToolkit.properties file in a text editor.
2. Make the following changes in XmlToolkit.properties:
   - Add an ResourceXpathQuery parameter whose value is a valid XPath query that identifies the required XML message element, for example:

     ResourceXPathQuery=/SOAP-ENV:Envelope/SOAP-ENV:Header/method

   - (Optional) Add a NodeProperty parameter which specifies a property of an element or attribute to be returned from the XPath query and passed by the SOAP envelope handler to the XPath evaluate method.
3. Save and close the XmlToolkit.properties file.
4. Restart the target SOA Agent.

Notes:

- XPath query processing is namespace aware.
- The XPath query must be rooted at the document root—not at the header or body.
- You can configure only one XPath query per agent instance.
- If the XPath query fails, the target URL will be used as the resource.
- The XPath query is loaded at SOA Agent startup; if it is changed, you must restart the agent.

Chapter 18: Global Policies, Rules, and Responses

This section contains the following topics:

Global Policies (see page 439)
How to Configure Global Policies (see page 443)
Enable and Disable Global Policies (see page 453)
Configure a Global Active Policy (see page 453)
Allowable IP Addresses for Global Policies (see page 454)
Add and Remove Global Policy Time Restrictions (see page 457)

Global Policies

Standard SOA Security Manager policies are created in the context of a single policy domain. However, large production environments may contain thousands of domains. In this type of environment it can be useful to define types of behavior (represented by policies) that are common for many domains. Using standard policies, the same policy must be recreated for each domain that requires the same behavior. Global policies allow you to configure policies (and their associated rules and responses) as system level objects, that are applied across all domains.

The following terms are used for discussing global policies:

Access Rule

An access rule allows or denies access to a resource. Global policies do not include access rules. Only event rules may be added to global policies.

Event Rule

An event rule is invoked when an authentication or authorization event occurs. Behaviors that are commonly implemented across all domains are associated with event rules, and may be included in global policies.

Global Policy

A policy which is defined as a system object.

Global Rule

A rule which is defined as a system object.

Global Response

A response which is defined as a system object.

Policy Link

A logical entity used for policy definition. It consists of a rule- response pair. A policy may contain one or more policy links.

More information:

Policies (see page 385)

Global Policy Object Characteristics

The following sections discuss the characteristics of global policy objects, outlining the basic similarities and differences when compared to their standard (non-global) counterparts.

Global response vs. standard response

Differences:

- Defined at the system level. Only system level administrators can define a global response.
- Cannot use variables-based attributes.
- Used in any global or domain-specific policies.
- Associated with the specific agent type.
- Can be a member of any global or domain-specific response group.

Similarities:

- Can use active expressions.
- Is not returned unless it is specified in a particular policy.

Global rule vs. standard rule

Differences:

- Defined at the system level. Only system level administrators can define a global rule.
- The filter for the global rule is not bound to a specific realm. The filter for the global rule is an absolute filter, which may or may not use a regular expression.
- Bound to specific agent or agent group. The agent is explicitly specified when the rule is created.
- Available only for SOA Security Manager agents. You cannot associate a global rule with a RADIUS Agent because RADIUS Agents do not support authentication and authorization events.
- Only defined for an authentication or authorization event.

- Only used in global policies.
- Cannot be added to rule groups. There are no global rule groups.
- Can fire for resources on any domain for which global policy processing is enabled.

Global policy vs. standard policy

Differences:

- Defined at the system level. Only system level administrators can define a global policy.
- Bound to all users. Specific users cannot be included or excluded from a global policy.

  Note: Individual domains can be explicitly enabled or disabled for global policy processing.
- Defined using only global rules, global responses and groups of these global objects.
- Cannot use variable-based attributes or variable expressions.
- Cannot contain allow/deny access rules. Only event rules may be included in a global policy.
- Cannot include or exclude a particular resource/realm from the global policy. The global policy is applicable to all resources that match at least one of the rule filters defined for the policy, on the domain for which global policy processing is enabled.
- Are not supported through the Java Policy Management API.

Similarities:

- Can use active expressions.
- Associated with the specific Agent. However, it's possible to create a group containing all the Agents of the same type and bind a global rule to such group.

When the global policy is being processed, the responses defined for the fired global rules, are added to the list of other responses. A global rule fires when the following is true:

- The resource being accessed matches the absolute resource filter defined for the global rule.
- The event that occurs is as defined for the global rule.
- The resource being requested is protected by the same agent/agent group, which was specified for the rule.
- The resource/realm being accessed belongs to a domain for which global policies processing is enabled.

More information:

Disable Global Policy Processing for a Domain (see page 327)

Global Policy Concept

SOA Security Manager uses a policy-based access control model. A SOA Security Manager policy defines the type of access a user has to a particular resource and what happens when the user accesses the resource. Each standard SOA Security Manager policy is a linkage between a set of users and a set of resources, and is designed to protect resources by binding together users, rules and responses. Every policy must specify the users or groups of users to which the policy applies. Users can be either included or excluded from the policy.

In addition, a standard policy must contain at least one rule or rule group. Rules are the parts of a policy that determine precisely which resources are protected and what type of action should cause a rule to fire. A rule identifies a resource or resources that are included in the policy using a combination of a string-based *resource filter* and *action*. The filter in turn consists of *realm filter* and *rule filter*.

SOA Security Manager objects can be of two types: system level and domain level. In a standard (non-global) SOA Security Manager policy, all policy objects must be created in the context of a specific domain. However, global policies are system level policies that may be applied across all domains in a SOA Security Manager deployment. An administrator with system level privileges can define global policies, that include global rules and global responses. These global policies may be applied to any resource in any domain.

Global objects are similar to their standard, domain-specific counterparts. The roles of global objects in a global policy definition are different from domain-specific policy objects in the way they are created and linked to form policies. However, there are no global domain or global realm objects.

More information:

Realms Overview (see page 329)
Rules Overview (see page 337)
Rule Group Overview (see page 347)
Responses Overview (see page 351)
Response Groups (see page 380)

Global Policy Processing

Policies are evaluated as described in Policy Processing. In addition, any global rules contained in global policies will fire if the following conditions are met:

- The requested resource belongs to a domain on which global policy processing was not disabled
- The requested resource matches the absolute resource filter defined for the global rule. It is important, that in the case of global rule the filter will be obtained from the rule (not from the realm).
- The event that occurs is the same as that defined for the global rule.
- The requested resource is protected by the same agent or agent group, which was specified for the rule.

Whenever an authentication or an authorization event happens the responses defined for the fired global rules are added to the list of other responses.

How to Configure Global Policies

A global policy is comprised of global rule objects and global response objects, including response attributes. The following process lists the procedures for creating a global policy:

1. Create a Global Rule for Authentication Events (see page 445) or Create a Global Rule for Authorization Events (see page 447)
2. Configure a Global Response (see page 450)
3. Configure a Global Web Agent Response Attribute
4. Configure the Global Policy (see page 451)

Important! You can configure both global policies and domain-specific policies that affect the same resources. For example, you can configure domain-specific policies for access control, and global policies that provide a standard set of responses. However, in order for global policies to function, the realms included in the domain-specific policies must be configured to allow event processing.

Global Rules

Global rules are the part of a global policy that define a resource and events that trigger the processing of a global policy. Global rules are similar to domain-specific rules. However, a global rule must be associated with an authentication or authorization event. There are no global allow/deny access rules.

Global Rules for Authentication Events

Global rules that include SOA Security Manager authentication events let you control actions that occur when users authenticate to gain access to a resource (On-Auth event).

Note: OnAuth event results are per realm, so for example, if a user goes from realm A to realm B and had an OnAuthAccept header in realm A, it will not be available in realm B. When the user goes back to realm A, the header will be set again.

The following is a list of possible On-Auth events:

On-Auth-Accept

Occurs if authentication was successful. This event may be used to redirect a user after a successful authentication.

On-Auth-Reject

Occurs if authentication failed for a user that is bound to a policy containing an On-Auth-Reject rule. This event may be used to redirect the user after a failed authentication.

OnAuthAccept and OnAuthReject events fire both at authentication time (when the user enters his / her username and password) and at validation time (when the user's cookie is read for user information). However, there are certain special actions that only occur at authentication time:

Realm timeout override (unless EnforceRealmTimeouts is used).

Unless you have a version of the Web Agent that supports the EnforceRealmTimeouts option and that option is enabled, the Idle and Max Timeouts for the user will stay at the values for the realm in which the user last authenticated (only changes if the user has to reenter credentials).

Note: More information on EnforceRealmTimeouts exists in section 3.3 of the *SiteMinder 4.x Web and Affiliate Agent Quarterly Maintenance Release 4 Release Notes*.

Redirects.

Redirects are only allowed at authentication time for a number of reasons, but one of the most practical is that it would be very easy to configure an infinite loop of redirection if OnAuth redirection were allowed at validation time as well.

Access to the user's password.

The password is not stored in the SMSESSION cookie, so the only time it is available is when the user actually enters it (authentication time).

On-Auth-Attempt

Occurs if the user was rejected because SOA Security Manager does not know this user (an unregistered user, for example, can be redirected to register first).

On-Auth-Challenge

Occurs when custom challenge-response authentication schemes are activated (for example, a token code).

When a user is authenticated (or rejected), the Policy Server passes any global responses associated with the applicable On-Auth rule back to the requesting Agent.

More information:

Global Response Objects (see page 449)

Create a Global Rule for Authentication Events

You create a global rule for authentication events to control actions that occur when users authenticate to gain access to a resource.

To create a global rule

1. Click Policies, Global.
2. Click Global Rule, Create Global Rule.

   The Create Global Rule pane appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Enter the global rule name.
4. Specify agent and resource settings in the Realm and Resource group box.

   Note: If you specify an Agent Group and have also configured domain-specific rules associated with the same resource, you may adversely affect system performance by effectively duplicating processing steps. Consider domain-specific rules that may duplicate the responses generated by global rules. In such cases, only one response is returned to the Agent because the Policy Server automatically deletes duplicate responses before passing information back to the requesting Agent.
5. Select Authentication events from the Action group box.
6. Select an OnAuth event from the Action List.
7. Click Submit.

   The global rule is saved.

Global Rules for Authorization Events

Global rules that include SOA Security Manager authorization events allow SOA Security Manager to call responses based on whether a user is or is not authorized for the resource the user requested. Authorization events occur after a user is authenticated, if a rule that protects a resource contains an On-Access event. When the user has been granted or denied access based on their privileges, the appropriate event is triggered.

The following is a list of possible On-Access events:

On-Access-Accept

Occurs as the result of successful authorization. This event may be used to redirect users who are authorized to access a resource.

On-Access-Reject

Occurs as the result of failed authorization. This event may be used to redirect users who are not authorized to access a resource.

When a user is authorized (or rejected), the Policy Server passes any responses associated with the applicable On-Access rule back to the requesting Agent.

Policy Considerations for OnAccessReject Rules

Consider how the Policy Server processes global policies and the special circumstances created by OnAccessReject rules when creating global rules that include OnAccessReject events.

An OnAccessReject rule will not fire if it is in the same policy as a GET / POST rule. When a user is authenticated, SOA Security Manager resolves the identity of the user. Therefore, if the OnAccessReject rule and the GET / POST rule are in the same policy, then a user who is allowed access to a resource is the same user who should be redirected on an OnAccessReject event. Since the user is allowed access, the reject event never applies.

To resolve this discrepancy, create a separate policy for the OnAccessReject rule, which may include other event rules, and specify the users for which it should apply.

For example, in an LDAP user directory, User1 should have access to a resource and everyone else in the group, ou=People, o=company.com, should be redirected to an OnAccessReject page. Two policies are required:

Policy1

Includes a GET / POST rule that allows access for User1.

Policy2

Includes the OnAccessReject rule and a Redirect response, and specifies the group ou=People, o=company.com.

Since User1 is authorized, the OnAccessReject rule will not fire when User1 access the resource. However, the OnAccessReject rule will fire for all other users in the group, ou=People, o=company.com, because they are not authorized to access the resource.

Create a Global Rule for Authorization Events

You create a global rule for authentication events to control actions that occur when users authenticate to gain access to a resource.

To create a global rule

1. Click Policies, Global.
2. Click Global Rule, Create Global Rule.

   The Create Global Rule pane appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Enter the global rule name.
4. Specify agent and resource settings in the Realm and Resource group box.

   Note: If you specify an Agent Group and have also configured domain-specific rules associated with the same resource, you may adversely affect system performance by effectively duplicating processing steps. Consider domain-specific rules that may duplicate the responses generated by global rules. In such cases, only one response is returned to the Agent because the Policy Server automatically deletes duplicate responses before passing information back to the requesting Agent.
5. Select Authentication events from the Action group box.
6. Select an OnAuth event from the Action List.
7. Click Submit.

   The global rule is saved.

More information:

Responses and Response Groups (see page 351)
Start the Administrative UI (see page 42)
Resource Matching and Regular Expressions (see page 342)

Enable and Disable Global Rules

You enable a global rule to ensure SOA Security Manager fires the rule if a user accesses the specified resource and triggers the authentication or authorization event. You disable a global rule to prevent SOA Security Manager from firing the rule if a user accesses the specified resource and triggers the authentication or authorization event.

To enable or disable a global rule

1. Open the global rule.
2. Select the Enabled check box to enable the rule; clear the Enabled check box to disable the rule.
3. Click Submit.

   The rule is saved.

More information:

Start the Administrative UI (see page 42)

Add Time Restrictions to Global Rules

You add time restrictions to a global policy to ensure that the global policy only fires at specific times. If a user attempts to access a resource outside of the period specified by the time restriction, the policy does not fire.

To add a time restriction to a global rule

1. Open the global policy.
2. Click Set in the Time Restrictions group box.

   The Time Restrictions pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Specify starting and expiration dates.
4. Specify time restrictions in the Hourly Restrictions table.

   Note: Each check box represents one hour. When a check box is selected, the rule fires during that hour, and the rule applies to the specified resources. When a check box is cleared, the rule does not fire during that hour, and the rule will not apply to the specified resources.
5. Click OK.
6. The time restrictions are saved.

Configure an Active Global Rule

You configure an active rule for dynamic authorization based on external business logic. The Policy Server invokes a function in a customer-supplied shared library. This shared library must conform to the interface specified by the Authorization API, which is available separately via the Software Development Kit

Note: More information on shared libraries exists in the *API Reference Guide for C*.

To configure an Active Rule

1. Select the Edit Active Rule check box in the Active Rule group box.

   Active rule fields appear.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.

2. Specify the library name, function name, and function parameters in the respective fields.

   The active rule string appears in the Active Rule field.

3. Click Submit.

   The active rule is saved.

Delete a Global Rule

If you delete a global rule, the rule is automatically removed from any global policies that include the global rule. The global policies remain on your system. Verify that the global policies function without the deleted rule.

Global policies must contain at least one global rule.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 42).

Global Response Objects

Global responses are the part of a global policy that define the attributes to be returned after a user triggers the authentication or authorization event specified in a global rule.

Note: You may use global responses in domain policies. In order to be returned, a global response must be added to a domain-specific or global policy. Within policies, the global response will be processed like a domain-specific response.

Configure a Global Response

You configure a global response to define the attributes that are returned after the authentication or authorization event occurs in an associated global rule.

To configure a global response

1. Click Policies, Global.
2. Click Global Response, Create Global Response.

   The Create Global Response pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Enter the global response name.
4. Select a SOA Security Manager Agent type from the SOA Security Manager Agent type list.
5. Click Submit.

   The global response is saved. You can now add response attributes to the response. See the next section for details about adding response attributes for each Agent type.

More information:

Start the Administrative UI (see page 42)

Response Attributes for Global Responses

Each SOA Security Manager global response may contain one or more response attributes. Response attributes identify the pieces of information that the Policy Server passes to a SOA Security Manager Agent. Each SOA Security Manager Agent type can accept different response attributes.

Global Response Attribute Types

SOA Security Manager supports different types of response attributes. The types of response attributes determine where the Policy Server finds the proper values for the response attributes. The types of response attributes that you can configure for a global response are identical to the types of response attributes you can configure for a domain-specific response.

Configure a Global Web Agent Response Attribute

You can configure a response attribute to store the pieces of information that the Policy Server passes to a SOA Agent. SOA Agent response attributes support HTTP header variables, cookie variables, redirections to other resources, text, and timeout values.

You create a response attribute for a SOA Agent by selecting SiteMinder and Web Agent on the Attributes group box on the Response pane.

To create a response attribute

1. Click Create Response Attribute on the Attribute List group box on the Response pane.

   The Create Response Attribute pane opens.

2. Select a response attribute from the Attribute drop-down list.

3. Select an attribute type on the Attribute Kind (one of Static, User Attribute, DN Attribute, and Active Response) group box.

   The fields on the Attribute Fields group box are updated to match the specified attribute type.

4. Complete the fields on the Attribute Fields group box.

   Note: For WebAgent-SAML-Session-Ticket-Variable and WebAgent-WS-Security-Token attributes, you can either enter values directly in the Variable Name and Variable Value fields or populate those fields with valid values from the Select a Name and Select a Value lists that appear.

5. Specify Cache Value or Recalculate value every ... seconds on the Attribute Caching group box.

6. Click Submit.

   The Create Response Attribute Task is submitted for processing, and the response attribute is added to the Attribute List on the Response pane.

How to Configure Global Policy Objects

Configuring a global policy requires you to complete the following procedures:

1. Create the Global Policy (see page 452)
2. Add Global Rules to the Global Policy (see page 452)
3. (Optional) Associate a Global Rule with a Response (see page 452)

More information:

Start the Administrative UI (see page 42)

Create the Global Policy

You create a global policy to define how users interact with resources.

To create a global policy

1. Click Policies, Global.
2. Click Global Policy, Create Global Policy.

   The Create Global Policy pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Enter the global policy name.
4. Add global rules and global responses.

Add Global Rules to a Global Policy

Global rules indicate the specific resources included in a global policy. You must add at least one global rule to a global policy.

To add global rules to a global policy

1. Click the Rules tab.

   The Rules group box opens.
2. Click Add Rule.

   The Available Rules pane opens and lists the available global rules.

   Note: If the global rule you require does not appear, click New Rule. Rules you create in this manner are added to the global policy.
3. Select the global rules you want to add, and click OK.

   The Rules group box lists the selected rules and rule groups.
4. (Optional) Associate the rule with a response or response group.

Associate a Global Rule with a Response

Global responses indicate the actions that should take place when the rule fires. When the rule fires, the associated response also fires.

To associate a response with a global rule

1. Click Add Response for the global rule for which you want to associate a response.

   The Available Responses pane opens and lists the available responses, response groups, and global responses.

2. Select a response, response group, or global response, and click OK.

   The response opens in the Rules group box, and is associated with the respective rule.

   Note: If the response you require does not exist, click New Response to create the response.

Enable and Disable Global Policies

The Administrative UI allows you to enable and disable global policies. By default, when you create a global policy, the policy is enabled. When a global policy is enabled, global rules contained in the global policy fire when users attempt to access the resources specified in the global rules.

If you disable a global policy, the rules contained in the policy do not fire.

To enable or disable a policy

1. Open the policy.
2. Select or clear the Enabled check box.

   If the check box is selected, the policy is enabled. If the check box is cleared, the policy is disabled. A disabled policy does not fire.

3. Click Submit.

   The policy is saved.

More information:

Start the Administrative UI (see page 42)

Configure a Global Active Policy

An active policy is used for dynamic authorization based on external business logic. An active policy is included in the authorization decision by having the Policy Server invoke a function in a customer-supplied shared library.

This shared library must conform to the interface specified by the Authorization API (available separately with the Software Development Kit.

Note: More information exists in API Reference Guide for C.

The process for configuring active policies for global policies is identical to the process for configuring active policies for domain-specific policies.

To configure an Active Policy

1. Open the global policy.
2. Select the Edit Active Policy check box in the Advanced Group box.

   Active policy settings appear.
3. Enter the name of the shared library in the Library Name field.
4. Enter the name of the function in the shared library that is to implement the active policy.
5. Click Submit.

   The policy is saved.

More information:

Configure an Active Policy (see page 404)

Allowable IP Addresses for Global Policies

You specify that a global policy should only fire for users who access the policy's resources from a specific:

- IP address
- host name
- subnet mask
- range of IP addresses

For example, if you include a rule that allows access to resources in the policy, and then you specify a range of IP addresses, only those users who login in from one of the specified IP addresses will be allowed access to the protected resources.

More information:

Allowable IP Addresses for Policies (see page 400)

Specify a Single IP Address for a Global Policy

You specify a single IP address to ensure that the global policy only fires for users who access the policy's resources from the specified IP address.

To specify single IP address

1. Open the policy.
2. Click Add in the IP Address group box.

   Settings for IP addresses appear.
3. Select the Single Host radio button.

   Settings specific to a single host appear.
4. Enter the IP Address, and click OK.

   The IP address appears in the IP Address group box.

   Note: If you do not know the IP address, but have the domain name for the address, click DNS Lookup. Enter a fully qualified host name to look up the IP address.
5. Click Submit.

   The policy is saved.

Add a Host Name for a Global Policy

You specify a host name to ensure the global policy only fires for users who access the policy's resources from the specified host.

To specify a host name

1. Open the policy.
2. Click Add in the IP Address group box.

   Settings for IP Addresses appear.
3. Select the Host Name radio button.

   Settings specific to a host name appear.
4. Enter the host name, and Click OK

   The host name appears in the IP Address group box.
5. Click Submit.

   The policy is saved.

Add a Subnet Mask for a Global Policy

You specify a subnet mask to ensure the global policy only fires for users who access the policy's resources from the specified subnet mask.

To add a subnet mask

1. Open the policy.
2. Click Add in the IP Address group box.

   Settings for IP Addresses appear.
3. Select the Subnet Mask radio button.

   Settings specific to the subnet mask appear.
4. Enter an IP address in the IP Address field.

   Note: If you do not know the IP address, but have the domain name for the address, click DNS Lookup. Enter a fully qualified host name to look up the IP address.
5. Enter a subnet mask in the Subnet Mask field.
6. Click OK.

   The subnet mask appears in the IP Address group box.
7. Click Submit.

   The policy is saved.

Add a Range of IP Addresses for a Global Policy

You specify a range of IP addresses to ensure that the policy only fires for users who access the policy's resources from one of the IP addresses included in the range of addresses.

To add a range of IP addresses

1. Open the policy
2. Click Add in the IP Address group box.

   Settings IP Addresses appear.
3. Select the Range radio button.

   Settings specific to a range of IP addresses appear.
4. Enter a starting IP Address in the From field.

   Note: If you do not know an IP address, but have the domain name for the address, click DNS Lookup. Enter a fully qualified host name to look up the IP address.

5. Enter an ending IP address in the To field.
6. Click OK.

   The range of IP addresses appears in the IP Address group box.
7. Click Submit.

   The policy is saved.

Add and Remove Global Policy Time Restrictions

You can add time restrictions to a global policy. When you add a time restriction, the global policy only fires during the period specified in the time restriction. If a user attempts to access a resource associated with the policy outside of the period specified by the time restriction, the global policy does not fire.

Note: Time restrictions are based on the system clock of the server on which the Policy Server is installed.

To add a time restriction to a policy

1. Open the policy.
2. Click Set in the Time group box.

   The Time Restrictions pane appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Specify starting and expiration dates.
4. Specify time restrictions in the Hourly Restrictions table.

   Note: Each check box represents one hour. When a check box is selected, the rule fires during that hour, and the rule applies to the specified resources. When a check box is cleared, the rule does not fire during that hour, and the rule will not apply to the specified resources.
5. Click OK.

   The time restrictions are saved.

More information:

Time Restrictions for Policies (see page 403)

Appendix A: LanMan User Directories

This section contains the following topics:

About LanMan User Directories (see page 459)
LanMan Directory Connection Prerequisites (see page 459)
Configure a LanMan Directory Connection (see page 460)
Failover for Windows User Directories (see page 462)
LanMan User Directory Search Criteria (see page 462)

About LanMan User Directories

In a Windows environment, the Policy Server enumerates and manages the resources in a directory service through the Microsoft Active Directory Service Interface (ADSI) layer. This layer abstracts the capabilities of directory services from different network providers in a distributed computing environment. However, the current version of ADSI has its own limitations which can adversely affect the performance of the Policy Server.

With ADSI, every Windows directory request must always pass through the Primary Domain Controller (PDC) first. This compounds the network traffic that the PDC must handle. A custom solution to this dilemma is for the Policy Server to channel Windows directory requests to Backup Domain Controllers (BDCs) while bypassing the PDC. The Policy Server handles this sort of custom solution by using LanMan directory connections.

The LanMan user directory connection option allows you to specify a failover list of BDCs used for each user directory lookup in the Windows Registry. Using a LanMan directory connection, the Policy Server sends Windows directory requests to the first active BDC in the Registry list, rather than forcing requests to pass through the PDC.

LanMan Directory Connection Prerequisites

The following conditions must be met before the Policy Server can use a LanMan directory connection to access user data in a Windows directory:

- The file SmDsLanman.dll must reside in the Policy Server installation directory. The default location is:

  SOA_HOME\siteminder\bin\

- You must configure a connection to the LanMan directory.

Configure a LanMan Directory Connection

You can configure a LanMan user directory. The following process lists the steps for creating a user directory connection to the Policy Server.

1. Configure Registry Keys for a LanMan Directory Connection (see page 460)
2. Configure a LanMan User Directory Connection (see page 461)

Configure Registry Keys for a LanMan Directory Connection

The first procedure in configuring a LanMan directory connection is configuring the appropriate registry keys.

To configure registry keys for a LanMan directory connection

1. Select Run from the Windows Start menu.

   The Run dialog opens.

2. Enter **regedit,** and click OK.

   The Registry Editor opens.

3. Modify the following registry key:

   - In HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\Siteminder\CurrentVersion\Ds, the NameSpaces key is set to the following string value:

     "LDAP:,ODBC:,OCI:,WinNT:,Custom:,AD:"

   - Add the following string to the value data for the NameSpaces registry key: Lanman.

4. Create the following registry key:

   \HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Ds\Lanman_DC

5. Create a registry key of the NT Domain Name under the Lanman_DC key:

   \HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Ds\Lanman_DC\<*NT_domain_name*>

   For example:

   \HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Ds\Lanman_DC\MyDomain

6. Create a registry value named NumUserDir of type DWORD under the newly created NT Domain key. For the value data, enter the actual number of separate sets of user directories (maximum 16) in this NT domain.

7. Create String registry values of UserDir0, UserDir1, …, UserDirN, in sequential order starting from 0, for each failover list of BDCs.

8. Enter comma delimited strings for each failover list. SmDsLanman will read the lists and will find the first active BDC in each failover list to look up NT users and groups.
9. Repeat steps 5 through 7 for other NT domains.
10. Restart the Policy Server services.

    Note: More information on starting and stopping the Policy Server exists in the *Policy Server Management* guide.

Configure a LanMan User Directory Connection

You can configure a user directory connection that lets the Policy Server communicate with a LanMan Directory user store.

Note: The following procedure assumes you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure a LanMan user directory connection

1. Click Infrastructure, Directory.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create User Directory: *Name* pane opens.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Type the name and a description of the user directory in the fields on the General group box.
5. Select LanMan from the Namespace list.

   LanMan settings open.
6. Type the name of the NT Domain that you configured in the registry keys in the Domain Controller Key field.
7. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

User Directories (see page 85)
Configure Registry Keys for a LanMan Directory Connection (see page 460)

Failover for Windows User Directories

The list of registry keys you create for the LanMan user directory connection determines failover order.

LanMan User Directory Search Criteria

LanMan directory connections are a type of Windows user directory connection. A LanMan directory connection functions similarly to a regular Windows connection, with the exception of which actual Domain Controller handles requests. This does not affect the procedure for executing a user directory search.

More information:

Search User Directories (see page 167)

Appendix B: Attributes and Expressions Reference

This section contains the following topics:

Data Types (see page 463)
Expression Syntax Overview (see page 466)
Pasting (see page 467)
Operators (see page 469)
Functions Available within Expressions (see page 486)

Data Types

All constant data is one of three main literal data types: strings, numbers, or Booleans. The string data type includes two sub-types: sets and LDAP distinguished names. The number data type includes one sub-type: dates. All functions and operations result in one of these types or their sub-types.

- Strings
  - Sets
  - LDAP Distinguished Names
- Numbers
  - Dates
- Booleans

In addition to the literal data types, there are data types that function as variables:

- User Attributes
- Named Expressions

Strings

Strings represent character data as a string of zero or more characters enclosed by a pair of single or double quotes. String values are constants that can be manipulated by the built-in operators and functions. For example, strings can be converted to another data type or concatenated.

Strings that start with an optional positive or negative sign and contain the characters "0" through "9" can be converted to number values. The strings "TRUE" and "YES" (or "true" and "yes") can be converted to the Boolean value TRUE. All other string values are converted to FALSE. Two strings can be concatenated. In concatenation, the beginning of the second string is joined to the end of the first string.

Sets and LDAP distinguished names (DNs) are special cases of the string data type. A set is a string of elements that are separated by the caret character, for example, 'element1^element2'. Each element in the set is a string.

An LDAP DN is a simple string that uniquely identifies an entry in an LDAP directory and whose format is defined by the LDAP specification.

Numbers

Numbers must be integers or whole numbers with an optional leading positive or negative sign. Four-byte integers are supported or whole numbers that range from -2^{31} to 2^{31}. Decimal points are ignored. Number values are constants that can be manipulated by the built-in operators and functions. For example, numbers can be converted to another data type.

Numbers can be converted to strings that start with an optional negative sign and contain only the characters "0" through "9". Numbers can also be converted to Boolean values. Non-zero numbers are converted to TRUE; zero is converted to FALSE.

Dates are a special case of the number data type. They are represented as the number of seconds that have passed since January 1, 1970.

There are numerous functions that manipulate dates. For example, the DOW function accepts the numeric representation of a date and returns a number in the range 0-6 that corresponds to a day of the week. There are also functions that convert a date in string format to a number and a number representation of a date to a string.

Booleans

Booleans are one of two values: TRUE or FALSE. Boolean values are constants that can be compared and manipulated by the built-in operators and functions. For example, Booleans can be converted to another data type.

When a Boolean value is converted to a number value, TRUE is converted to 1, and FALSE is converted to 0. When a Boolean value is converted to a string value, TRUE is converted to "TRUE" and FALSE is converted to "FALSE".

User Attributes

User attributes have names and values. User attribute *names* are *unquoted*. User attribute *values* are strings and are *quoted*. User attribute names must conform to the following rules:

- They are not case-sensitive.
- The first character must be a letter.
- They contain only US-ASCII alphabetic characters, numeric characters, and the underscore character.

A user attribute name functions as a variable data type, not as a literal. When a user attribute name is encountered, the corresponding attribute value is retrieved from the user directory. When an attribute has multiple values, they are returned as a set. A set is a string of elements separated by the caret character, for example, 'value1^value2'. Each element in the set is a string.

Named Expressions

There are two types of named expressions: virtual user attributes and user classes.

Virtual user attributes differ from user attributes. Unlike user attributes, which are stored in the user directory as strings, virtual user attributes name expressions that are calculated at runtime and that result in a string, number, or Boolean value. Also unlike user attributes, virtual user attributes are read-only.

User classes are a special case of virtual user attributes. Like virtual user attributes, user classes name expressions that are calculated at runtime. Unlike virtual user attributes, user classes name expressions that test membership in a user group or directory and that result in a Boolean value only.

Both virtual user attribute names and user class names must conform to the following rules:

- They are not case-sensitive.
- In the case of virtual user attributes, the first character must be the pound sign (#).

- In the case of user classes, the first character must be the at sign (@).
- The second character must be a letter or an underscore.
- The remaining characters must be US-ASCII alphabetic characters, numeric characters, or the underscore character.

Note: Active expressions and named expressions are not the same. While both types of expressions are evaluated at run-time, they differ in the following ways:

- While active expressions are Boolean expressions, named expressions can return a string, number, or Boolean value.
- While active expressions are referenced as is and must be reentered each time that they are used, named expressions are referenced by name and can be referenced from anywhere and reused.

Expression Syntax Overview

The syntax described in this appendix belongs to an internal SOA Security Manager expression evaluator. You can use the data types, operators, and built-in functions that comprise this syntax in expressions, when you define Roles or Entitlements in the Administrative UI. An unnamed expression is local to the particular Role or Entitlement that you are defining, or you can use named expressions, which are defined globally.

Named expressions include virtual user attributes (whose names begin with #) and user classes (whose names begin with @).

A virtual user attribute calculates a value when the required information cannot be read directly from a user's directory entry. Virtual user attributes return a string, number, or Boolean value. For example, if you wanted to format name information so that it could be used frequently for sorting, you could define a virtual user attribute called #SortName in the user interface as follows:

```
UCase(RTrim(LastName + "," + FirstName + " " + Initial))
```

This example uses two built-in functions, UCASE and RTRIM. Note that these name are not case sensitive.

A user class is an expression that determines whether a user belongs to a particular category based on user type, such as a manager or an administrator. A user is either a member of a particular user class or not, so the result of a user class expression is always Boolean.

When you define a virtual user attribute or user class, you can specify that it is private, which means that it can only be called from other named expressions. Similarly, some of the built-in functions are designated as privileged functions, which means that they can only be called from within another named expression. Privileged functions are noted in the Remarks section as "Privileged". Functions that accept one or more LDAP Distinguished Names as parameters are noted in the Remarks section as "LDAP Only".

Pasting

Expressions can be stored as objects in the Policy Store, where they can be referenced by name from anywhere, including from other expressions. Any expression can pass values to a named expression through the placeholders %1 through %9 and the special placeholder %0. This is called *pasting*.

There are two types of named expressions: virtual user attributes and user classes. Virtual user attribute names start with a pound sign, and user class names start with an at sign. Both types of named expressions are followed by up to nine parameters. The syntax is similar to the syntax of a function:

```
#virtual_user_attribute(P1, P2, P3, P4, P5, P6, P7, P8, P9)

@user_class(P1, P2, P3, P4, P5, P6, P7, P8, P9)
```

When creating a named expression in the Policy Store, you can use built-in operators and functions, the literal data types, the placeholders, and other named expressions. Use the placeholders to pass variable data to the named expression. In the following example, the URL is updated during each iteration and thus, must be represented by the placeholder %1.

Example:

You can create a virtual attribute named #URLFile that accepts a URL and returns a filename:

```
#URLFile := { FIND(%1, '/')=0 ? %1 : #URLFile(AFTER(%1, '/')) }

Return_value=#URLFile('C:\My Documents\expression_syntax.doc')
Return_value='expression_syntax.doc'
```

In this example, the URL is passed to the built-in function FIND through the placeholder %1. FIND finds the first instance of "/" in the URL and returns its position. If "/" is not found, FIND returns 0 and #URLFile returns the filename. Otherwise, the URL is passed to the built-in function AFTER through the placeholder %1. AFTER returns that part of the URL that follows "/". The shortened URL is then passed to #URLFile. Recursion is supported.

The following table shows the values of the position and the URL at the completion of each iteration in this example:

Iteration	Position	URL
1	3	'My Documents\expression_syntax.doc'
2	16	'expression_syntax.doc'

When certain built-in functions, such as ENUMERATE or LOOP, call a named expression multiple times, once for each element in a set, the named expression must be created using the special placeholder %0. For example, you can create an expression named #RTrimset that removes trailing spaces from any number of set elements:

```
#RTrimset := RTrim(%0)
```

Then, you can pass a set to #RTrimset through the built-in function ENUMERATE:

```
Return_value=ENUMERATE('First_name    ^Middle_name    ^Last_name    ',#RTrimset)
Return_value='First_name^Middle_name^Last_name'
```

In this example, the set consists of three elements: the first, middle, and last names. ENUMERATE passes each name to #RTrimset. #RTrimset removes the trailing spaces and returns the shortened name to ENUMERATE. ENUMERATE includes each returned name in the resulting string and uses the caret character to separate them.

More information:

ENUMERATE Function--Test Set Elements (see page 509)
LOOP Function--Call a Virtual Attribute in a Loop (see page 525)

Operators

This topic lists all supported operators by category.

Comparative Operators

- Equality (= and ~=)
- Inequality (!= and ~!=)
- Greater-than (> and ~>)
- Less-than (< and ~<)
- Greater-than or Equal-to (>= and ~>=)
- Less-than or Equal-to (<= and ~<=)

String Operators

- BEGINS_WITH and ~BEGINS_WITH
- ENDS_WITH and ~ENDS_WITH
- CONTAINS and ~CONTAINS
- Pattern Matching (LIKE)
- Concatenation (+)

Set Operators

- Set Inclusion (IN and ~IN)
- INTERSECT and ~INTERSECT
- UNION and ~UNION
- Indexing ([..])

Logical Operators

- AND, &, and &&
- NOT
- OR, |, and ||
- XOR

Arithmetic Operators

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)

Miscellaneous Operator

- Conditional Decision (? and :)

Equality Operators

The equality operator (=) compares two values. If the values are equal, the result of the operation is TRUE. Otherwise, the result of the operation is FALSE. If the two values are strings, the operation is case-sensitive.

The equality operator (~=) only compares string values and is not case-sensitive.

Examples:

1 = 1
Result = TRUE

1 = 2
Result = FALSE

"sparrow" = "SPARROW"
Result = FALSE

"sparrow" ~= "SPARROW"
Result = TRUE

More information:

Inequality Operators (see page 470)

Inequality Operators

The inequality operator (!=) compares two values. If the values are not equal, the result of the operation is TRUE. Otherwise, the result of the operation is FALSE. If the two values are strings, the operation is case-sensitive.

The inequality operator (~!=) only compares string values and is not case-sensitive.

Examples:

1 != 1
Result = FALSE

1 != 2
Result = TRUE

"sparrow" != "SPARROW"
Result = TRUE

"sparrow" ~!= "SPARROW"
Result = FALSE

More information:

Equality Operators (see page 470)

Less-than Operators

The less-than operator (<) compares two values. If the first value is less than the second value, the result of the operation is TRUE. Otherwise, the result of the operation is FALSE. If the two values are Boolean, TRUE is greater than FALSE. If the two values are strings, the operation is case-sensitive.

The less-than operator (~<) only compares string values and is not case-sensitive.

Examples:

1 < 2
Result = TRUE

2 < 1
Result = FALSE

'crow' < 'CROW'
Result = FALSE

'crow' ~< 'CROW'
Result = FALSE

More information:

Greater-than Operators (see page 472)

Greater-than Operators

The greater-than operator (>) compares two values. If the first value is greater than the second value, the result of the operation is TRUE. Otherwise, the result of the operation is FALSE. If the two values are Boolean, TRUE is greater than FALSE. If the two values are strings, the operation is case-sensitive.

The greater-than operator (~>) only compares string values and is not case-sensitive.

Examples:

1 > 2
Result = FALSE

2 > 1
Result = TRUE

'crow' > 'CROW'
Result = TRUE

'crow' ~> 'CROW'
Result = FALSE

More information:

Less-than Operators (see page 471)

Less-than or Equal-to Operators

The less-than or equal-to operator (<=) compares two values. If the first value is less than or equal to the second value, the result of the operation is TRUE. Otherwise, the result of the operation is FALSE. If the two values are Boolean, TRUE is greater than FALSE. If the two values are strings, the operation is case-sensitive.

The less-than or equal-to operator (~<=) only compares string values and is not case-sensitive.

Examples:

1 <= 2
Result = TRUE

2 <= 1
Result = FALSE

'junco' <= 'JUNCO'
Result = FALSE

'junco' ~<= 'JUNCO'
Result = TRUE

More information:

Greater-than or Equal-to Operators (see page 473)

Greater-than or Equal-to Operators

The greater-than or equal-to operator (>=) compares two values. If the first value is greater than or equal to the second value, the result of the operation is TRUE. Otherwise, the result of the operation is FALSE. If the two values are Boolean, TRUE is greater than FALSE. If the two values are strings, the operation is case-sensitive.

The greater-than or equal-to operator (~>=) only compares string values and is not case-sensitive.

Examples:

1 >= 2
Result = FALSE

2 >= 1
Result = TRUE

'junco' >= 'JUNCO'
Result = TRUE

'junco' ~>= 'JUNCO'
Result = TRUE

More information:

Less-than or Equal-to Operators (see page 472)

Begins-with Operators

The two begins-with operators (BEGINS_WITH and ~BEGINS_WITH) are designed to be used with string values. If the first string begins with the second string, the result of the operation is TRUE. Otherwise, the result of the operation is FALSE.

The "BEGINS_WITH" operator is case-sensitive. The "~BEGINS_WITH" operator is not case-sensitive.

Examples:

'SiteMinder' BEGINS_WITH 'site'
Result = FALSE

'SiteMinder' ~BEGINS_WITH 'site'
Result = TRUE

More information:

Ends-with Operators (see page 474)
Containment Operators (see page 475)

Ends-with Operators

The two ends-with operators (ENDS_WITH and ~ENDS_WITH) are designed to be used with string values. If the first string ends with the second string, the result of the operation is TRUE. Otherwise, the result of the operation is FALSE.

The "ENDS_WITH" operator is case-sensitive. The "~ENDS_WITH" operator is not case-sensitive.

Examples:

'SiteMinder' ENDS_WITH 'DER'
Result = FALSE

'SiteMinder' ~ENDS_WITH 'DER'
Result = TRUE

More information:

Begins-with Operators (see page 474)
Containment Operators (see page 475)

Containment Operators

The two containment operators (CONTAINS and ~CONTAINS) are designed to be used with string values. If the first string contains the second string, the result of the operation is TRUE. Otherwise, the result of the operation is FALSE.

The "CONTAINS" operator is case-sensitive. The "~CONTAINS" operator is not case-sensitive.

Examples:

'SiteMinder' CONTAINS 'EMI'
Result = FALSE

'SiteMinder' ~CONTAINS 'EMI'
Result = TRUE

More information:

Begins-with Operators (see page 474)
Ends-with Operators (see page 474)

Set Inclusion Operators

The set inclusion operators (IN and ~IN) test whether the first operand, a string, is an element of the second operand, a set. If the string is an element of the set, the result of the operation is TRUE. Otherwise, the result of the operation is FALSE.

The IN operator is case-sensitive. The ~IN operator is not case-sensitive.

Examples:

'MON' IN 'Sun^Mon^Tue^Wed^Thu^Fri^Sat'
Result = FALSE

'MON' ~IN 'Sun^Mon^Tue^Wed^Thu^Fri^Sat'
Result = TRUE

More information:

Set Intersection Operators (see page 480)
Set Union Operators (see page 480)

Pattern Matching Operator

The pattern matching operator LIKE compares a string value to a pattern of characters that is also a string, for example, 'abc' LIKE '???'. If the string value matches the pattern, the result of the operation is TRUE. Otherwise, the result of the operation is FALSE. The pattern matching operation is case-sensitive.

Patterns are created by combining single characters, character ranges, or both. Character ranges are specified by concatenating the first character in the range, a hyphen, and the last character in the range, for example, 0-9. Valid characters include numbers, uppercase and lowercase letters, and reserved characters that have special meanings. Character sets contain one or more characters, character ranges, or both and are enclosed by square brackets. For example, [0-9A-Za-z], [0-2ABC], and [789x-z] are all valid character sets.

Note: Character ranges are always part of a character set.

The following table lists the reserved characters and their meanings:

Character	Meaning
' or "	Specifies a string, such as 'abc' or "abc"
-	Specifies a range of characters, such as 0-9
?	Matches any single character
*	Matches any sequence of characters, including one or none
[set]	Matches any single character in the specified set
[!set] or [^set]	Matches any single character *not* in the specified set
[set]?	Matches any single character in the specified set or an empty string
[set]*	Matches any sequence of characters in the specified set, including one or none
\	Treats any reserved character as a regular character, such as *

Examples that use '?'

'' LIKE '?'
Result = FALSE

'a' LIKE '?'
Result = TRUE

'ab' LIKE '?'
Result = FALSE

'abc' LIKE '???'
Result = TRUE

'191' LIKE '1??'
Result = TRUE

'201' LIKE '1??'
Result = FALSE

Examples that use '*'

'' LIKE '*'
Result = TRUE

'a' LIKE '*'
Result = TRUE

'a1b2c3' LIKE '*'
Result = TRUE

'robin' LIKE 'r*n'
Result = TRUE

'room' LIKE 'r*n'
Result = FALSE

Examples that use '[set]'

'' LIKE '[abcde]'
Result = FALSE

'f' LIKE '[abcde]'
Result = FALSE

'c' LIKE '[abcde]'
Result = TRUE

'abc' LIKE '[abcde]' **Compare:** 'abc' LIKE '[abcde]*'
Result = FALSE

Examples that use '[!set]' or '[^set]'

'' LIKE '[!abcde]'
Result = FALSE

'f' LIKE '[^abcde]'
Result = TRUE

'c' LIKE '[!abcde]'
Result = FALSE

'xyz' LIKE '[^abcde]'
Result = FALSE

Examples that use '[set]?'

'' LIKE '[abcde]?'
Result = TRUE

'a' LIKE '[abcde]?'
Result = TRUE

'ab' LIKE '[abcde]?'
Result = FALSE

'z' LIKE '[abcde]?'
Result = FALSE

Examples that use '[set]*'

'' LIKE '[abcde]*'
Result = TRUE

'a' LIKE '[abcde]*'
Result = TRUE

'aabbccddee' LIKE '[abcde]*'
Result = TRUE

'abcdef' LIKE '[abcde]*'
Result = FALSE

'abc' LIKE '[abcde]*' **Compare:** 'abc' LIKE '[abcde]'
Result = TRUE

Examples that use '\'

'123-456-7890' LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
Result = TRUE

'_!_^_*_?_' LIKE '_\!_\^_*_\?_'
Result = TRUE

Examples that test case-sensitivity

'a' LIKE '[a-z]'
Result = TRUE

'A' LIKE '[a-z]'
Result = FALSE

'A' LIKE '[A-Za-z]'
Result = TRUE

'Robin' LIKE '[A-Za-z]*'
Result = TRUE

'Robin' LIKE '[A-Z][a-z]*'
Result = TRUE

'robin' LIKE '[A-Z][a-z]*'
Result = FALSE

Set Intersection Operators

The set intersection operators (INTERSECT and ~INTERSECT) compare two sets. Sets are strings of elements separated by the caret character. The resulting set is a string that contains only those elements present in both sets. The INTERSECT operator is case-sensitive. The ~INTERSECT operator is not case-sensitive.

Important! The sequence of elements in the resulting set is not predictable. When the operation is not case-sensitive, the case of elements in the resulting set is also not predictable.

Examples:

'BLUE JAY^ORIOLE^WREN' INTERSECT 'BLUE JAY^wren'
Result = 'BLUE JAY'

'BLUE JAY^ORIOLE^WREN' ~INTERSECT 'BLUE JAY^wren'
Result = 'BLUE JAY^WREN'

More information:

Set Inclusion Operators (see page 475)
Set Union Operators (see page 480)

Set Union Operators

The set union operator (UNION) returns a set containing all unique elements of the two operand sets (the union of the two sets). Duplicate elements are removed.

If the tilde character (~) precedes the UNION operator (~UNION), then two elements that differ only in case are considered identical.

Important! The sequence of the resulting set is not predictable. Also, if the ~UNION operator is specified, the case of a resulting intersecting element is not predictable.

Examples:

"JUAN^BART^CHUCK" UNION "CHUCK^Bart"
Result = "JUAN^BART^CHUCK^Bart"

"JUAN^BART^CHUCK" ~UNION "CHUCK^Bart"
Result = "JUAN^BART^CHUCK"

"JUAN^BART^JUAN^CHUCK" UNION "JUAN^BART^JUAN^CHUCK"
Result = "JUAN^BART^CHUCK"

More information:

Set Inclusion Operators (see page 475)
Set Intersection Operators (see page 480)

NOT Operator

The NOT operator takes a single Boolean operand and reverses its value. It changes FALSE to TRUE and TRUE to FALSE. Note that this operator is usually used to reverse the result of a comparison operator.

Examples:

NOT ("SiteMinder" ENDS_WITH "R")
Result = TRUE

NOT ("SiteMinder " ~ENDS_WITH "R")
Result = FALSE

More information:

AND Operator (see page 481)
OR Operator (see page 482)
Exclusive OR Operator (see page 482)

AND Operator

The AND operator (also written & and &&) takes two Boolean operands and returns TRUE if both operands are TRUE. Note that this operator is usually used to connect two comparisons.

The evaluator does not evaluate the second Boolean operand if the first one is FALSE (since the result must be FALSE).

Examples:

(1 > 2) AND ("JUAN" ~= "JUAN")
Result = FALSE

(1 < 2) AND ("JUAN" ~= "JUAN")
Result = TRUE

More information:

NOT Operator (see page 481)
OR Operator (see page 482)
Exclusive OR Operator (see page 482)

OR Operator

The OR operator (also written | or ||) takes two Boolean operands and returns TRUE if either operand is TRUE. Note that this operator is usually used to connect two comparisons.

The evaluator does not evaluate the second Boolean operand if the first one is TRUE (since the result must be TRUE already).

Examples:

(1 > 2) OR ("JUAN" ~= "JUAN")
Result = TRUE

(1 < 2) OR ("JUAN" ~= "JUAN")
Result = TRUE

More information:

NOT Operator (see page 481)
AND Operator (see page 481)
Exclusive OR Operator (see page 482)

Exclusive OR Operator

The exclusive OR (XOR) operator takes two Boolean operands and returns TRUE if either operand is TRUE, but not both. This operator is usually used to connect two comparisons.

Examples:

(1 > 2) XOR ("JUAN" ~= "JUAN")
Result = TRUE

(1 < 2) XOR ("JUAN" ~= "JUAN")
Result = FALSE

More information:

NOT Operator (see page 481)
AND Operator (see page 481)
OR Operator (see page 482)

String Concatenation Operator

The string concatenation operator (+) returns a string that contains the combination of its two string operands.

If the first operand is a string, but the second is a number or a Boolean, the second operand is converted to a string. If the first operand is a number, the operator (+) indicates arithmetic addition, not concatenation.

Examples:

"JUAN" + " " + "Jones"
Result = "JUAN Jones"

"JUAN" + 2
Result = "JUAN2"

Arithmetic Addition Operator

The arithmetic addition (+) operator returns the sum of two numeric operands.

If the first operand is a number, but the second is a string or a Boolean, the second operand is converted to a number.

Examples:

1 + 2
Result = 3

1 + "JUAN"
Result = 1

1 + "32JUAN"
Result = 33

More information:

Arithmetic Subtraction Operator (see page 484)
Arithmetic Multiplication Operator (see page 484)
Arithmetic Division Operator (see page 485)

Arithmetic Subtraction Operator

The arithmetic subtraction (-) operator returns the difference between two numeric operands.

If the first operand is a number, but the second is a string or a Boolean, the second operand is converted to a number.

Examples:

1 - 2
Result = -1

1 - "JUAN"
Result = 1

100 - "32JUAN"
Result = 68

More information:

Arithmetic Addition Operator (see page 483)
Arithmetic Multiplication Operator (see page 484)
Arithmetic Division Operator (see page 485)

Arithmetic Multiplication Operator

The arithmetic multiplication (*) operator returns the product of two operands.

If the first operand is a number, but the second is a string or a Boolean, the second operand is converted to a number.

Examples:

1 * 2
Result = 2

1 * "JUAN"
Result = 0

100 * "32JUAN"
Result = 3200

More information:

Arithmetic Addition Operator (see page 483)
Arithmetic Subtraction Operator (see page 484)
Arithmetic Division Operator (see page 485)

Arithmetic Division Operator

The arithmetic division operator (/) returns the quotient of two operands.

If the first operand is a number, but the second is a string or a Boolean, the second operand is converted to a number.

All division is integer division.

Note: Division by zero, which is arithmetically undefined, always results in an error in this environment.

Examples:

1 / 2
Result = 0

1 / "JUAN"
Result = 0

100 / "32JUAN"
Result = 3

More information:

Arithmetic Addition Operator (see page 483)
Arithmetic Subtraction Operator (see page 484)
Arithmetic Multiplication Operator (see page 484)

Conditional Operator

The conditional operator evaluates a Boolean expression (the first operand) and based on the result returns one of two other operands.

If the Boolean operand is TRUE, the result of the operator is the second operand (the THEN clause), otherwise the third operand is returned (the ELSE clause). The second and the third operand must be the same type.

Only one clause of this operator is ever evaluated. In other words, if the THEN clause (operand) is evaluated, the ELSE clause (operand) is not evaluated. (And the opposite is also true.)

Examples:

"JUAN" = "juan" ? "YES" : "NO"
Result = "NO"

"JUAN" ~= "juan" ? "YES" : "NO"
Result = "YES"

Indexing Operator

The indexing operator takes two arguments: a set and a number (the index). The set is broken into components and the component corresponding to the specified index is returned (like a traditional array). If the index is out of range, a blank string is returned.

The first element in the set is at index zero (0).

Examples:

"Sun^Mon^Tue^Wed^Thu^Fri^Sat"[2]
Result = "Tue"

"Sun^Mon^Tue^Wed^Thu^Fri^Sat"[0]
Result = "Sun"

Functions Available within Expressions

This topic lists all of the functions by area of utility.

Numeric Functions

- ABS
- ALL
- ANDBITS
- ANY
- HEX
- MAX
- MIN
- MOD

- NOTBITS
- ORBITS
- SIGN
- XORBITS

String Functions

- AFTER
- BEFORE
- CENTER
- CHAR
- FIND
- LCASE
- LEFT
- LEN
- LPAD
- LTRIM
- MID
- PCASE
- RIGHT
- RPAD
- RPT
- RTRIM
- SPACE
- TRANSLATE
- UCASE

Set Functions

- COUNT
- ENUMERATE
- FILTER

- LOOP
- SORT

Date Functions

- DATE (form 1)
- DATE (form 2)
- DATEFROMSTRING
- DATETOSTRING
- DAY
- DOW
- DOY
- HOUR
- HOUR24
- MINUTE
- MONTH
- NOW
- NOWGMT
- SECOND
- YEAR
- YEAR4

Conversion Functions

- BOOLEAN
- NUMBER
- STRING

Generic User Directory I/O Functions

- GET
- SET

LDAP Functions

- ABOVE
- AT
- BELOW
- COMMONDN

- EXPLODEDN
- PARENTDN
- RDN
- RELATIONDN

URL/Path Handling Functions

- QS
- URL
- URLDECODE
- URLENCODE

Logging Functions

- ERROR
- INFO
- TRACE
- WARNING

File I/O Functions

- EXISTS
- KEY
- LOG

Error Handling Functions

- MAYBE
- THROW
- VEXIST

User Context Management Functions

- WITH

Miscellaneous Functions

- EVALUATE

ABOVE Function--Is User Above Specified LDAP DN

The ABOVE function returns a TRUE if the current user exists within a container that is *above* the container specified by the LDAP Distinguished Name (DN) that is passed to the function.

Note: The user's DN and the specified DN are assumed to be in the same directory.

Syntax

The ABOVE function has the following format:

```
ABOVE(LDAP_Distinguished_Name)
```

Parameters

The ABOVE function accepts the following parameter:

LDAP_Distinguished_Name (string)

Return Value

The ABOVE function returns a Boolean value.

Remarks

LDAP Only: Yes

ABS Function--Find the Absolute Value

The ABS function finds the absolute value of a number.

Syntax

The ABS function has the following format:

```
ABS(number)
```

Parameters

The ABS function accepts the following parameter:

number (number)

Return Value

The ABS function returns a number.

Example

```
Return_value=ABS(3)
Return_value=3

Return_value=ABS(-2)
Return_value=2
```

AFTER Function--Find a String

The AFTER function finds the specified instance of a search string in a source string and returns that part of the source string that follows the search string. If the search string is not found, then the AFTER function returns a blank string.

Syntax

The AFTER function has the following format:

```
AFTER(source_string, search_string[, not_case_sensitive][, n])
```

Parameters

The AFTER function accepts the following parameters:

source_string (string)

search_string (string)

not_case_sensitive (Boolean)

(Optional) Specifies case sensitivity. If the *not_case_sensitive* flag is omitted or set to FALSE, the function searches the source string for an exact match. If the *not_case_sensitive* flag is set to TRUE, the function ignores case.

n (number)

(Optional) Specifies the instance of the search string in the source string. If *n* is set to zero or one or omitted, the function finds the first instance of the search string. Otherwise, the function finds the *nth* instance of the search string. If *n* is negative, the function begins the search at the end of the source string.

Return Value

The AFTER function returns a string.

Example

```
Return_value=AFTER('EricEric', 'r')
Return_value='icEric'

Return_value=AFTER('EricEric', 'R')
Return_value=''

Return_value=AFTER('EricEric', 'R', TRUE)
Return_value='icEric'

Return_value=AFTER('EricEric', 'r', -1)
Return_value='ic'

Return_value=AFTER('EricEric', 'R', -1)
Return_value=''

Return_value=AFTER('EricEric', 'R', TRUE, -1)
Return_value='ic'

Return_value=AFTER('EricEric', 'r', 2)
Return_value='ic'

Return_value=AFTER('EricEric', 'R', 2)
Return_value=''

Return_value=AFTER('EricEric', 'R', TRUE, 2)
Return_value='ic'
```

More information:

BEFORE Function--Find a String (see page 495)
FIND Function--Return Position in String (see page 515)

ALL Function--All Bits Set

The ALL function accepts two numbers and returns a TRUE if *all* of the bits that are set in the second number are also set in the first number.

Syntax

The ALL function has the following format:

```
ALL(number1, number2)
```

Parameters

The ALL function accepts the following parameters:

number1 (number)

number2 (number)

Return Value

The ALL function returns a Boolean value.

Example

```
Return_value=ALL(7, 2)
Return_value=TRUE

Return_value=ALL(7, 15)
Return_value=FALSE
```

ANDBITS Function--Perform a Bitwise AND Operation

The ANDBITS function performs a bitwise AND operation on two numbers.

Syntax

The ANDBITS function has the following format:

```
ANDBITS(number1,number2)
```

Parameters

The ANDBITS function accepts the following parameters:

number1 (number)

number2 (number)

Return Value

The ANDBITS function returns a number.

Example

```
Return_value=ANDBITS(7,2)
Return_value=2

Return_value=ANDBITS(7,15)
Return_value=7
```

More information:

NOTBITS Function--Perform a Bitwise NOT (see page 534)
ORBITS Function--Perform a Bitwise OR Operation (see page 537)
XORBITS Function--Perform a Bitwise XOR Operation (see page 560)

ANY Function--Any Bits Set

The ANY function accepts two numbers and returns a TRUE if *any* of the bits that are set in the second number are also set in the first number.

Syntax

The ANY function has the following format:

```
ANY(number1, number2)
```

Parameters

The ANY function accepts the following parameters:

number1 (number)

number2 (number)

Return Value

The ANY function returns a Boolean value.

Example

```
Return_value=ANY(7, 2)
Return_value=TRUE

Return_value=ANY(7, 15)
Return_value=TRUE
```

AT Function--Is User at Specified LDAP DN

The AT function returns a TRUE if the current user exists within the container specified by the LDAP Distinguished Name (DN) that is passed to the function.

Note: The user's DN and the specified DN are assumed to be in the same directory.

Syntax

The AT function has the following format:

AT(LDAP_Distinguished_Name)

Parameters

The AT function accepts the following parameter:

LDAP_Distinguished_Name (string)

Return Value

The AT function returns a Boolean value.

Remarks

LDAP Only: Yes

BEFORE Function--Find a String

The BEFORE function finds the specified instance of a search string in a source string and returns that part of the source string that precedes the search string. If the search string is not found, then the BEFORE function returns the entire source string.

Syntax

The BEFORE function has the following format:

BEFORE(source_string, search_string[, not_case_sensitive][, n])

Parameters

The BEFORE function accepts the following parameters:

source_string (string)

search_string (string)

not_case_sensitive (Boolean)

Specifies case sensitivity. If the *not_case_sensitive* flag is set to FALSE or omitted, the BEFORE function searches the source string for an exact match. If the *not_case_sensitive* flag is set to TRUE, the function ignores case.

n (number)

Specifies the instance of the search string in the source string. If *n* is set to zero or one or omitted, the BEFORE function finds the first instance of the search string. Otherwise, the function finds the *nth* instance of the search string. If *n* is negative, the function begins the search at the end of the source string.

Return Value

The BEFORE function returns a string.

Example

```
Return_value=BEFORE('EricEric', 'r')
Return_value='E'

Return_value=BEFORE('EricEric', 'R')
Return_value='EricEric'

Return_value=BEFORE('EricEric', 'R', TRUE)
Return_value='E'

Return_value=BEFORE('EricEric', 'r', -1)
Return_value='EricE'

Return_value=BEFORE('EricEric', 'R', -1)
Return_value='EricEric'

Return_value=BEFORE('EricEric', 'R', TRUE, -1)
Return_value='EricE'

Return_value=BEFORE('EricEric', 'r', 2)
Return_value='EricE'

Return_value=BEFORE('EricEric', 'R', 2)
Return_value='EricEric'

Return_value=BEFORE('EricEric', 'R', TRUE, 2)
Return_value='EricE'
```

More information:

AFTER Function--Find a String (see page 491)
FIND Function--Return Position in String (see page 515)

BELOW Function--Is User Below Specified LDAP DN

The BELOW function returns a TRUE if the current user exists within a container that is *below* the container specified by the LDAP Distinguished Name (DN) that is passed to the function.

Note: The user's DN and the specified DN are assumed to be in the same directory.

Syntax

The BELOW function has the following format:

```
BELOW(LDAP_Distinguished_Name)
```

Parameters

The BELOW function accepts the following parameter:

LDAP_Distinguished_Name (string)

Return Value

The BELOW function returns a Boolean value.

Remarks

LDAP Only: Yes

BOOLEAN Function--Convert to "TRUE" or "FALSE"

The BOOLEAN function converts a number or string value to a string value of either "TRUE" or "FALSE". Numeric values of zero are converted to "FALSE". All other numeric values are converted to "TRUE". String values of "true" or "yes" are converted to "TRUE". All other string values are converted to "FALSE". The BOOLEAN function is not case-sensitive.

Syntax

The BOOLEAN function has the following format:

```
BOOLean(number | string)
```

Parameters

The BOOLEAN function accepts either one of the following two parameters:

number (number)

string (string)

Return Value

The BOOLEAN function returns one of the following string values:

- "TRUE"
- "FALSE"

Example

```
Return_value=BOOLEAN('Phoebe')
Return_value="FALSE"

Return_value=BOOLEAN('Yes')
Return_value="TRUE"

Return_value=BOOLEAN(123)
Return_value="TRUE"

Return_value=BOOLEAN(0)
Return_value="FALSE"
```

More information:

NUMBER Function--Convert to a Numeric Value (see page 536)
STRING Function--Convert to a String (see page 550)

CHAR Function--Convert an ASCII Value

The CHAR function converts the specified ASCII value to a one-character string.

Syntax

The CHAR function has the following format:

```
CHaR(ASCII_value)
```

Parameters

The CHAR function accepts the following parameter:

ASCII_value (number)

Specifies the ASCII value. If the *ASCII_value* is zero, the function returns an empty string. If the *ASCII_value* is greater than 255, the function converts the modulus 256 of the *ASCII_value* to a one-character string.

Note: Performing a modulus 256 on a value is equivalent to performing a bitwise AND with 255.

Return Value

The CHAR function returns a one-character string.

Example

```
Return_value=CHAR(0)
Return_value=''

Return_value=CHAR(36)
Return_value='$'

Return_value=CHAR(299)
Return_value='+'
```

CENTER Function--Pad a Source String

The CENTER function pads both ends of a source string with a specified character until the resulting string is a specified length. If the padding is uneven, the function adds an extra character to the end of the string.

Syntax

The CENTER function has the following format:

```
CENTER(source, length[, padding])
```

Parameters

The CENTER function accepts the following parameters:

source (string or number)

Specifies the source string. The function automatically converts a number to a string.

length (number)

Specifies the length of the resulting string.

padding (string)

(Optional) Specifies the character that the function uses to pad the source string.

- If the *padding* is more than one character long, the function uses the first character.
- If the *padding* is an empty string, the function does not pad the source.
- If the *padding* is omitted and the source is a string, the function uses a space for padding.
- If the *padding* is omitted and the source is a number, the function uses a zero for padding.

Return Value

The CENTER function returns a string.

Example

```
Return_value=CENTER('Robin',  9, '*')
Return_value='**Robin**'

Return_value=CENTER('Robin', 7, 'ooo')
Return_value='oRobino'

Return_value=CENTER('Robin', 7, '')
Return_value='Robin'

Return_value=CENTER('Robin', 11)
Return_value='   Robin'

Return_value=CENTER(123, 9)
Return_value='000123000'
```

More information:

LPAD Function--Pad a Source String on the Left (see page 526)
RPAD Function--Pad a String on the Right (see page 544)

COMMONDN Function--Find a Common Root

The COMMONDN function returns the common root of two LDAP distinguished names (DNs) without calling an LDAP server.

Syntax

The COMMONDN function has the following format:

```
COMMONDN(ldapdn1, ldapdn2)
```

Parameters

The COMMONDN function accepts the following parameters:

ldapdn1 (string)

Specifies an LDAP distinguished name (DN).

ldapdn2 (string)

Specifies an LDAP distinguished name (DN).

Note: If either *ldapdn1* or *ldapdn2* is not a valid LDAP DN or if the two DNs do not share a common root, the function returns an empty string. An LDAP DN is not case-sensitive.

Return Value

The COMMONDN function returns a string.

Remarks

LDAP Only: Yes

Example

```
Return_value=COMMONDN('uid=Vincent,o=NDS.com',
'ou=People,o=nds.com')
Return_value='o=NDS.com'
```

More information:

EXPLODEDN Function--Convert LDAP DN to Set (see page 513)
PARENTDN Function--Retrieve Parent in LDAP Tree (see page 538)
RDN Function--Retrieve First Component of LDAP DN (see page 541)
RELATIONDN Function--Compare Two Distinguished Names (see page 542)

COUNT Function--Count the Elements in a Set

The COUNT function counts the number of elements in a set.

Syntax

The COUNT function has the following format:

```
COUNT(set[, case_sensitive])
```

Parameters

The COUNT function accepts the following parameters:

set (string)

Specifies a string of elements that are separated by the caret character: 'element1^element2'. Each element is a string.

case_sensitive (Boolean)

(Optional) Specifies case sensitivity.

- If the *case_sensitive* flag is omitted, the function counts all elements.
- If the *case_sensitive* flag is supplied, the function counts only unique elements.
- If the *case_sensitive* flag is TRUE, the function is case-sensitive.
- If the *case_sensitive* flag is FALSE, the function is not case-sensitive.

Return Value

The COUNT function returns a number.

Example

```
Return_value=COUNT('phoebe^PHOEBE^robin^robin')
Return_value=4

Return_value=COUNT('phoebe^PHOEBE^robin^robin', FALSE)
Return_value=2

Return_value=COUNT('phoebe^PHOEBE^robin^robin', TRUE)
Return_value=3
```

More information:

ENUMERATE Function--Test Set Elements (see page 509)
FILTER Function--Test Set Elements (see page 514)
SORT Function--Sort a Set (see page 549)

DATE Function--Set to Midnight (form 1)

The DATE function (form 1) accepts a numeric representation of date and time and sets the time to midnight on the specified date.

Syntax

The DATE function (form 1) has the following format:

```
DATE(date_time)
```

Parameters

The DATE function (form1) accepts the following parameter:

date_time (number)

Specifies the date and time as the number of seconds that have passed since January 1, 1970. If *date_time* is not a valid representation of date and time, the function returns -1.

Return Value

The DATE function (form1) returns a number.

DATE Function--Convert Year, Month, Day, Hours, Minutes, and Seconds (form 2)

The DATE function (form 2) accepts six numbers that represent the year, month, day, hours, minutes, and seconds and converts them to a numeric representation of date and time. Date and time are represented numerically as the number of seconds that have passed since January 1, 1970. If the three time parameters are omitted, the function sets the time to midnight on the specified date.

Syntax

The DATE function (form 2) has the following format:

```
DATE(year, month, day[, hours, minutes, seconds])
```

Parameters

The DATE function (form 2) accepts the following parameters:

year (number)

Specifies a four-digit representation of the year.

Example: 2007

month (number)

Specifies the month.

Range: 1-12

day (number)

Specifies the day.

Range: 1-31

hours (number)

(Optional) Specifies the number of hours.

Range: 0-23

minutes (number)

(Optional) Specifies the number of minutes.

Range: 0-59

seconds (number)

(Optional) Specifies the number of seconds.

Range: 0-59

Return Value

The DATE function (form 2) returns a number.

DATEFROMSTRING Function--Convert String to Number

The DATEFROMSTRING function accepts a string representation of a date, a time, or both and converts the string to a numeric representation. Date and time are represented numerically as the number of seconds that have passed since January 1, 1970.

Syntax

The DATEFROMSTRING function has the following format:

```
DATEFROMSTRING(date_time[, format_string])
```

Parameters

The DATEFROMSTRING function accepts the following parameters:

date_time (string)

format_string (string)

(Optional) Specifies the format of *date_time*. For example, the *format_string* "YYYYMMDD" specifies the format of "20020223." If the *format_string* is omitted, the function uses the default format of "YYYYMMDDhhmmss." If *date_time* is invalid, the function returns -1.

Return Value

The DATEFROMSTRING function returns a number.

Example

Return_value=DATEFROMSTRING('20020223')
Return_value=1024804800

DATETOSTRING Function--Convert Number to String

The DATETOSTRING function accepts a numeric representation of a date, a time, or both and converts the number to a string representation.

Syntax

The DATETOSTRING function has the following format:

DATETOSTRING(date_time[, format_string])

Parameters

The DATETOSTRING function accepts the following parameters:

date_time (number)

Specifies a date, a time, or both as the number of seconds that have passed since January 1, 1970. If the *date_time* is invalid, the function returns the value "INVALID DATE".

format_string (string)

(Optional) Specifies the format of a date, a time, or both in the resulting string using the format codes listed in the table. Any characters or character combinations not listed in the table are inserted in the resulting string as is. If the *format_string* is omitted, the function formats the resulting string using the date and time representation appropriate for the locale that is currently stored in the Policy Server. The format codes and resulting formats are:

Code	Resulting Format
%a	Abbreviated weekday name (English)
%A	Full weekday name (English)
%b	Abbreviated month name (English)
%B	Full month name (English)
%c	Date and time representation appropriate for current locale
%#c	Long date and time representation appropriate for current locale
%d	Day of month as decimal number (01-31)
%H	Hour in 24-hour format (00-23)
%I	Hour in 12-hour format (01-12)
%j	Day of year as decimal number (001-366)
%m	Month as decimal number (01-12)
%M	Minute as decimal number (00-59)
%P	Local am and pm indicator for 12-hour clock
%S	Second as decimal number (00-59)
%U	Week of year as decimal number with Sunday as first day of week (00-53)
%w	Weekday as decimal number (0-6 with Sunday as 0)
%W	Week of year as decimal number with Monday as first day of week (00-53)
%x	Date representation appropriate for current locale
%#x	Long date representation appropriate for current locale
%X	Time representation appropriate for current locale
%y	Year without century as decimal number (00-99)
%Y	Year with century as decimal number
%z, %Z	Time-zone name or abbreviation if known
%%	Percent sign

Note: The pound sign modifies the meaning of the format codes, as follows.

- The format code %#c specifies the long date and time representation appropriate for the current locale.
- The format code %#x specifies the long date representation appropriate for the current locale.
- In all other cases, the pound sign, which is inserted in the format code after the percent sign, results in the removal of leading zeros, if any.
- The pound sign has no effect on codes that format alpha characters, such as the names of months.

Return Value

The DATETOSTRING function returns a string.

Example

```
Return_value=DATETOSTRING(1024804800, '%Y%m%d')
Return_value='20020223'

Return_value=DATETOSTRING(1024804800, '%Y%#m%#d')
Return_value='2002223'
```

DAY Function--Return Day of Month

The DAY function accepts a numeric representation of date and time and returns the number corresponding to the day of the month.

Syntax

The DAY function has the following format:

```
DAY(date_time)
```

Parameters

The DAY function accepts the following parameter:

date_time (number)

Specifies the date and time as the number of seconds that have passed since January 1, 1970. If the date is invalid, the function returns -1.

Return Value

The DAY function returns a number from 1 to 31.

Example

Return_value=DAY(1024804800)
Return_value=23

More information:

DOW Function--Return Day of Week (see page 508)
DOY Function--Return Day of Year (see page 509)

DOW Function--Return Day of Week

The DOW function accepts a numeric representation of date and time and returns a number corresponding to the day of the week.

Syntax

The DOW function has the following format:

DOW(date_time)

Parameters

The DOW function accepts the following parameter:

date_time (number)

Specifies the date and time as the number of seconds that have passed since January 1, 1970. If the date is invalid, the function returns -1.

Return Value

The DOW function returns a number from 0 (Sunday) to 6 (Saturday).

Example

Return_value=DOW(1024804800)
Return_value=0

More information:

DAY Function--Return Day of Month (see page 507)
DOY Function--Return Day of Year (see page 509)

DOY Function--Return Day of Year

The DOY function accepts a numeric representation of date and time and returns a number corresponding to the day of the year.

Syntax

The DOY function has the following format:

```
DOY(date_time)
```

Parameters

The DOY function accepts the following parameter:

date_time (number)

Specifies the date and time as the number of seconds that have passed since January 1, 1970. If the date is invalid, the function returns -1.

Return Value

The DOY function returns a number from 1 to 366.

Example

Return_value=DOY(1024804800)
Return_value=173

More information:

DAY Function--Return Day of Month (see page 507)
DOW Function--Return Day of Week (see page 508)

ENUMERATE Function--Test Set Elements

The ENUMERATE function passes each element in the specified set to the named expression specified by the parameter *#virtual_user_attribute* or *@user_class* using pasting. If the named expression returns a value, the element is included in the resulting string.

Syntax

The ENUMERATE function has the following format:

```
ENUMERATE(set, #virtual_user_attribute | @user_class)
```

Parameters

The ENUMERATE function accepts the following parameters:

set (string)

Specifies a string of elements that are separated by the caret character: 'element1^element2'. Each element is a string.

#virtual_user_attribute (named expression)

Specifies a named expression that calculates a user attribute.

@user_class (named expression)

Specifies a named expression that tests for membership in a user directory or group.

Return Value

The ENUMERATE function returns a string.

Example

Assume that the following statements are true:

- The *#virtual_user_attribute* is #RTrimset.
- The named expression specified by #RTrimset is the built-in function RTRIM.
- RTRIM accepts one parameter.
- The parameter is represented by the placeholder %0.
- Therefore, #RTrimset := RTRIM(%0).

```
Return_value=ENUMERATE('First     ^Middle     ^Last      ', #RTrimset)
Return_value='First^Middle^Last'
```

More information:

COUNT Function--Count the Elements in a Set (see page 502)
FILTER Function--Test Set Elements (see page 514)
LOOP Function--Call a Virtual Attribute in a Loop (see page 525)
SORT Function--Sort a Set (see page 549)
Pasting (see page 467)

ERROR Function--Write Error Message to Console Log

The ERROR function writes the specified error message to SOA Security Manager's Console Log.

Syntax

The ERROR function has the following format:

```
ERROR(error_message)
```

Parameters

The ERROR function accepts the following parameter:

error_message (string)

Return Value

The ERROR function returns an empty string. When used in a Boolean context, the ERROR function returns the value TRUE.

Example

```
Return_value=ERROR('Invalid Access')
Return_value=''
```

More information:

INFO Function--Write INFO Message to Console Log (see page 520)
TRACE Function--Write Trace Entry to Console Log (see page 552)
TRACE Function--Write Trace Entry to Console Log (see page 559)

EVALUATE Function--Evaluate an Expression

The EVALUATE function evaluates an expression in the context of the current user and returns the result as a string. If an optional user path is provided, the function evaluates the expression in the context of the specified user.

Syntax

The EVALUATE function has the following format:

```
EVALUATE([user_path, ]expression)
```

Parameters

The EVALUATE function accepts the following parameters:

user_path (string)

(Optional) Specifies a user other than the current user.

expression (string)

Specifies the expression to be evaluated.

Return Value

The EVALUATE function returns a string.

Remarks

Privileged: Yes

Example

```
Return_value=EVALUATE("sn + ',' + givenname")
Return_value="Hood, Robin"

Return_value=EVALUATE(manager, "sn + ',' + givenname")
Return_value="Webb, Charlotte"
```

EXISTS Function--Look Up File Name

The EXISTS function looks up the specified file and returns a TRUE if the file exists. Otherwise, the function returns a FALSE.

Syntax

The EXISTS function has the following format:

```
EXISTS(filename)
```

Parameters

The EXISTS function accepts a filename.

filename (string)

Return Value

The EXISTS function returns a Boolean value.

Remarks

Privileged: Yes

Example

Return_value=EXISTS('SmUtilities.dll')
Return_value=TRUE

More information:

KEY Function--Look Up Key (see page 520)

EXPLODEDN Function--Convert LDAP DN to Set

The EXPLODEDN function converts an LDAP distinguished name (DN) to a set without calling an LDAP server.

Syntax

The EXPLODEDN function has the following format:

EXPLODED(ldapdn[, remove_attribute_names])

Parameters

The EXPLODEDN function accepts the following parameters:

ldapdn (string)

Specifies an LDAP distinguished name (DN).

remove_attribute_names (Boolean)

(Optional) Specifies the *remove_attribute_names* flag. If the flag is TRUE, the function removes the attribute names from the LDAP distinguished name (DN).

Return Value

The EXPLODEDN function returns a string of elements that are separated by the caret character: 'element1^element2'. Each element is a string.

Remarks

LDAP Only: Yes

Example

Return_value=EXPLODEDN('uid=hawk,o=NDS.com', FALSE)
Return_value='uid=hawk^o=NDS.com'

Return_value=EXPLODEDN('uid=hawk,o=NDS.com', TRUE)
Return_value='hawk^NDS.com'

More information:

COMMONDN Function--Find a Common Root (see page 501)
PARENTDN Function--Retrieve Parent in LDAP Tree (see page 538)
RDN Function--Retrieve First Component of LDAP DN (see page 541)
RELATIONDN Function--Compare Two Distinguished Names (see page 542)

FILTER Function--Test Set Elements

The FILTER function compares each element in the specified set to the specified pattern and returns a new set containing only the elements that match the pattern. If the optional NOT operator is included, the FILTER function returns a new set containing only the elements that do not match the pattern.

Syntax

The FILTER function has the following format:

FILTER(set, [NOT]pattern)

Parameters

The FILTER function accepts the following parameters:

set (string)

Specifies a string of elements that are separated by the caret character: 'element1^element2'. Each element is a string.

pattern (string)

Specifies a pattern of characters. A *pattern* can include single characters, ranges of characters, or both. A range of characters is expressed as two characters separated by a hyphen. The following are examples of valid character ranges: 0-9, a-z, and A-Z. The *pattern* parameter can also include characters that are reserved and have special meanings. The reserved characters are:

Character	Meaning
?	Matches any single character

Character	Meaning
*	Matches any sequence of characters, including one or none
[set]	Matches any single character in the specified set
[!set] or [^set]	Matches any single character *not* in the specified set
\	Treats any reserved character as a regular character, such as *

NOT (operator)

(Optional) If the NOT operator is included, the function returns a new set containing only the elements that do not match the pattern.

Return Value

The FILTER function returns a string of elements separated by the caret character: 'element1^element2'. Each element is a string.

Example

Return_value=FILTER('Faith^Earl^Emilie^Fred', 'E*')
Return_value='Earl^Emilie'

Return_value=FILTER('Faith^Earl^Emilie^Fred', NOT 'E*')
Return_value='Faith^Fred'

More information:

COUNT Function--Count the Elements in a Set (see page 502)
ENUMERATE Function--Test Set Elements (see page 509)
SORT Function--Sort a Set (see page 549)

FIND Function--Return Position in String

The FIND function finds the specified instance of the search string in the source string and returns its position. If the search string is not found, then the function returns a zero.

Syntax

The FIND function has the following format:

```
FIND(source_string, search_string[, not_case_sensitive][, n])
```

Parameters

The FIND function accepts the following parameters:

source_string (string)

search_string (string)

not_case_sensitive (Boolean)

(Optional) Specifies case sensitivity. If the *not_case_sensitive* flag is omitted or set to FALSE, the function searches the source string for an exact match. If the *not_case_sensitive* flag is set to TRUE, the FIND function ignores case.

n (number)

(Optional) Specifies the instance of the search string in the source string. If *n* is set to zero or one or omitted, the function finds the first instance of the search string. Otherwise, the function finds the *nth* instance of the search string. If *n* is negative, the function begins the search at the end of the source string.

Return Value

The FIND function returns a number.

Example

```
Return_value=FIND('PhoebePhoebe', 'oe', FALSE, 1)
Return_value=3

Return_value=FIND('PhoebePhoebe', 'OE', FALSE, 1)
Return_value=0

Return_value=FIND('PhoebePhoebe', 'OE', TRUE, 1)
Return_value=3

Return_value=FIND('PhoebePhoebe', 'oe', FALSE, -1)
Return_value=9

Return_value=FIND('PhoebePhoebe', 'OE', FALSE, -1)
Return_value=0

Return_value=FIND('PhoebePhoebe', 'OE', TRUE, -1)
Return_value=9
```

Return_value=FIND('PhoebePhoebe', 'oe', FALSE, 2)
Return_value=9

Return_value=FIND('PhoebePhoebe', 'OE', FALSE, 2)
Return_value=0

Return_value=FIND('PhoebePhoebe', 'OE', TRUE, 2)
Return_value=9

More information:

AFTER Function--Find a String (see page 491)
BEFORE Function--Find a String (see page 495)

GET Function--Locate Attributes in a User Directory

The GET function locates the specified attribute or attributes in a user directory and returns the attribute values. Multiple attribute values are separated by the caret character. If the function cannot find an attribute, it returns an empty string.

Syntax

The GET function has the following format:

GET(user_attribute_name | user_attributes_string)

Parameters

The GET function accepts one of the following parameters:

user_attribute_name (unquoted string)

Specifies a single user attribute.

user_attributes_string (string)

Specifies a string of user attribute names separated by a character. The function uses this character to separate one attribute's values from another attribute's values in the resulting string.

Return Value

The GET function returns a string.

Remarks

Privileged: Yes

LDAP Only: Yes

Example

```
Return_value=GET('sn,givenname')
Return_value='Finch,Robin'
```

HEX Function--Convert to Hexadecimal

The HEX function converts a decimal number to a hexadecimal number and returns it as a string.

Syntax

The HEX function has the following format:

```
HEX(decimal_number)
```

Parameters

The HEX function accepts the following parameter:

decimal_number (number)

Return Value

The HEX function returns a string.

Example

```
Return_value=HEX(16)
Return_value='10'
```

HOUR Function--Convert to Hour

The HOUR function converts the specified date and time to an hour from 1 to 12.

Syntax

The HOUR function has the following format:

```
HOUR(date_time)
```

Parameters

The HOUR function accepts the following parameter:

date_time (number)

Specifies the date and time as the number of seconds that have passed since January 1, 1970. If *date_time* is not a valid representation of date and time, the function returns -1.

Return Value

The HOUR function returns a number from 1 to 12.

More information:

HOUR24 Function--Convert to Hour (see page 519)

HOUR24 Function--Convert to Hour

The HOUR24 function converts the specified date and time to an hour from 0 to 23.

Syntax

The HOUR24 function has the following format:

```
HOUR24(date_time)
```

Parameters

The HOUR24 function accepts the following parameter:

date_time (number)

Specifies the date and time as the number of seconds that have passed since January 1, 1970. If *date_time* is not a valid representation of date and time, the function returns -1.

Return Value

The HOUR24 function returns a number from 0 to 23.

More information:

HOUR Function--Convert to Hour (see page 518)

INFO Function--Write INFO Message to Console Log

The INFO function writes the string argument to the SOA Security Manager Console Log as an INFO message.

Syntax

The INFO function has the following format:

```
INFO(source_string)
```

Parameters

The INFO function accepts the following parameter:

source_string (string)

Return Value

The INFO function returns a Boolean, always TRUE.

Example

```
Return_value=INFO("86% Complete")
Return_value=TRUE
```

KEY Function--Look Up Key

The KEY function looks up the specified key name in the specified application section of the specified file and returns a key value. If the key, application, or file is not found, then the KEY function returns an empty string.

Syntax

The KEY function has the following format:

```
KEY(filename, [application_name, ]key_name)
```

Parameters

The KEY function accepts the following parameters:

filename (string)

Specifies the file. In the specified file, comment lines start with a semicolon, pound sign, or two forward slashes. Comment lines, blank lines, and leading and trailing spaces are ignored.

application_name (string)

(Optional) Specifies the name of the application section. In the specified file, application names are enclosed by square brackets and specify the beginning of an application section: *[application_name*]. Application names are case-sensitive.

key_name (string)

Specifies the name of the key. In the specified file, key names and values are enclosed by angle brackets and paired by an equal sign, one pair per line: <*key_name*>=<key_value>. Key names are case-sensitive.

Return Value

The KEY function returns a string.

Remarks

Privileged: Yes

LDAP Only: No

Example

Return_value=KEY('application.dat', 'login user')
Return_value='key_value'

More information:

EXISTS Function--Look Up File Name (see page 512)

LCASE Function--Convert to Lowercase

The LCASE function converts any uppercase letters in the specified string to lowercase.

Syntax

The LCASE function has the following format:

```
LCASE(specified_string)
```

Parameters

The LCASE function accepts the following parameter:

specified_string (string)

Return Value

The LCASE function returns a string.

Example

Return_value=LCASE('BARRED OWL')
Return_value='barred owl'

More information:

PCASE Function--Convert a String to Proper Case (see page 538)
UCASE Function--Convert to Upper Case (see page 553)

LEFT Function--Return Part of a String

The LEFT function returns a specified number of characters of a string. If the string is shorter than the specified number of characters, the entire string is returned.

Syntax

The LEFT function has the following format:

```
LEFT(source_string, length)
```

Parameters

The LEFT function accepts the following parameters:

source_string (string)

length (number)

Return Value

The LEFT function returns a string.

Example

```
Return_value=LEFT('JuanJuan', 2)
Return_value='Ju'

Return_value=LEFT('JuanJuan', 10)
Return_value=('JuanJuan')

Return_value=LEFT('JuanJuan', 0)
Return_value=''
```

More information:

RIGHT Function--Retrieve Characters from a String (see page 543)
MID Function--Return Part of a String (see page 530)

LEN Function--Return the Length of a String

The LEN function returns the length of a string.

Syntax

The LEN function has the following format:

```
LEN(source_string)
```

Parameters

The LEN function accepts the following parameter:

source_string (string)

Return Value

The function returns a number.

Example

```
Return_value=LEN("JuanJuan")
Return_value=8
```

LOG Function--Write a String to a File

The LOG function writes the string argument to the specified file.

Syntax

The LOG function has the following format:

```
LOG(filename, source_string)
```

Parameters

The LOG function accepts the following parameters:

filename (string)

source_string (string)

Return Value

The LOG function returns a Boolean, which is always TRUE.

Remarks

Privileged: Yes

Example

```
Return_value=LOG("auditlog.txt", "Accessing Realm XXX")
Return_value=TRUE
```

LOOP Function--Call a Virtual Attribute in a Loop

The LOOP function calls the virtual attribute once for each possible number in the loop, starting and ending at the specified values. If the step value is specified, the numbers are incremented by the step value. If the virtual attribute returns a non-blank value, the value is included in the resulting set.

Syntax

The LOOP function has the following format:

LOOP(start_value, end_value, [step,] #virtual_user_attribute)

Parameters

The LOOP function accepts the following parameters:

start_value (number)

end_value (number)

step (number)

(Optional) The default is 1. Negative values are allowed.

#virtual_user_attribute (Named Expression)

Name of a defined virtual attribute. The virtual attribute can access the current loop counter by using a reference to %0.

Return Value

The LOOP function returns a set.

Examples

For these examples assume the virtual user attribute is #Padset := LPAD(%0, 2)

Return_set=LOOP(1, 5, #Padset)
Return_set="001^002^003^004^005"

Return_set=LOOP(1, 5, 2, #Padset)
Return_set="001^003^005"

Return_set=LOOP(5, 1, -1, #Padset)
Return_set="005^004^003^002^001"

More information:

Pasting (see page 467)
ENUMERATE Function--Test Set Elements (see page 509)
FILTER Function--Test Set Elements (see page 514)

LPAD Function--Pad a Source String on the Left

The LPAD function pads a source string on the left with the first character of the specified padding until the resulting string is the specified length.

Syntax

The LPAD function has the following format:

```
LPAD(source_string, length[, padding])
```

Parameters

The LPAD function accepts the following parameters:

source_string (string)

This parameter can also be a number; it is automatically converted to a string.

length (number)

The number of characters of the final string.

padding (string)

(Optional) If the padding is more than one character long, only the first character is used. If the padding is zero length, no padding is done. If the source is a string and padding omitted, a space is used for padding. If the source is a number and padding is omitted, a zero is used for padding.

Return Value

The LPAD function returns a string.

Examples

```
Result_value=LPAD('Juan', 5)
Result_value=' Juan'

Result_value=LPAD('Juan', 5, 'X')
Result_value= 'XJuan'

Result_value=LPAD('Juan', 6, 'XY')
Result_value= 'XXJuan'

Result_value=LPAD(5, 2)
Result_value= '05'

Result_value=LPAD(5, 2, ' ')
Result_value=' 5'
```

More information:

CENTER Function--Pad a Source String (see page 499)
RPAD Function--Pad a String on the Right (see page 544)

LTRIM Function--Remove Leading Spaces in a String

The LTRIM function returns a string representing the *source_string* with any leading spaces removed.

Syntax

The LTRIM function has the following format:

```
LTRIM(source_string)
```

Parameters

The LTRIM function accepts the following parameter:

source_string (string)

Return Value

The LTRIM function returns a string.

Example

```
Return_value=LTRIM('  Juan ')
Return_value='Juan '
```

More information:

RTRIM Function--Remove Trailing Spaces from a String (see page 546)

MAX Function--Determine the Larger of Two Values

The MAX function returns the larger of two numeric arguments.

Syntax

The MAX function has the following format:

```
MAX(int_1, int_2)
```

Parameters

The MAX function accepts the following parameters:

int_1 (number)

int_2 (number)

Return Value

The MAX function returns a number.

Example

Return_value=MAX(-2, 4)
Return_value=4

More information:

MIN Function--Determine the Lesser of Two Numbers (see page 531)

MAYBE Function--Report an Indeterminate Result

You can write an expression that tests a condition and calls the function MAYBE if information needed for the test is missing or incomplete. When the expression evaluator encounters MAYBE, it tries to resolve the expression without the needed information. This is only possible when MAYBE is part of a compound expression.

For example, when the operator is AND and one operand is FALSE, the evaluator can determine that the result of the operation is FALSE. When one operand is TRUE and the other operand is undefined or both operands are undefined, the evaluator cannot determine the result of the operation. When both operands are TRUE, the result of the AND operation is TRUE.

AND Operator	True	False	Undefined
True	True	False	Indeterminate
False	False	False	False
Undefined	Indeterminate	False	Indeterminate

Likewise, when the operator is OR and one operand is TRUE, the evaluator can determine that the result of the operation is TRUE. When one operand is FALSE and the other operand is undefined or both operands are undefined, the evaluator cannot determine the result of the operation. When both operands are FALSE, the result of the OR operation is FALSE.

OR Operator	True	False	Undefined
True	True	True	True
False	True	False	Indeterminate
Undefined	True	Indeterminate	Indeterminate

If the evaluator cannot resolve the expression, it stops processing and the specified message is output to the console log or report depending on the context. MAYBE is typically called during role evaluation either in the context of a policy or report generation.

Typically, conditions depend on the time of day or an IP address or the value of a virtual user attribute (specified by the # sign), a user class (specified by the @ sign), a context variable (specified by the % sign), or a user attribute.

Note: In the case of LDAP user directories, the evaluator cannot determine whether a user attribute is defined.

Syntax

The MAYBE function has the following format:

```
MAYBE(message)
```

Parameters

The MAYBE function accepts the following parameter:

message (string)

Specifies the information that is missing and needed to evaluate a condition in an expression.

Return Value

The MAYBE function does not return.

Example

VEXIST(%ClientIP) ? #CheckIP : MAYBE('Client IP address is not defined.')
Message Output to the Console Log or Report: 'Client IP address is not defined.'

MID Function--Return Part of a String

The MID function returns the characters of the *source_string* starting at the *start* position (numbered from one) up to the specified *length*. If no *length* is specified, the rest of the source_string (after the *start* position) is returned.

Syntax

The MID function has the following format:

```
MID(source_string, start[,length])
```

Parameters

The MID function accepts the following parameters:

source_string (string)

start (number)

length (number) (Optional)

Return Value

The MID function returns a string.

Example

Return_value=MID('JuanJuan', 2, 3)
Return_value='uan'

Return_value=MID('JuanJuan', 2)
Return_value='uanJuan'

More information:

LEFT Function--Return Part of a String (see page 522)
RIGHT Function--Retrieve Characters from a String (see page 543)

MIN Function--Determine the Lesser of Two Numbers

The MIN function returns the lesser of two numeric arguments.

Syntax

The MIN function has the following format:

MIN(int_1, int_2)

Parameters

The MIN function accepts the following parameters:

int_1 (number)

int_2 (number)

Return Value

The MIN function returns a number.

Example

Return_value=MIN(-2, 4)
Return_value=-2

More information:

MAX Function--Determine the Larger of Two Values (see page 527)

MINUTE Function--Return the Minutes Component for a Date

The MINUTE function returns the number representing the minute component for a specified *date_time* expressed in seconds since January 1, 1970.

Syntax

The MINUTE function has the following format:

```
MINUTE(date_time)
```

Parameters

The MINUTE function accepts the following parameter:

date_time (number)

The date in the number of seconds.

Return Value

The MINUTE function returns a number between 0 and 59. If the *date_time* is invalid, MINUTE returns -1.

More information:

HOUR Function--Convert to Hour (see page 518)
HOUR24 Function--Convert to Hour (see page 519)
SECOND Function--Return the Number of Seconds in a Date (see page 546)

MOD Function--Return Division Remainder

The MOD function returns the modulus (remainder) of the division of the first number by the second. If the second number is zero, zero is returned.

Syntax

The MOD function has the following format:

```
MOD(int_1, int_2)
```

Parameters

The MOD function accepts the following parameters:

int_1 (number)

The dividend of the division operation.

int_2 (number)

The divisor of the division operation.

Return Value

The MOD function returns a number.

Example

Return_value=MOD(3, 2)
Return_value=1

Return_value=MOD(6, 3)
Return_value =0

MONTH Function--Return the Month Component of a Date

The MONTH function returns the number representing the month component for a specified *date_time* expressed in seconds since January1, 1970.

Syntax

The MONTH function has the following format:

MONTH(date_time)

Parameters

The MONTH function accepts the following parameter:

date_time (number)

The date represented in the number of seconds.

Return Value

The MONTH function returns a number between 1 and 12. If the *date_number* is invalid, MONTH returns -1.

More information:

DAY Function--Return Day of Month (see page 507)
DOW Function--Return Day of Week (see page 508)
DOY Function--Return Day of Year (see page 509)
YEAR Function--Return the Year Component of a Numeric Date (see page 560)
YEAR4 Function--Return the Year Component of a Date (4 digits) (see page 561)

NOTBITS Function--Perform a Bitwise NOT

The NOTBITS function performs a bitwise NOT operation on a number.

Syntax

The NOTBITS function has the following format:

NOTBITS(number)

Parameters

The NOTBITS function accepts the following parameter:

number (number)

Return Value

The NOTBITS function returns a number.

Examples

Return_value=NOTBITS(0)
Return_value=-1

Return_value=NOTBITS(1)
Return_value=-2

More information:

ANDBITS Function--Perform a Bitwise AND Operation (see page 493)
ORBITS Function--Perform a Bitwise OR Operation (see page 537)
XORBITS Function--Perform a Bitwise XOR Operation (see page 560)

NOW Function--Return Current Time in Seconds

The NOW function returns a numeric value representing the number of seconds since January 1, 1970 at the time the function is invoked. The time is local to the current server system.

This value is computed at the beginning of operations in a specific expression. Multiple references to the NOW function within the same expression always have the same result.

Syntax

The NOW function has the following format:

NOW()

Parameters

The NOW function accepts no parameters.

Return Value

The NOW function returns a number.

Example

Return_value=NOW()
Return_value=1024804800

More information:

NOWGMT Function--Return Current Time in Seconds (see page 535)

NOWGMT Function--Return Current Time in Seconds

The NOWGMT function returns a numeric value representing the number of seconds since January 1, 1970 at the time the function is invoked. The time is in the Greenwich (ZULU) time zone.

This value is computed at the beginning of operations in a specific expression. Multiple references to the NOWGMT function within the same expression always have the same result.

Syntax

The NOWGMT function has the following format:

NOWGMT()

Parameters

The NOWGMT function accepts no parameters.

Return Value

The NOWGMT function returns a number.

More information:

NOW Function--Return Current Time in Seconds (see page 535)

NUMBER Function--Convert to a Numeric Value

The NUMBER function converts its argument to a numeric value. Strings are converted up to the first non-digit. Booleans that have a value of TRUE are converted to 1; otherwise, they are converted to zero.

Syntax

The NUMBER function has the following format:

```
NUMBER(source_string | bool_val)
```

Parameters

The NUMBER function accepts either of the following parameters:

source_string (string)

bool_val (Boolean)

Return Value

The NUMBER function returns a number.

Example

```
Return_value=NUMBER('juan')
Return_value=0

Return_value=NUMBER('45juan')
Return_value=45

Return_value=NUMBER(TRUE)
Return_value=1
```

More information:

BOOLEAN Function--Convert to "TRUE" or "FALSE" (see page 497)
STRING Function--Convert to a String (see page 550)

ORBITS Function--Perform a Bitwise OR Operation

The ORBITS function performs a bitwise OR operation on its two arguments.

Syntax

The ORBITS function has the following format:

ORBITS(int_1, int_2)

Parameters

The ORBITS function accepts the following parameters:

int_1 (number)

int_2 (number)

Return Value

The ORBITS function returns a number.

Examples

Result_value=ORBITS(6, 1)
Result_value=7

Result_value=ORBITS(7, 8)
Result_value=15

More information:

ANDBITS Function--Perform a Bitwise AND Operation (see page 493)
NOTBITS Function--Perform a Bitwise NOT (see page 534)
XORBITS Function--Perform a Bitwise XOR Operation (see page 560)

PARENTDN Function--Retrieve Parent in LDAP Tree

The PARENTDN function returns the next level up in the LDAP Directory Information Tree above the specified distinguished name (DN). If the specified DN is invalid, or is already at the top of the tree, a blank string is returned.

Syntax

The PARENTDN function has the following format:

```
PARENTDN(source_string)
```

Parameters

The PARENTDN function accepts the following parameters:

source_string (string)

The LPAP distinguished name (DN).

Return Value

The PARENTDN function returns a string.

Remarks

LDAP Only: Yes

Example

```
Return_value=PARENTDN("uid=juan,o=NDS.com")
Return_value="o=NDS.com"

Return_value=PARENTDN("o=NDS.com")
Return_value=""
```

PCASE Function--Convert a String to Proper Case

The PCASE function converts the specified string to proper case (initial capital letters).

Syntax

The PCASE function has the following format:

```
PCASE(source_string)
```

Parameters

The PCASE function accepts the following parameters:

source_string (string)

Return Value

The PCASE function returns a string.

Example

Return_value=PCASE("framingham, mass")
Return_value="Framingham, Mass")

QS Function--Retrieve Items from a Query String

The QS function retrieves items from the query string associated with the resource being accessed by the user when the expression is evaluated.

If no arguments are supplied to this function, the entire query string (and only the query string) is returned. The query string is returned unchanged.

If the string argument is supplied as a blank string (“”), then all of the arguments in the query string that are unnamed are returned. If multiple values exist, they are returned as a set.

If the string argument is supplied as a non-blank string, all of the arguments in the query string that are named with a matching name are returned. Case-sensitivity is controlled by the optional Boolean flag. If multiple values exist, they are returned as a set.

Syntax

The QS function has the following format:

QS([input_string,][not_case_sensitive])

Parameters

The QS function accepts the following optional parameters:

input_string (string)

(Optional) Name of an argument in the query string.

not_case_sensitive (Boolean)

(Optional) If the *not_case_sensitive* flag is set to FALSE or omitted, the function searches the query string for an exact match. If the *not_case_sensitive* flag is set to TRUE, the function ignores case.

Return Value

The QS function returns a string.

Example

Assume this resource:
http://myserver.com/index.jsp?Test=A&X&TEST=D&c&Dbg

Return_value=QS()
Return_value='Test=A&X&TEST=D&c&Dbg'

Return_value=QS("")
Return_value='X^c'

Return_value=QS("Test")
Return_value= 'A^D'

Return_value=QS("Test", false)
Return_value= 'A'

"Dbg" IN QS("")
Return_value=TRUE

More information:

URL Function--Returns a Component of a URL String (see page 554)

RDN Function--Retrieve First Component of LDAP DN

The RDN function returns the first component of the specified LDAP Distinguished Name (DN). If the optional Boolean argument is TRUE (the default), the attribute name is removed and only the value is returned.

If the specified DN is invalid, a blank string is returned. This function does not call any LDAP server.

Syntax

The RDN function has the following format:

```
RDN(DN_string[, remove_name])
```

Parameters

The RDN function accepts the following parameters:

DN_string (string)

LDAP Distinguished Name

remove_name

(Optional) When set to TRUE (the default), the attribute name is removed from the returned string. When set to FALSE, the attribute is included in the returned string.

Return Value

The RDN function returns a string.

Remarks

LDAP Only: Yes

Example

```
Return_value=RDN("uid=juan,o=NDS.com")
Return_value="juan"

Return_value=RDN("uid=juan,o=NDS.com", TRUE)
Return_value="juan"

Return_value=RDN("uid=juan,o=NDS.com", FALSE)
Return_value="uid=juan"
```

More information:

COMMONDN Function--Find a Common Root (see page 501)
EXPLODEDN Function--Convert LDAP DN to Set (see page 513)
PARENTDN Function--Retrieve Parent in LDAP Tree (see page 538)
RELATIONDN Function--Compare Two Distinguished Names (see page 542)

RELATIONDN Function--Compare Two Distinguished Names

The RELATIONDN function compares the two specified LDAP distinguished names (DNs) and returns a string indicating the relationship between them.

If either of the two DNs is invalid, or the two DNs are completely unrelated, a blank string is returned.

If the two DNs are related, the difference in levels (of the Directory Information Tree) is returned as a string. If the first DN is the ancestor of the second, the number is positive. If the first DN is a descendent of the second, the number is negative. If the two DNs are equal or siblings, the return is 0 (indicating no levels).

Note: This function does not call any LDAP server functions.

Syntax

The RELATIONDN function has the following format:

```
RELATIONDN(dn_1, dn_2)
```

Parameters

The RELATIONDN function accepts the following parameters:

dn_1 (string)

dn_2 (string)

Return Value

The RELATIONDN function returns a string.

Remarks

LDAP Only: Yes

Example

```
Return_value=RELATIONDN("uid=eric,o=NDS.com", "o=NDS.com")
Return_value="-1"

Return_value=RELATIONDN("o=NDS.com", "uid=eric,o=NDS.com")
Return_value="1"

Return_value=RELATIONDN("uid=dave,o=NDS.com", "uid=eric,o=NDS.com")
Return_value="0"

Return_value=RELATIONDN("uid=dave,o=XYZ.com", "uid=eric,o=NDS.com")
Return_value=""
```

RIGHT Function--Retrieve Characters from a String

The RIGHT function returns the specified number of characters from the end of a string. If the string is shorter than the number, the entire string is returned.

Syntax

The RIGHT function has the following format:

```
RIGHT(source_string, length)
```

Parameters

The RIGHT function accepts the following parameters:

source_string (string)

length (number)

Number of characters to extract, counting from the end of the string.

Return Value

The RIGHT function returns a string.

Example

```
Return_value=RIGHT('JuanJuan', 2)
Return_value='an'

Return_value=RIGHT('JuanJuan', 10)
Return_value='JuanJuan'

Return_value=RIGHT('JuanJuan', 0)
Return_value=''
```

More information:

LEFT Function--Return Part of a String (see page 522)
MID Function--Return Part of a String (see page 530)

RPAD Function--Pad a String on the Right

The RPAD function adds the first character of the specified padding to the end of the string until the source string becomes the specified length.

If the padding is more than one character long, only the first character is used. If the padding is zero length, no padding is added.

If the source is a string, and the padding is not specified, a space is used for padding. If the source is a number and padding is not specified, a zero is used for padding.

Syntax

The RPAD function has the following format:

```
RPAD(source_string|number, length[, padding])
```

Parameters

The RPAD function accepts the following parameters:

source_string (string)

This parameter can be a number; it is converted to a string.

length (number)

padding (string)

(Optional)

Return Value

The RPAD function returns a string.

Example

```
Return_value=RPAD('Juan', 5)
Return_value='Juan '

Return_value=RPAD('Juan', 5, 'X')
Return_value='JuanX'

Return_value=RPAD('Juan', 6, 'XY')
Return_value='JuanXX'

Return_value=RPAD(5, 2)
Return_value='50'

Return_value=RPAD(5, 2, ' ')
Return_value='5 '
```

RPT Function--Repeat a String

The RPT function returns a string that repeats a source string the specified number of times.

Syntax

The RPT function has the following format:

```
RPT(source_string|number, repeat_count)
```

Parameters

The RPT function accepts the following parameters:

source_string (string)

This parameter can be a number; it is converted to a single character.

repeat_count (string)

Return Value

The RPT function returns a string.

Example

```
Return_value=RPT('Juan', 3)
Return_value='JuanJuanJuan')

Return_value=RPT('*', 10)
Return_value="**********')
```

RTRIM Function--Remove Trailing Spaces from a String

The RTRIM function eliminates trailing spaces from a source string and returns the result.

Syntax

The RTRIM function has the following format:

```
RTRIM(source_string)
```

Parameters

The RTRIM function accepts the following parameter:

source_string (string)

Return Value

The RTRIM function returns a string.

Example

```
Return_value=RTRIM(' JuanJuan  ' )
Return_value=' JuanJuan'
```

More information:

LTRIM Function--Remove Leading Spaces in a String (see page 527)

SECOND Function--Return the Number of Seconds in a Date

The SECOND function returns a value that represents the seconds component of a date expressed as the number of seconds since January 1, 1970.

Syntax

The SECOND function has the following format:

```
SECOND(date_time)
```

Parameters

The SECOND function accepts the following parameter:

date_time (number)

The number of seconds.

Return Value

The SECOND function returns a number from 0 to 59.

More information:

MINUTE Function--Return the Minutes Component for a Date (see page 532)

SET Function--Set the Value of an Attribute

The SET function assigns a specified value to a specified attribute. Multiple values are specified for multi-valued attributes by a set. This function works for all User Directories supported by SOA Security Manager.

The SET function returns TRUE if SOA Security Manager returns success on the modification.

If the attribute is not visible to SOA Security Manager, the function fails. The attribute may not be visible due to security reasons (security in the User Directory) or, in the case of ODBC directories, because it is not in the configured Query or not an attribute listed in the Set Properties setting of the SOA Security Manager Query Scheme.

Syntax

The SET function has the following format:

```
SET(attr_name, value)
```

Parameters

The SET function accepts the following parameters:

attr_name (string)

value (string)

Specifies one or more values. Multiple values are separated by the caret character.

Return Value

The SET function returns a Boolean.

Remarks

Privileged: Yes

Example

Return_value=SET("Retries", STRING(NUMBER(Retries) + 1))

More information:

GET Function--Locate Attributes in a User Directory (see page 517)

SIGN Function--Return the Sign of a Number

The SIGN function accepts a number. If the number is negative, SIGN returns a negative one. If the number is zero, SIGN returns a zero. If the number is positive, SIGN returns a positive one.

Syntax

The SIGN function has the following format:

SIGN(number)

Parameters

The SIGN function accepts the following parameter:

number (number)

Return Value

The SIGN function returns a number: -1, 0, or +1.

Example

Return_value=SIGN(-40)
Return_value=-1

Return_value=SIGN(0)
Return_value=0

Return_value=SIGN(999)
Return_value=1

SORT Function--Sort a Set

The SORT function sorts a specified set. Case sensitivity is specified by an optional Boolean parameter. Duplicate items are eliminated from the resulting set.

Syntax

The SORT function has the following format:

```
SORT(source_set[, not_case_sensitive]
```

Parameters

The SORT function accepts the following parameters:

source_set (string)

Set to be sorted.

not_case_sensitive (Boolean)

(Optional) If this parameter is omitted or set to FALSE, the sort treats entries as identical unless the cases are different. If this parameter is set to TRUE, the sort ignores case.

Return Value

The SORT function returns a set.

Examples

```
Return_value=SORT("Eric^Bart^Chuck^BART^Chuck")
Return_value="BART^Bart^Chuck^Eric"

Return_value=SORT("Eric^Bart^Chuck^BART^Chuck", FALSE)
Return_value="BART^Bart^Chuck^Eric"

Return_value=SORT("Eric^Bart^Chuck^BART^Chuck", TRUE)
Return_value="Bart^Chuck^Eric"
```

SPACE Function--Return a String of Spaces

The SPACE function returns a string that consists of the specified number of spaces.

Syntax

The SPACE function has the following format:

```
SPACE(repeat_count)
```

Parameters

The SPACE function accepts the following parameters:

repeat_count (number)

The number of spaces to include in the string.

Return Value

The SPACE function returns a string.

Example

```
Return_value=SPACE(3)  Return_value='   '

Return_value=SPACE(10) Return_value="          "
```

STRING Function--Convert to a String

The STRING function converts a number or Boolean into a string value.

Syntax

The STRING function has the following format:

```
STRING(num_value|bool_value)
```

Parameters

The STRING function accepts either one of these parameters:

num_value (number)

bool_value (Boolean)

Return Value

The STRING function returns a string.

Example

Return_value=STRING(TRUEVAL)
Return_value="TRUE"

Return_value=STRING(123)
Return_value='123'

More information:

BOOLEAN Function--Convert to "TRUE" or "FALSE" (see page 497)
NUMBER Function--Convert to a Numeric Value (see page 536)

THROW Function--Stop Processing and Report Custom Error

You can write an expression that tests for an error and if an error has occurred, calls THROW, passing a custom error message. When the expression evaluator encounters THROW, it stops processing the expression and outputs the custom error message to the console log.

Syntax

The THROW function has the following format:

THROW(error_message)

Parameters

The THROW function accepts the following parameter:

error_message (string)

Specifies the custom error message that is output to the console log.

Return Value

The THROW function does not return.

Example

EXISTS('MyFile') ? #Process('MyFile') : THROW('File does not exist.')
Message Output to the Console Log: 'File does not exist.'

VEXIST(#Sortname) ? #Sortname : THROW('Sortname is not defined.')
Message Output to the Console Log: 'Sortname is not defined.'

TRACE Function--Write Trace Entry to Console Log

The TRACE function writes the string argument to the SOA Security Manager Console Log as a trace entry.

Syntax

The TRACE function has the following format:

```
TRACE(source_string)
```

Parameters

The TRACE function accepts the following parameter:

source_string (string)

Return Value

The TRACE function returns a Boolean, always TRUE.

Example

```
Return_value=TRACE("Executing Code")
Return_value=TRUE
```

More information:

ERROR Function--Write Error Message to Console Log (see page 511)
INFO Function--Write INFO Message to Console Log (see page 520)
TRACE Function--Write Trace Entry to Console Log (see page 559)

TRANSLATE Function--Replace String Value

The TRANSLATE function replaces all occurrences of one string found within a second string with a third string. The search is case-sensitive unless the optional Boolean is set to TRUE.

Syntax

The TRANSLATE function has the following format:

```
TRANSLATE(source_string, search_string, replace_string[, not_case_sensitive])
```

Parameters

The TRANSLATE function accepts the following parameters:

source_string (string)

search_string (string)

replace_string (string)

not_case_sensitive (Boolean)

(Optional) If this parameter is not set or set to FALSE, case is considered in the search. If set to TRUE, case is ignored.

Return Value

The TRANSLATE function returns a string.

Example

```
Return_value=TRANSLATE('Eric','r','x')
Return_value='Exic'

Return_value=TRANSLATE('Eric','ri','x')
Return_value='Exc'

Return_value=TRANSLATE('Eric','r','xy')
Return_value='Exyic'

Return_value=TRANSLATE('Eric','R','x')
Return_value='Eric'

Return_value=TRANSLATE('Eric','R','x',TRUE)
Return_value= 'Exic'
```

UCASE Function--Convert to Upper Case

The UCASE function converts the source string to upper case.

Syntax

The UCASE function has the following format:

```
UCASE(source_string)
```

Parameters

The UCASE function accepts the following parameter:

source_string (string)

Return Value

The UCASE function returns a string.

Example

Return_value=UCASE('framingham, mass')
Return_value='FRAMINGHAM, MASS'

More information:

LCASE Function--Convert to Lowercase (see page 522)
PCASE Function--Convert a String to Proper Case (see page 538)

URL Function--Returns a Component of a URL String

The URL function parses the supplied URL (or path) and returns the specified component. This function ignores characters that are URL-encoded.

Note: The URL function only parses strings that use slashes, not backslashes. If you are using a system running Windows, use the TRANSLATE function to convert backslashes to slashes.

Syntax

The URL function has the following format:

```
URL(url_string, component)
```

Parameters

The URL function accepts the following parameters:

url_string (string)

The URL or path must be specified following this format:

[<protocol>:][//][<user>[:<password>]@]<server>[:<port#>] [[/<directory>]/[<file>]][?<querystring>]

component (string)

Component names are not case-sensitive. You can specify any of the following components:

- PROTOCOL, DRIVE, or NAMESPACE
- USER
- PASSWORD
- SERVER
- PORT
- DOMAIN
- URI
- PATH
- DIRECTORY
- FILENAME
- BASENAME
- EXTENSION
- QUERYSTRING

Return Value

The URL function returns a string.

Examples

Return_value=URL("http://www.myserver.com:8080/app1/xyzzy.jsp?id=12", "PROTOCOL")
Return_value="http:"

Return_value=URL("ftp://joe:dog@ftp.myserver.com/dir1/xyzzy.zip", "USER")
Return_value="joe"

Return_value=URL("ftp://joe:dog@ftp.myserver.com/dir1/xyzzy.zip", "PASSWORD")
Return_value="dog"

```
Return_value=URL("http://www.myserver.com:8080/app1/xyzzy.jsp?id=12",
"SERVER")
Return_value="www.myserver.com"

Return_value=URL("http://www.myserver.com:8080/app1/xyzzy.jsp?id=12",
"PORT")
Return_value="8080"

Return_value=URL("http://www.myserver.com:8080/app1/xyzzy.jsp?id=12",
"DOMAIN")
Return_value="myserver.com"

Return_value=URL("ftp://joe:dog@ftp.myserver.com/dir1/xyzzy.zip", "URI")
Return_value="/dir1/xyzzy.zip"

Return_value=URL("ftp://joe:dog@ftp.myserver.com/dir1/xyzzy.zip", "PATH")
Return_value="/dir1/xyzzy.zip"

Return_value=URL("ftp://joe:dog@ftp.myserver.com/dir1/xyzzy.zip",
"DIRECTORY")
Return_value="/dir1"

Return_value=URL("ftp://joe:dog@ftp.myserver.com/dir1/xyzzy.zip",
"FILENAME")
Return_value="xyzzy.zip"

Return_value=URL("ftp://joe:dog@ftp.myserver.com/dir1/xyzzy.zip",
"BASENAME")
Return_value="xyzzy"

Return_value=URL("ftp://joe:dog@ftp.myserver.com/dir1/xyzzy.zip",
"EXTENSION")
Return_value="zip"

Return_value=URL("ftp://joe:dog@ftp.myserver.com/dir1/xyzzy.zip",
"QUERYSTRING")
Return_value=""

Return_value=URL("http://www.myserver.com:8080/app1/xyzzy.jsp?id=12",
"QUERYSTRING")
Return_value="id=12"
```

URLDECODE Function--Decode a URL String

The URLDECODE function decodes the specified URL string.

Syntax

The URLDECODE function has the following format:

URLDECODE(url)

Parameters

The function accepts the following parameter:

url (string)

Return Value

The URLDECODE function returns a string.

Example

Return_value=URLDECODE("Framingham%2C+Mass")
Return_value="Framingham, Mass"

More information:

URLENCODE Function--Encode a String (see page 557)

URLENCODE Function--Encode a String

The URLENCODE function encodes the passed-in string to URL format.

Syntax

The URLENCODE function has the following format:

URLENCODE(source_string)

Parameters

The URLENCODE function accepts the following parameter:

source_string (string)

Return Value

The URLENCODE function returns a string.

Example

Return_value=URLENCODE('Framingham, Mass')
Return_value='Waltham%2C+Mass'

More information:

URLDECODE Function--Decode a URL String (see page 557)

VEXIST Function--Is the Parameter Defined?

The VEXIST function accepts a named expression, a context variable, or user attribute and determines whether it is defined. If defined, VEXIST returns TRUE. If not, VEXIST returns FALSE.

Note: In the case of LDAP user directories, SOA Security Manager cannot determine whether a user attribute is defined and returns FALSE.

Syntax

The VEXIST function has the following format:

VEXIST(#virtual_user_attribute | @user_class | %context_variable | user_attribute_name | user_attribute_string)

Parameters

The VEXIST function accepts one of the following parameters:

#virtual_user_attribute (named expression)

Specifies a named expression that calculates a user attribute.

@user_class (named expression)

Specifies a named expression that tests for membership in a user directory or group.

%context_variable (context variable)

Specifies a context variable.

user_attribute_name (unquoted string)

Specifies a single user attribute.

user_attributes_string (string)

Specifies a string of user attribute names separated by a character.

Return Value

The VEXIST function returns a Boolean.

Example

```
Return_value=VEXIST(#Age)
Return_value=TRUE

Return_value=VEXIST(@IsDolphin)
Return_value=FALSE

Return_value=VEXIST(%ClientIP)
Return_value=TRUE

Return_value=VEXIST(givenname)
Return_value=FALSE

Return_value=VEXIST('last,first')
Return_value=TRUE
```

TRACE Function--Write Trace Entry to Console Log

The WARNING function writes the string argument to the SOA Security Manager Console Log as a warning message.

Syntax

The WARNING function has the following format:

```
WARNING(source_string)
```

Parameters

The WARNING function accepts the following parameter:

source_string (string)

Return Value

The WARNING function returns a Boolean, always TRUE.

Example

```
Return_value=WARNING("Buffer Full")
Return_value=TRUE
```

XORBITS Function--Perform a Bitwise XOR Operation

The XORBITS function performs a bitwise XOR operation on its two arguments.

Syntax

The XORBITS function has the following format:

XORBITS(num_1,num_2)

Parameters

The XORBITS function accepts the following parameters:

num_1 (number)

num_2 (number)

Return Value

The XORBITS function returns a number.

Example

Return_value=XORBITS(7, 2)
Return_value=5

Return_value=XORBITS(7, 15)
Return_value=8

More information:

ANDBITS Function--Perform a Bitwise AND Operation (see page 493)
NOTBITS Function--Perform a Bitwise NOT (see page 534)
ORBITS Function--Perform a Bitwise OR Operation (see page 537)

YEAR Function--Return the Year Component of a Numeric Date

The YEAR function returns a number that indicates the year component of a date in seconds since January 1, 1970.

Syntax

The YEAR function has the following format:

YEAR(date_time)

Parameters

The YEAR function accepts the following parameter:

date_time (number)

The date represented in seconds.

Return Value

The YEAR function returns a number between 70 and 138.

YEAR4 Function--Return the Year Component of a Date (4 digits)

The YEAR4 function returns the four-digit year component of a date expressed in seconds since January 1, 1970.

Syntax

The YEAR4 function has the following format:

```
YEAR4(date_time)
```

Parameters

The YEAR4 function accepts the following parameters:

date_time (number)

Return Value

The YEAR4 function returns a returns a number between 1970 and 2038.

Index

L

Load WSDL files • 40, 305, 308

S

Secure web services from WSDL • 40, 305, 308

W

WSDL file, secure web services from • 40, 305, 308