CA SiteMinder®

Policy Server Configuration Guide

r12.5

Second Edition

CA Technologies Product References

This document references the following CA Technologies products:

- CA DLP
- CA Single Sign–On
- CA SiteMinder®
- CA Enterprise Log Manager

Contact CA Technologies

Contact CA Support

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

- 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

Providing Feedback About Product Documentation

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

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following updates have been made since the last release of this documentation:

Topics Added

-

Topics Updated

- External Administrator Store Considerations (see page 71)–The reference to the SiteMinder object export utility (smobjexport) was removed.
- Asynchronous Call Support During Failover and Connection Pooling (see page 238)–Updated the Oralce versions that support asynchronous calls.
- Active Directory Considerations (see page 198)—Removed incorrect note regarding the IgnoreDefaultRedirectOnADnativeDisabled registry setting. This update resolves CQ170141 and STAR Issue # 21243447:03

Contents

Chapter 5: SiteMinder Administrators 67

Chapter 25: Password Policies 667

Chapter 26: SiteMinder Test Tool 683

Appendix A: Troubleshooting SSL Authentication Schemes 703

Appendix B: LanMan User Directories 713

Appendix C: CA SSO/WAC Integration 717

Appendix E: Attributes and Expressions Reference 769

Chapter 1: SiteMinder Overview

This section contains the following topics:

SiteMinder Components

SiteMinder consists of two core components:

Policy Server

The Policy Server provides policy management, authentication, authorization, and accounting.

SiteMinder Agents

Integrated with a standard Web server or application server, SiteMinder Agents enable SiteMinder to manage access to Web applications and content according to predefined security policies. Other types of SiteMinder Agents allow SiteMinder to control access to non-Web entities. For example, a SiteMinder RADIUS Agent manages access to RADIUS devices, while a SiteMinder Affiliate Agent manages information passed to an affiliate's Web site from a portal site.

Policy Server Overview

The Policy Server typically runs on a separate Windows or Solaris system to perform SiteMinder's key security operations. The Policy Server provides the following:

Authentication

The Policy Server supports a range of authentication methods. It can authenticate users based on user names and passwords, via tokens, using forms based authentication, and through public-key certificates.

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 SiteMinder Administrative UI. The Administration service of the Policy Server is what allows the Administrative UI to record configuration information in the Policy Store.

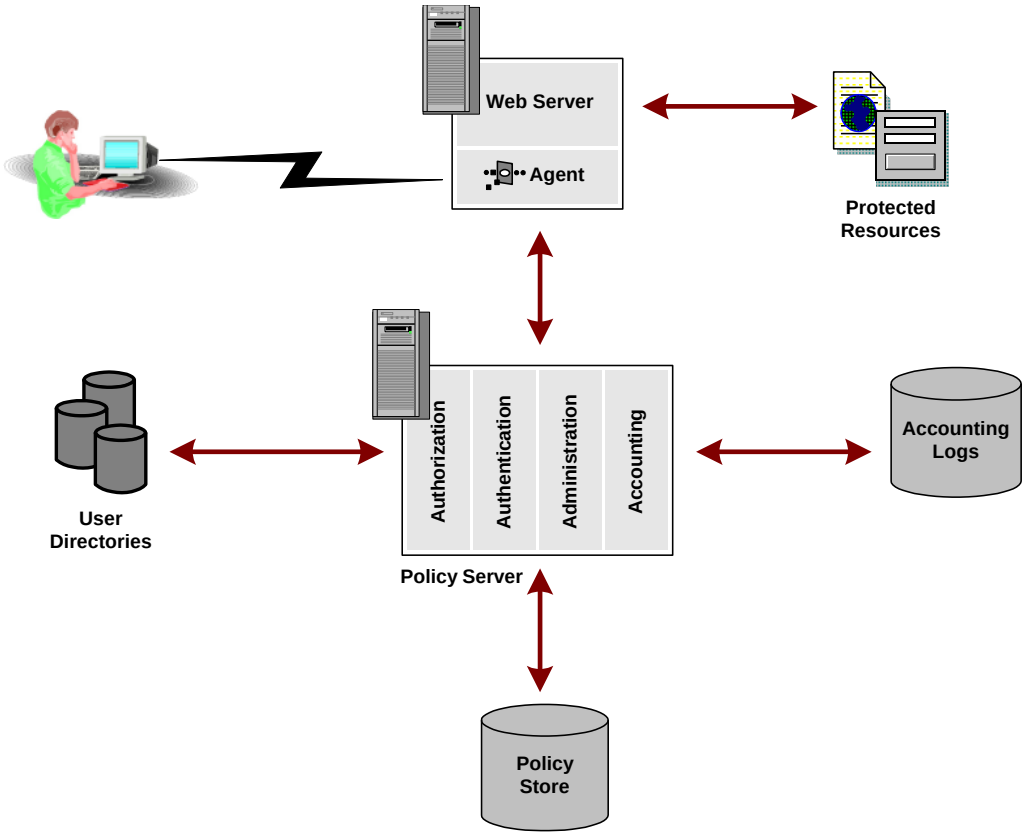
Accounting

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 SiteMinder deployment.

The following figure illustrates a simple SiteMinder environment.



In a Web implementation, a user requests a resource through a browser. That request is received by the Web Server and intercepted by the SiteMinder Web Agent. The Web Agent determines whether or not the resource is protected, and if so, gathers the user's credentials 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.

Note: Other types of Agents can be created using the Agent API.

More information:

Custom Agents (see page 152)

Policy Server Management Console Overview

The majority of Policy Server configuration tasks are performed using the Administrative UI. However, there are 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*.

Chapter 2: Policy Server Object Overview

This section contains the following topics:

Policy Server Object Types

There are three main categories of objects that the policy server uses:

- Infrastructure Objects
- Policy Domain Objects
- Global Objects (see page 34)

Infrastructure Objects

You use infrastructure objects throughout a SiteMinder deployment. Infrastructure 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 web servers, application servers, or other network entities to secure access to resources. Once an Agent is installed on a server, configure a SiteMinder 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 SiteMinder for groups of servers that distribute the workload for access to a common 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 SiteMinder is an object that contains details for connecting to an existing user directory that is external to SiteMinder. User directory connections let you configure a connection to an existing user directory, instead of replicating user information within SiteMinder.

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 group of resources (realm), and the users who can access the resources, and the administrator who sets up entitlements.

Affiliate Domains

Affiliate domains are only for legacy federation. An affiliate domain is a logical grouping of SAML affiliates that is associated with one or more user directories and administrators.

Note: For more information about affiliate domains, see the *Federation Manager Guide: Legacy Federation*.

Administrators

An administrator is an object that contains profile information for a SiteMinder administrator account. Everyone who logs in to SiteMinder 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 that a user requires to access a protected resource. Authentication schemes are assigned to realms or Applications (see page 461). When a user tries to access a resource in a realm or Application, the assigned authentication scheme determines the credentials that a user must supply 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 that a type of Agent supports. For example, Web, Affiliate, RADIUS, or custom.

SQL Query Schemes

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

Password Policies

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

SAML Affiliations

SAML affiliations are only for legacy federation. A SAML affiliation is a group of SAML 2.0 entities that share a name identifier for a single principal.

Note: Note: For more information about SAML affiliations, see the *Federation Manager Guide: Legacy Federation*.

Trusted Hosts

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

Policy Domain Objects

A policy domain is a group of objects that deal with a specific domain of resources. For example, a company may divide its network resources by business unit, creating one policy domain for marketing, another policy domain for engineering, and so on. Policy domain objects are those objects that pertain to a specific policy domain. These objects include rules and policies for controlling access to resources.

Policy domain objects include:

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 are allowed or denied for the resource. Rules can also include actions that are associated with specific events. For example, 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 are 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 are 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 SiteMinder protected entity. The policy is what SiteMinder ultimately uses to resolve the request when a user attempts to access a resource.

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.

Affiliates

Affiliate objects are only for legacy federation. An affiliate object binds users, and optionally, time restrictions and IP address restrictions together. An affiliate object also contains configuration data and a list of user entitlement attributes to be passed to an affiliate after a user is authenticated.

Note: For more information about affiliates, see the *Federation Manager Guide: Legacy Federation*.

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

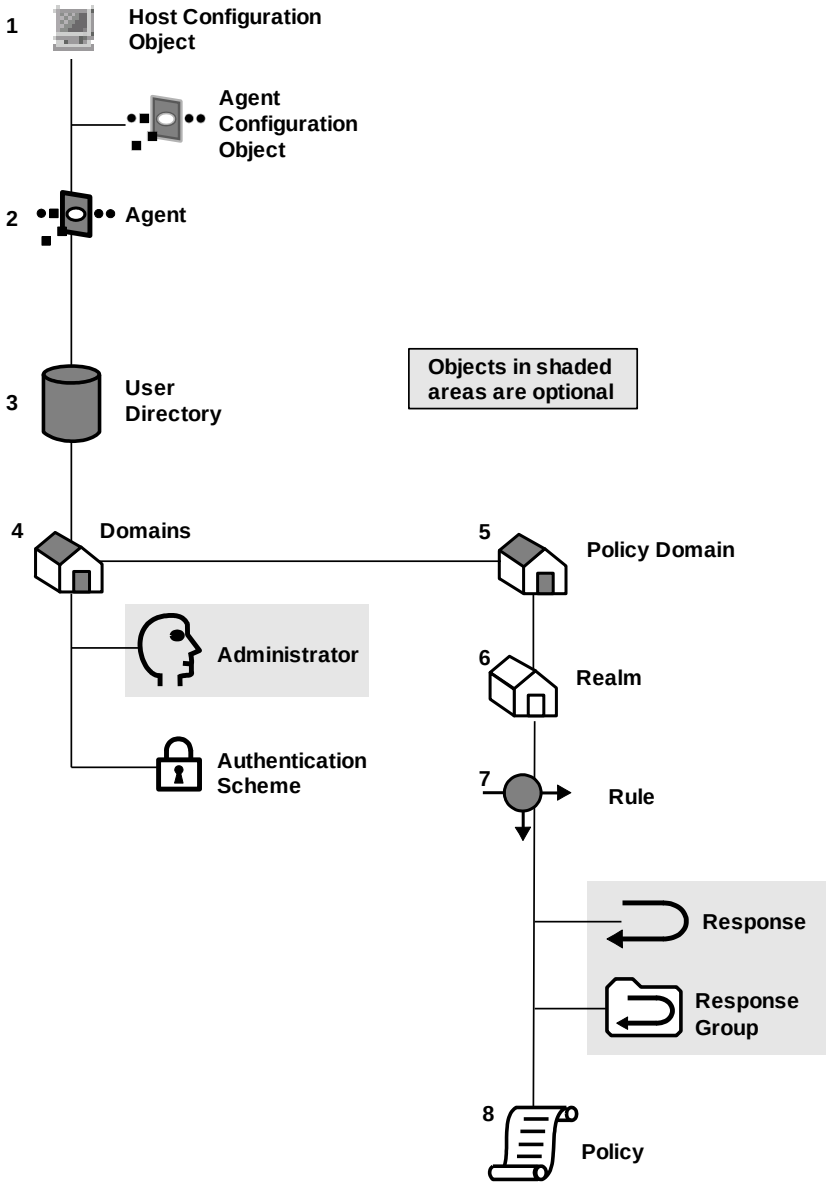
A global response is a Policy Server object that determines a reaction to a global rule. Global responses are included in global policies, and take place when a global rule is triggered.

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 SiteMinder ultimately uses to resolve the request.

Configuration Order

A SiteMinder deployment requires that you configure core SiteMinder objects in a specific order. The following figure illustrates the configuration order for the core SiteMinder objects:



Chapter 3: Implementing Policy-based Security

This section contains the following topics:

Policy-based Security Overview

Managing users and securing resources successfully are critical aspects of administering an application, a Web site, or a portal. To manage users and secure resources effectively, you must:

- Provide users with easy and fast access to appropriate information.
- Provide security users can trust.
- Protect resources from users who are not allowed access.

Failing to manage users and secure resources effectively can have negative effects on you and your clients, such as:

- Jeopardizing resources if security is inadequate.
- Losing sales if potential customers lack confidence in the security of your site and avoid the site.
- Frustrating customers if the process of accessing resources is cumbersome.
- Losing consumer data if you do not track users adequately.

Developing and implementing a strategy that secures your resources and also manages your user base is critical when you build a Web site or Web application.

Strategies for Managing Security and Users

A complete security solution should answer the following questions:

- Where is security needed and how much security is needed?
- What kind of tasks will be performed? Security-related tasks?
- What resources need to be protected and what level of protection is required?

- Who has access to the resource and what are they authorized to do?
- How is control over user access to protected resources enforced?

Two of the most common methods for managing users and securing resources are access control lists (ACLs) and security policies.

Security policies provide the most complete security solution by defining not only the type of access a user or user group has to a resource but also what happens when a user or user group accesses the resource. Security policies go beyond the capabilities of ACLs by enabling you to manage the user experience. The SiteMinder authorization model is based on security policies.

Access Control Lists

An access control list is an object associated with a resource that defines access privileges for individual users or groups of users. ACLs are associated with resources to establish:

- Who can access a resource
- What type of access the user has

ACLs provide a straightforward way of granting or denying a specified user or groups of users access to a resource. For example:

```
{Manager:ALL, Clerk:READ, Others:NONE}
```

The access control list above assigns users in the manager group complete access, users in the clerk group read-only access, and users in all other groups no access to the resource.

Several drawbacks are associated with ACLs:

- Controlling a user's interaction with a resource once the user is authorized and authenticated is difficult. For example, you cannot use ACLs to define session timeouts.
- Managing ACLs for large numbers of resources can be prohibitively time consuming and costly. ACLs create a significant administrative burden.
- Profiling user information is difficult. Users are added to ACLs, which are then associated with resources, making it difficult to determine all of the resources a user has access to at any given moment.

- Qualifying access rights is difficult. It's virtually impossible to restrict access based on a time or a location.
- Personalizing content and defining responses is difficult. If a user has access to a resource, it's difficult to personalize the content or define what happens when the user is allowed or denied access. For instance, you can't use an ACL to display or hide a set of buttons when the user accesses the resource.

ACLs are an effective way to protect a resource but an ineffective way to manage the user experience.

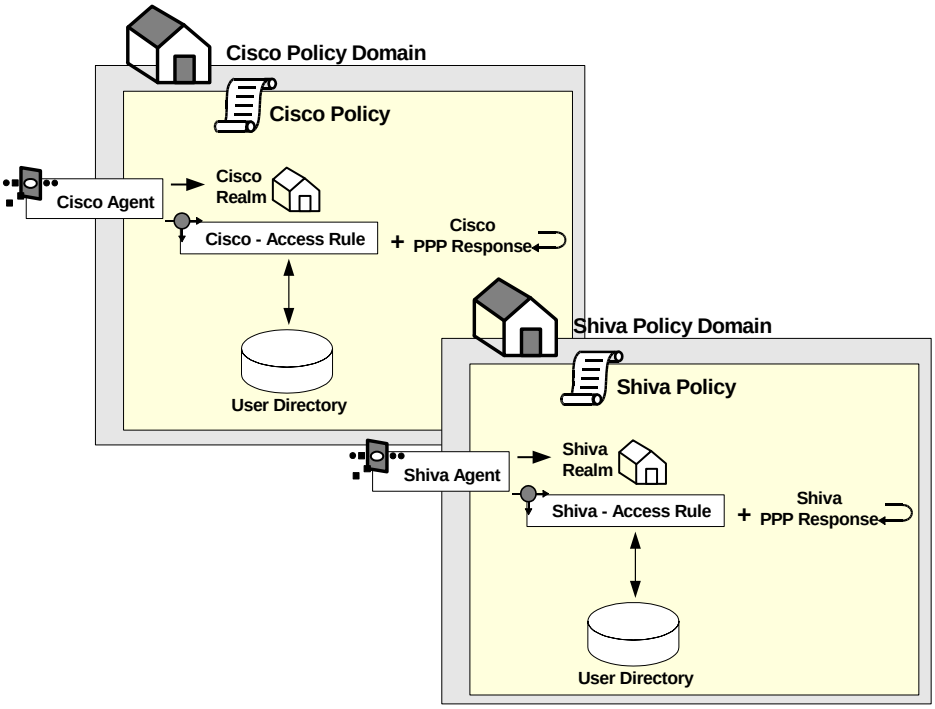
SiteMinder Security Policies

Unlike ACLs, policies serve a dual purpose: policies provide security and manage the user experience. Policies are user-centric: policies are constructed around the user group rather than the resource.

Policies define access permissions using rules, responses, and time/location constraints. Policies are then associated with users or user groups to establish:

- Where the resource is located
- Who can access a resource
- What type of access the user has
- When they can access a resource
- What happens when they access the resource
- What happens if they can't access the resource

The following graphic provides a definition of a SiteMinder security policy.



Policies provide an effective means of managing users and securing resources for the following reasons:

- Policies provide more granularity and the ability to personalize content. Responses enable you to define what happens when a user is allowed or denied access, such as which graphics are shown if a user is allowed access or where to redirect the user if the user is denied access.
- Policies are easy to maintain. When a user is modified, added to the group, or deleted from a group, all policy definitions that include the user are automatically updated.
- Policies provide fine-grained security. Policy definitions can include time restrictions and location (I.P.) restrictions.

Because of the power and flexibility of policies, authorization models based on security policies are more efficient and effective than models based on ACLs.

Manage the End-user Experience

In addition to securing resources, policies in SiteMinder can be used to manage the end-user experience by:

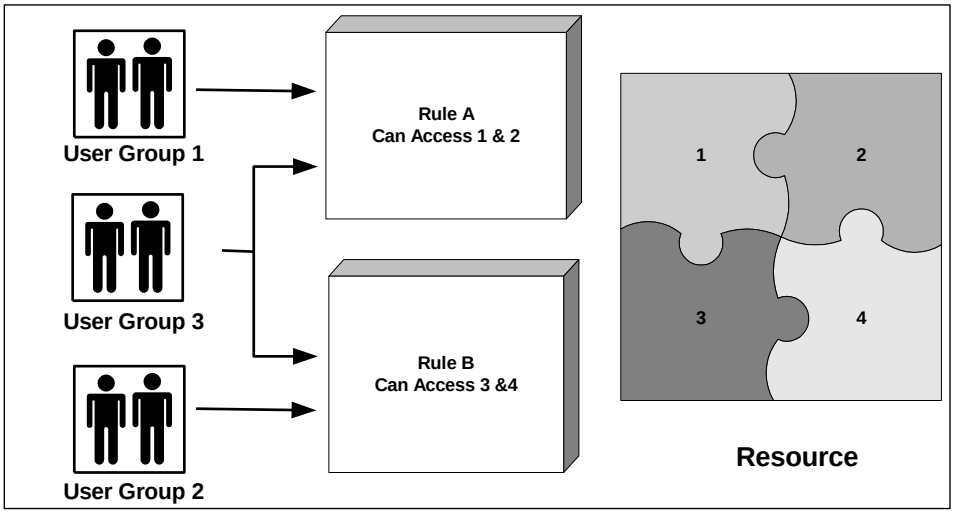
- How Privileges Are Established (see page 41)
- How Content Is Personalized (see page 41)
- How Sessions Are Managed (see page 42)

How Privileges Are Established

In a policy, the privilege to access a resource is established by adding a rule to a policy. Rules identify specific resources and either allow or deny the user access to the resources. A single policy can establish privileges for many users: policies can be associated with an individual user, a user group, or the members of an entire user directory.

For example, in the following graphic:

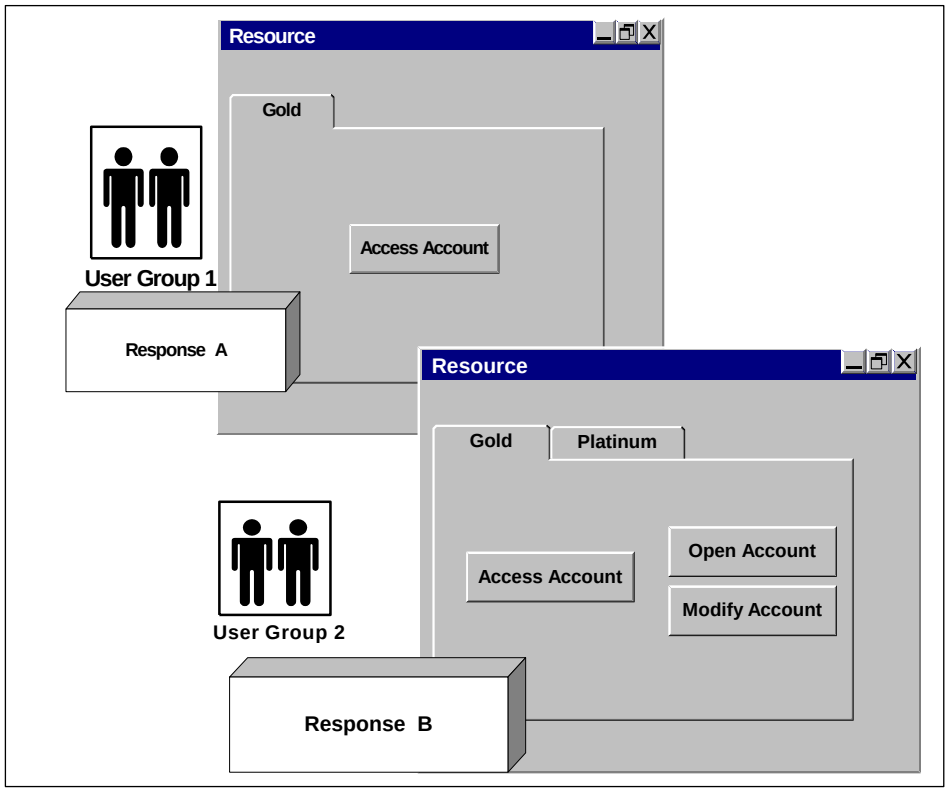
- Group 1 has been assigned Rule A and can access resources 1 & 2,
- Group 2 has been assigned Rule B and can access resources 3 & 4, and
- Group 3 has been assigned both rules and can access all resources.



How Content Is Personalized

Once a user has been granted access to a resource, the policy can then personalize the user's view of the resource. Policies can customize the view of a resource through the use of responses. Responses are paired with rules and are triggered when a rule fires.

For example, in the following graphic, both Group 1 and Group 2 can access the Resource dialog. However, the view each group has of the dialog differs. The policy for Group 1 uses Response A, which does not provide two buttons (Open Account and Modify Account) or the Platinum tab that Response B makes available.



How Sessions Are Managed

By managing sessions, you control how long an authenticated and authorized user can access the resource. You can control sessions by:

- Specifying the amount of time a user can remain idle, without interacting with the resource.

  Idle timeouts protect against unauthorized use of the resource by limiting the amount of time the session remains active if it is not being used. The idle timeout is particularly useful in cases where users leave their computer without logging out of their session. When the idle timeout limit is reached, the session automatically ends.

- Specifying the maximum amount of time a user can access a resource before they must re-authenticate.

  Maximum timeouts protect against unauthorized use of a resource by forcing an authenticated user to re-authenticate after a specified time. This safeguard ensures that if an authenticated user leaves the computer without logging out and someone else uses the open session, the session will end after a specified amount of time, and the user must re-authenticate to continue using the resource.

- Revoking a user's access to a resource at any time.

  In addition to managing how long a session can remain active, you can also end a session immediately if you suspect the integrity of a resource has been compromised. Once a user session has been revoked, the user is disabled in the user directory until you have re-enabled the user using the Administrative UI.

  Note: More information about managing sessions exists in the *Administration Guide*.

- Enabling persistent sessions.

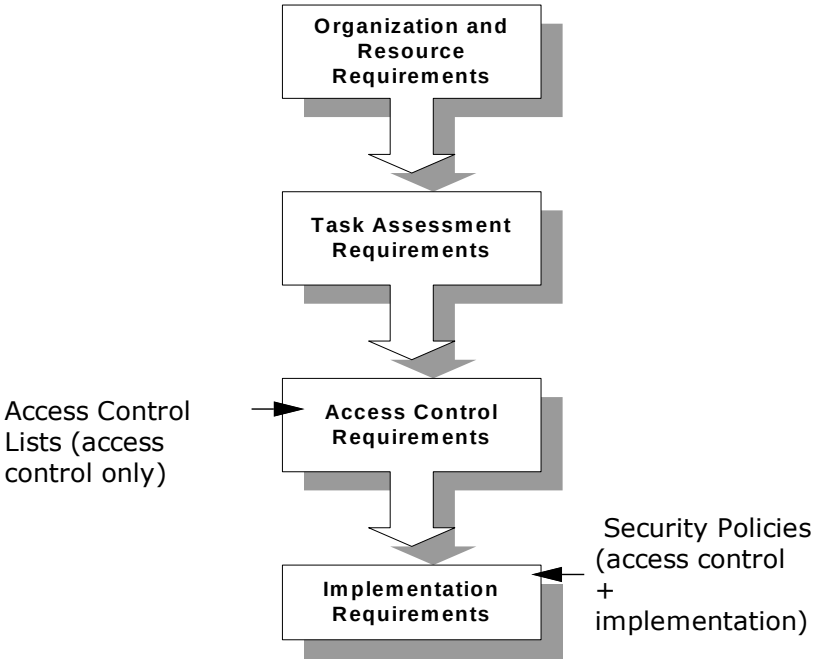
  You can also implement persistent sessions to provide Windows security context functionality and support for Federated Web Services.

More information:

How Privileges Are Established (see page 41)
How Content Is Personalized (see page 41)
How Persistent Sessions for User Security Contexts Are Maintained (see page 112)

Security Model Implementation

To implement a security model that best meets the needs of your organization, you may create security policies using information gathered in the design phases shown and described below.



1. Organization and resource requirements—set the basic objective of the security model and identify the resources.
2. Task assessment requirements—identify users and roles, and link the roles to tasks.
3. Access control requirements—establish access requirements for users based on their role requirements.

   Authorization models based only on access control lists (ACLs) end at this point.
4. Implementation requirements—define how the access is implemented (in terms of how users are tracked and how content is personalized for users) and how user sessions are managed.

   Authorization models based on SiteMinder security policies incorporate both access control and implementation models.

More information:

Organization and Resource Requirement Considerations (see page 45)
Define Task-Assessment Requirements (see page 47)

Organize Security Model Requirements

When completing each phase of the security model design methodology, use a table similar to the one shown below to organize your information. Once you have completed the columns in this table, you can use the information to build SiteMinder security policies.

Resource	Role	Task	Access	Implementation

Organization and Resource Requirement Considerations

Before you can setup a security model, you must establish the organization and resource needs of your organization. Some general issues to consider include:

- Are there security guidelines, regulations, or laws your organization is required to meet?
- If security needs conflict with business requirements, which requirements take precedence?
- Has a lack of security affected the reputation of your organization?
- Have security incidents in the past caused financial loss or taken down the site?
- Once you have established the overall security requirements and concerns, you are ready to define the more specific security requirements outlined below.

How Organization Security Requirements Are Defined

To determine the more specific security needs, consider the following issues:

These organization requirements:	Affect these tasks:
■ Who requires access to the resources? ■ How much access do they require? ■ Can you categorize users with similar access requirements into groups?	Configuring user directory connections
■ Which resources require protection? ■ Do different resources require different levels of protection?	Creating policy domains and realms
■ How sensitive and valuable is the information? ■ How much do you trust your users? ■ Are your users local or remote? ■ What type of security do your users expect? ■ Will you lose customers if security does not match their expectations?	Creating authentication schemes using authentication templates
■ Are there security guidelines, regulations, or laws your organization is required to meet? ■ Do different objects require fine-grained protection or personalization? ■ What type of actions do you want to control?	Defining rules
■ What type of security and controls do your users and customers expect? ■ Do different groups of users require different views of the resource? ■ What events should take place when a user is authenticated or authorized?	Defining responses
■ How will you implement your requirements?	Defining policies

Identify Resources and Roles

The second part of establishing organization requirements is to identify resources and map resources to roles.

The purpose of this step is to link resources with roles. By linking these two components, you will have a better understanding of what needs protection and what type of protection is required.

When identifying resources:

- Identify all known resources, including resources that are planned but do not yet exist. Planning security for all known resources at once, whether they currently exist or not, will save you time.
- Create a site map for Web sites to better understand the structure of the resources.

How this applies to policies:

Resources are defined in realms and rules. Roles of users are implied based on the user group to which they belong or based on their user attributes. In an airline application, for example, a user belonging to the Pilot user group would perform tasks associated with the Pilot role.

To identify resources and roles

1. Using a table or chart similar to the security model table described earlier in this chapter, list the resources in the Resources column.
2. Identify all subdivisions of a single resource. For example, if a directory called /bidding had two subdirectories below, such as /bidding/flights and /bidding/standby, both subdirectories would be listed as resources. By treating each subdirectory as a separate resource, it will be easier to determine if each resource requires separate security.
3. Next to each resource, list the roles that will need access to the resource.

Define Task-Assessment Requirements

Task assessment requirements bridge the gap between roles and access-control requirements. When you identify task-assessment requirements, you define the tasks each role performs using the resource.

How this applies to policies:

Although tasks are not specifically defined in policies, you will use this information when assessing access rights for each resource-role pairing in the next section.

To define task-assessment requirements

1. For each role, identify the tasks the role must perform using the resource.
2. Enter the task in the Task column, next to the associated resource and role.

More information:

Define Access Control Requirements (see page 48)
Organize Security Model Requirements (see page 45)
Organization and Resource Requirement Considerations (see page 45)
Define Implementation Requirements (see page 48)

Define Access Control Requirements

Access control requirements establish whether or not a role requires access to a resource, and if so, the type of access they require. Two roles that access the same resource may not require the same access to the resource.

For example, two groups of users might perform tasks in the same directory. Group A might use this resource to view and modify the resource, while Group B members would view the resource. Because each user group uses the resource in a different way, access rights would differ:

- Group A: Get, Post
- Group B: Get

How this applies to policies:

Access rights are defined in rules.

To define access control requirements

1. For each task, identify the type of access that is required to perform the associated task.
2. Enter the access requirement in the Access column.

Define Implementation Requirements

Implementation requirements define how the resource is used. How a resource is used can be configured by many variables, including:

- Personalization
- Time limitations for using the resource
- Redirections

How this applies to policies:

Implementation requirements are defined in responses.

To define implementation requirements

1. For each task, identify how access should be implemented.
2. Enter the implementation requirement in the Implementation column.

With the security model information complete, you can begin constructing policies that are appropriate for your site.

More information:

Organize Security Model Requirements (see page 45)
Organization and Resource Requirement Considerations (see page 45)
Define Task-Assessment Requirements (see page 47)

SiteMinder Application Roles

Application roles let you specify a group of users for access control based on your organization's business requirements.

SiteMinder Administrators create and assign roles that determine access to a protected application. For example, a business rule may require that only employees with the title "accountant" use a financial application. A SiteMinder Administrator creates a role whose membership is to include employees with the "accountant" title. The administrator then creates an application security policy to protect the application, associating the role with the policy. The policy protects the financial application and only authorizes users with the "accountant" title.

Unlike adding users and user groups to a policy, the scope of roles is not limited to a single directory nor is it limited to a specific directory type. A SiteMinder administrator expresses business requirements in the Administrative UI by creating membership expressions. Membership expressions map to specific LDAP and ODBC user directory attributes. The SiteMinder administrator then defines the role using the membership expressions. As a result, roles are not dependent on user directory-specific attributes and can span across multiple user directories.

Note: More information on application roles exists in Enterprise Policy Management.

CA Identity Manager Roles and Access Control

If you have integrated CA Identity Manager and SiteMinder, you can implement policy–based access control using CA Identity Manager roles. CA Identity Manager roles enable centralized management of user privileges in external applications.

Note: Fore more information about configuring the integration, see the CA Identity Manager documentation.

The integration requires that:

- The Policy Server installation includes the required data definitions for the CA Identity Manager integration at the following location.

  `siteminder_home\xps\dd`

  siteminder_home

  Specifies the Policy Server installation path.

- The name of the file is:

  `IdmSmObjects.xdd`

  Important! Do not import this file in to the policy store until you have completed the CA Identity Manager integration. If you import the data definitions before completing the integration, the Policy Server can reach an indeterminate state. Coordinate the integration with your CA Identity Manager administrator.

- CA Identity Manager administrators manage environments and roles in CA Identity Manager to determine user access to the applications that SiteMinder secures. The SiteMinder Administrative UI refers to these environments as an IDM environment.

  Note: For more information about environments and roles, see the CA Identity Manager documentation.

- SiteMinder administrators use the Administrative UI to associate one or more IDM environments to a policy domain and user directory. A SiteMinder administrator cannot create or manage an IDM environment.

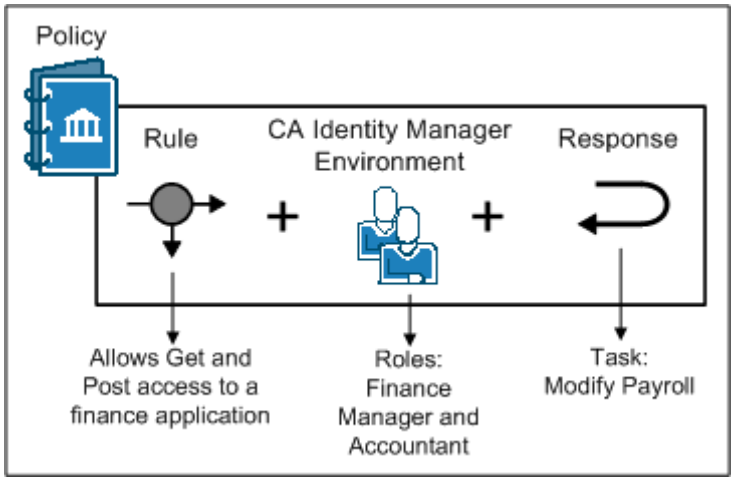
- SiteMinder administrators use the Administrative UI to create a policy and associate one or more roles that are available to the IDM environments. A SiteMinder administrator cannot create or manage a CA Identity Manager role.

  Note: You cannot apply a CA Identity Manager role to an enterprise management application.

SiteMinder can also provide details about entitlements that a CA Identity Manager user has in protected applications. As the following figure illustrates, a SiteMinder administrator associates a response with an access rule in the policy. The response contains a response attribute that specifies a SiteMinder–generated user attribute.

The SiteMinder–generated user attribute retrieves task information from CA Identity Manager. The Policy Server passes this information to the web agent as an HTTP header variable or a cookie. The web agent makes the header variable or cookie available to the protected application for fine–grained access control.

Figure 1: CA Identity Manager and fine-grained access control



Chapter 4: Administrative User Interface Management

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 log in to the Administrative UI and the common procedures that you 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.

Do not use the Refresh or Back buttons of the browser while using the Administrative UI. Using these buttons resubmits the form, repeats the action that was initiated by clicking a button in the form, and creates an invalid state.

Start the Administrative UI

Follow these steps:

1. Open a web browser:
   - If you installed the Administrative UI using the stand–alone option and you registered it over SSL, go to the following location:

     https://*host.domain*:8443/iam/siteminder/adminui
   - If you installed the Administrative UI using the stand–alone option and you did not register it over SSL, go to the following location:

     http://*host.domain*:8080/iam/siteminder/adminui

- If you installed the Administrative UI to an existing application server infrastructure, go to one of the following locations:

  http://*host*.*domain*:*port*/iam/siteminder/adminui

  https://*host*.*domain*:*port*/iam/siteminder/adminui

  host

  Specifies the name of the Administrative UI host system.

  domain

  Specifies the fully qualified domain name of the Administrative UI host system.

  port

  Specifies the port on which the application server listens for requests.

2. Enter the credentials of a SiteMinder administrator.
3. 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 log in to the Administrative UI.

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 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 an object by copying and modifying an existing object

1. Click *tab*, *sub_component*.

   tab

   Specifies one of the top–level tabs in the Administrative UI.

   sub_component

   Specifies a high–level category that is available when a tab is selected. These categories represent one or more SiteMinder objects.

   Example: Click Infrastructure, Agent.

2. Click *siteminder_object*.

   siteminder_object

   Specifies the type of a SiteMinder object.

   Example: Agent

   The *siteminder_object* screen appears.

3. Click Create *siteminder_object.*

   The *Create siteminder_object* screen opens.

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

   A list of objects that match the search criteria opens.

5. Select an object from the list and click OK.

   The Create *siteminder_object*: Copy of *object_name* screen appears.

   object_name

   Specifies the name of the object from which the new object is based.

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

   The Policy Server object is created.

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*, *sub_component*.

   tab

   Specifies one of the top–level tabs in the Administrative UI.

   sub_component

   Specifies a high–level category that is available when a tab is selected. These categories represent one or more SiteMinder objects.

   Example: Click Policies, Domain.

2. Click *siteminder_object*.

   siteminder_object

   Specifies the type of a SiteMinder object.

   Example: Domain

   The *siteminder_object* screen appears.

3. Specify search criteria and click Search.

   A list of objects that match the search criteria appears. The name of the object is a link.

4. Click the name of the object that you want to view.

   The View *siteminder_object*: *object_name* screen appears.

   object_name

   Specifies the name of the object that you are viewing.

   Note: To view another object of the same type, click Close to return to the search screen.

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 object

1. Click *tab*, *sub_component*.

   tab

   Specifies one of the top–level tabs in the Administrative UI.

   sub_component

   Specifies a high–level category that is available when a tab is selected. These categories represent one or more SiteMinder objects.

   Example: Click Policies, Domain.

2. Click *siteminder_object*.

   siteminder_object

   Specifies the type of a SiteMinder object.

   Example: Realms

   The *siteminder_object* screen appears.

3. Specify search criteria and click Search.

   A list of objects that match the search criteria appears. The name of the object is a link.

4. Click the name of the object that you want to modify.

   The View *siteminder_object*: *object_name* screen appears. All fields and controls are inactive.

   object_name

   Specifies the name of the object that you want to modify.

5. Scroll to the bottom of the page and click Modify.

   All fields and controls are active.

6. Make the required changes and click Submit.

   The Policy Server object is modified.

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*, *sub_component*.

   tab

   Specifies one of the top–level tabs in the Administrative UI.

   sub_component

   Specifies a high–level category that is available when a tab is selected. These categories represent one or more SiteMinder objects.

   Example: Click Infrastructure, Authentication.

2. Click *siteminder_object*.

   siteminder_object

   Specifies the type of a SiteMinder object.

   Example: Authentication Schemes

   The *siteminder_object* screen appears.

3. Specify search criteria and click Search.

   A list of objects that match the search criteria appears.

4. Select the object that you want to delete and click Delete *siteminder_object*.Click the name of the object that you want to view.

   A confirmation screen appears.

5. Click Yes.

   The Policy Server object is deleted.

Manage Task-persistence Database

Every Administrative UI task stays in the task-persistence database indefinitely or until removed by a SiteMinder administrator. You can remove tasks from the database and free up disk space by scheduling cleanup tasks. Cleanup tasks allow you to manage the size of the task-persistence database and improve runtime performance.

Every task exists in the task-persistence database in one of the following states:

- Audit State

  A task in the audit state has been initiated in the Administrative UI, but not submitted. For example, View tasks are initiated in the Administrative UI, but are never submitted.

- Submitted State

  Submitted tasks are tasks that have been submitted for processing in the Administrative UI, but that are not yet complete.

- Completed State

  Completed tasks are submitted tasks that completed processing. Completed tasks include tasks that completed processing successfully and tasks that failed to complete processing successfully, but are nonetheless complete.

Cleanup tasks can remove tasks in the audit state and completed state from the task-persistence database. Cleanup tasks cannot remove submitted tasks that are still pending.

In the Admin UI menu on the Administration tab in the Administrative UI, you can schedule, modify, and delete cleanup tasks through the following two options:

Clean Up Submitted Tasks

Use this option to schedule new cleanup tasks or modify existing ones.

Delete Recurring Tasks

Use this option to delete scheduled cleanup tasks.

Cleanup Submitted Tasks

You can manage the size of the task-persistence database and can improve runtime performance by scheduling cleanup tasks. You can configure cleanup tasks to remove tasks in the completed state and the audit state from the database. You can also configure limits for the cleanup task itself.

Note: The Cleanup Submitted Tasks option only appears in the Administrative UI (Administration, Admin UI) and you can run scheduled jobs for cleaning up the submitted tasks when you log in as the SiteMinder System Manager. The SiteMinder System Manager account is defined using the Configure Administrative Authentication option. This account that is used during the initial registration of the Administrative UI can be from an external administrator store. The Cleanup Submitted Tasks option does not appear in the Administrative UI for an administrator, even with superuser permissions, and so cannot schedule cleanup tasks.

To clean up submitted tasks

1. Click Administration, Admin UI, Clean Up Submitted Tasks.

   The Recurrence pane opens.

2. Select one of the follow option buttons:

   - Execute now

     Select this option and click Next to skip the scheduling step and go directly to the Clean Up Submitted Tasks pane.

   - Schedule new job

     Scheduling sections open.

   - Modify existing job

     Scheduling sections open.

3. Specify the name of the cleanup task in the Job Name field, the type of schedule, and the scheduling details on the scheduling sections, and click Next.

   The Clean Up Submitted Tasks pane opens.

4. Complete the following fields on the Clean Up Submitted Tasks pane:

   Minimum Age

   Specifies the minimum age in Months, Weeks, Days, Hours, or Minutes of the completed tasks to remove from the task-persistence database.

   Note: Task age is measured from the time that tasks are completed.

   Audit Timeout

   (Optional) Specifies the maximum number of days to keep tasks in the audit state in the task-persistence database.

   Limits: one or greater

   Default: one

Time Limit

(Optional) Specifies a time limit in minutes for the cleanup task.

Task Limit

(Optional) Specifies a task limit for the cleanup task.

5. Click Finish.

   The Cleanup task is submitted for processing.

Delete Recurring Tasks

You can delete scheduled cleanup tasks that are no longer needed.

To delete recurring tasks

1. Click Administration, Admin UI, Delete Recurring Tasks.

   The Delete Recurring Tasks pane opens.

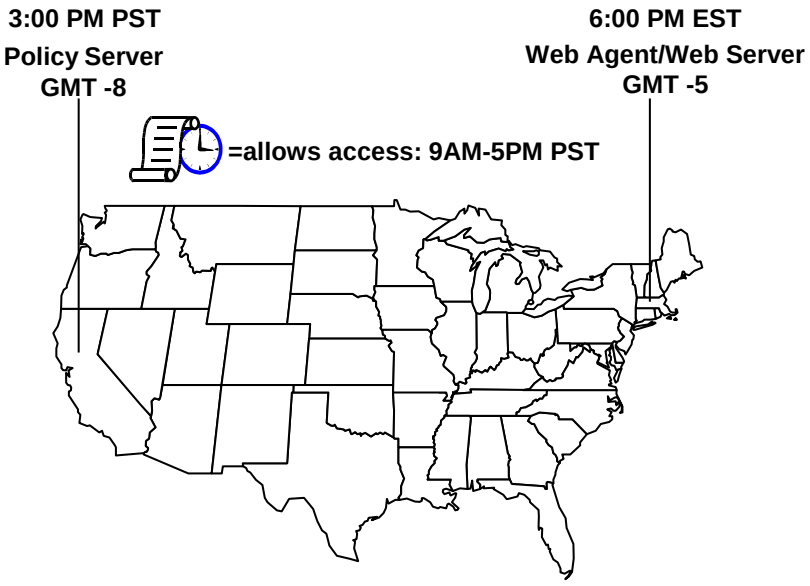
2. Select one or more scheduled cleanup tasks to delete, and click Submit.

   The Delete task is submitted for processing.

How the Web Agent and Policy Server Calculate Time

For each system that has a Policy Server or Web 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 figure shows how the Policy Server executes a policy relative to time. A resource is stored on a Web 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 Web Agent's time zone, Eastern Standard Time (EST).



Note: 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.

Protecting the Administrative UI with SiteMinder

Protecting the Administrative UI with SiteMinder requires that you configure an agent to function with a reverse proxy server and configure an external administrator store. Rather than accessing the Administrative UI directly on the application server, you access the Administrative UI through the reverse proxy server.

Consider the following:

- If you have more than one Administrative UI to protect, we recommend protecting each instance with a separate agent functioning with a reverse proxy server.
- If the Administrative UI is installed to WebSphere, the application server host system requires at least 500 MB of free memory to enable SiteMinder authentication.

How to Protect the Administrative UI with SiteMinder

You can protect the Administrative UI with SiteMinder:

Follow these steps:

1. Configure an agent to operate with a reverse proxy server.

   Certain types of web servers, such as Apache, that support SiteMinder Web agents can also function as reverse proxy servers. See the support matrix for the supported servers.

   Note: Update the configuration file of Apache web server to make the Apache web server function as a reverse proxy server. For more information about configuring a reverse proxy server and updating the configuration file, see the *Web Agent Configuration Guide*.

   Important! The URL used in the rules that are set for the proxy server must be the same URL used to register the Administrative UI initially.

   Example:

   If the Administrative UI was initially registered with the following URL, specify the same URL in the proxy server rules.

   http://*host_name*:8080/iam/siteminder/adminui

2. In your agent configuration object (ACO), set the value of the LogOffUri parameter as shown in the following example:

   ```
   /iam/siteminder/logout.jsp
   ```

   Note: For more information about setting the LogOffUri parameter, see the *Web Agent Configuration Guide*.

3. Configure an external administrator store.

   Note: The application server restarts automatically after you configure the external administrator store. The Administrative UI is protected with SiteMinder only after the restart.

More information:

How to Configure an External Administrator Store (see page 70)

Change the Authentication Scheme

The default SiteMinder authentication scheme used to protect the Administrative UI is basic user name and password. You can change the default authentication scheme to any SiteMinder supported authentication scheme, except SAML and WS-Fed authentication.

Follow these steps:

1. Click Policies, Domain.
2. Click Realms.
3. Search for the following realm and click the name to open it:

   SiteMinder_ims_realm

   Note: This realm is associated with a domain named SiteMinderDomain.
4. Click Modify to enable the settings.
5. Select the authentication scheme you want from the Authentication Scheme list .
6. Enter additional settings, if required.
7. Click Submit.

   The Administrative UI is protected using the selected authentication scheme.

More information:

Authentication Schemes (see page 301)

Disable SiteMinder Authentication for the Administrative UI

If you do not want to protect the Administrative UI with SiteMinder, you can disable SiteMinder authentication. You can access the Administrative UI through the reverse proxy server only even after you remove SiteMinder protection for the Administrative UI.

To access the Administrative UI directly on an application server, delete the data directory and reregister the Administrative UI with the Policy Server.

Follow these steps:

1. Log in to the Administrative UI.
2. Run the Administrative Authentication wizard to specify that you no longer want to protect the Administrative UI using SiteMinder authentication.

   Note: Leave the existing directory server or database connection information to continue using the external administrator store.
3. Log in to the Administrative UI host system.
4. Delete the Administrative UI data directory. The type of application server to which you deployed the Administrative UI determines where the data directory is located:
   - If you installed the Administrative UI using the stand-alone option, the data directory is located at the following location:

     install_dir/adminui/server/default/data
   - If you installed the Administrative UI to an existing application server, the default location for the data directory is:
     - (JBoss): <JBOSS Install>/server/default/data
     - (WebSphere): <WebSphere>\AppServer\profiles\<your_profile>\data
     - (WebLogic): <welbogic install>\user_projects\domains\<user_domain>\data
5. Log in to the Policy Server host system and reset the Administrative UI registration window using the XPSRegClient utility.
6. Register the Administrative UI with the Policy Server.

   Note: For more information about resetting the registration window and register the Administrative UI, see the *Policy Server Installation Guide*.

Chapter 5: SiteMinder Administrators

This section contains the following topics:

SiteMinder Administrators Overview

A SiteMinder administrator is anyone who has access to Policy Server objects and tools.

You can configure multiple SiteMinder administrator accounts so that different administrators can log in with the ability to manage different interfaces, resources, and features according to their different roles in an organization.

This fine-grained administrative model allows you to delegate the management of Policy Server objects and SiteMinder tools across a few or many individuals in an organization.

A default SiteMinder superuser account with full system privileges is created when you configure the policy store, which is the default source of administrator identities. This default configuration lets you manage the environment immediately after installing the software.

However, we recommend that you configure an external administrator user store, such as a corporate directory, and create additional administrator accounts whose privileges can be configured to delegate administrative authority.

Default Superuser Administrator

When you configure the policy store, a default superuser account is created. This account has the maximum system privileges, which you use for the following operations:

- Registering the Administrative UI with a Policy Server.
- Creating any other type of Policy Server object, including other administrator accounts.
- Using any of the Policy Server tools.
- Trusted Host administration.
- Managing the Policy Management API.

The default superuser account has the following credentials:

User Name

siteminder

Password

The password that you specified when configuring the policy store.

Note: For more information about configuring a policy store, see the *Policy Server Installation Guide*. For more information about registering an Administrative UI, see the Policy Server Installation Guide.

Administrator Accounts

Administrator accounts can be used to perform the following SiteMinder administration tasks:

- Manage SiteMinder objects using the Administrative UI
- Use Policy Server tools, such as XPSImport and XPSExport

Create Administrator accounts to delegate fine-grain privileges that determine the administrative capabilities available to that administrator. Specifically, Administrator accounts define the following properties:

Scope

Specifies whether the Administrator can access all SiteMinder data or only those objects defined in an assigned administrative *Workspace*.

Access methods

Specifies what methods the Administrator can use to access and manage the SiteMinder data.

Rights

Specifies what categories of SiteMinder objects the Administrator can access, and whether they can only view or view and modify those objects.

This granularity allows you to create administrators and assign privileges to match the administrative roles in your organization.

Note: You can only create new Administrator accounts that are associated with administrative users in an external administrator store. However, Administrator accounts are automatically generated for Legacy Administrator records stored in the policy store to allow those administrators to access the Administrative UI.

More information:

How to Configure an External Administrator Store (see page 70)
How to Create an Administrator (see page 85)
Limit Administrator Account Scope Using Workspaces Overview (see page 87)
How to Create a Scoped Administrator (see page 89)
Administrator Use Cases (see page 94)

Legacy Administrator Accounts

Legacy Administrator accounts can be used to perform the following administrative tasks:

- Use Policy Server tools, such as smobjimport and smobjexport.
- Function as a Trusted Host Administrator. A Trusted Host Administrator has the right to run the host registration process from the SiteMinder Agent host system. The registration process enables an Agent to communicate with the Policy Server.
- Use the Policy Management API.

  Note: If your environment includes a script or program that uses the Policy Management API, a Legacy Administrator account is required. Create a Legacy Administrator that has the authentication privileges to execute the functions through the Policy Management API.

Note: Legacy Administrators can also be used to access the Administrative UI if the policy store is configured as the source of administrator identities (the default). Once an external administrator store is configured, Legacy Administrator accounts can no longer be used to access the Administrative UI.

More information:

Administrator Store Options (see page 70)

Administrator Store Options

By default, the Administrative UI uses the policy store as its source of administrator identities. However, we recommend that you use an external administrator user store, such as a corporate directory, for further administrator accounts.

Consider the following factors when deciding where to store administrator identities:

- If you are configuring the Administrative UI with a single Policy Server, you can use the policy store to store administrator identities.
- If you are configuring the Administrative UI with multiple Policy Servers, you must use an external administrator store.
- If you store administrator identities in the policy store, you can only establish a new administrator record in the policy store by creating a Legacy Administrator.
- If you store administrator identities in an external store, you create Administrator accounts to locate administrator records in the external store.

  Note: You cannot create new Legacy Administrators or associate Administrator accounts with Legacy Administrator records to allow Administrative UI access once an external administrator store is configured.
- An external administrator store can be used to help ensure that multiple Administrative UI instances share the same set of administrators. By default, the Administrative UI uses the policy store that is configured with the registered Policy Server.

Note: For more information about installing the Administrative UI and configuring additional Policy Server connections, see the *Policy Server Installation Guide*.

How to Configure an External Administrator Store

Complete the following steps to configure a connection to an external administrator store.

1. (Optional) If you want to protect the Administrative UI with SiteMinder, configure an agent to function with a reverse proxy server.

   Note: For more information about configuring a reverse proxy server, see the *Web Agent Configuration Guide*.
2. Review the external administrator store considerations.
3. Review the SSL considerations.
4. Depending on your store type, do the following:
   - (LDAP) Gather directory server connection information.
   - (RDB) Gather database connection information.

- (RDB) Deploy a Java Database Connectivity (JDBC) data source to the application server.
  - If you installed the Administrative UI using the stand-alone option, use the smjdbcsetup utility to configure and deploy the data source.
  - If you installed the Administrative UI to an existing application server infrastructure, see your vendor-specific documentation for information about configuring and deploying a data source.

    Note: If you are deploying a data source to WebSphere, be sure that the JNDI name, under the datasource properties, is prefixed with the following:

    jdbc/

    Example: If the datasource name is abc, then the JNDI name is jdbc/abc.

5. Configure the connection to the external administrator store.
6. (Optional) Migrate Legacy Administrator Administrative UI permissions.

External Administrator Store Considerations

Before you configure an external administrator store connection, consider the following items:

- **Important!** Discontinuing the use of the policy store as the source of administrator identities is permanent. Configuring an external administrator store only affects the Administrative UI that is configured to use the external store. Any other Administrative UI not yet configured to use the external store continues to use the policy store to identify administrators.
- Legacy Administrators, including the default SiteMinder super user account, can continue to perform the following actions:
  - Manage the Policy Management API
  - Function as a Trusted Host administrator
  - Use Policy Server tools
- Legacy Administrators, including the default SiteMinder super user account, can no longer manage SiteMinder objects in the Administrative UI.

- If you have Legacy Administrators who must continue using the Administrative UI, use your vendor-specific tools to add these users to the external store. Once the user identities are established in the external store, you can reinstate these privileges by mapping the existing user paths from the policy store to the external store.

  Important! External administrator authentication does not let a single Legacy Administrator account retain rights to the Administrative UI, the Policy Management API, and Trusted Host privileges at the same time. If a Legacy Administrator must continue functioning in these roles, leave the Legacy Administrator unchanged. Be sure that the user is present in the external store and separately configure a new Administrator using the external user identity.

- A super user that you identify when configuring the connection to the external store replaces the default SiteMinder super user account. The external user becomes the super user and has maximum permissions in the Administrative UI and access to all Policy Server tools.

  Use the external super user to delegate permissions to new Administrators.

- If multiple Administrative UI instances are to use the same administrator authentication store, be sure to configure each connection using the same network identifier. Mixing network identifiers for multiple Administrative UI connections to the same external administrator authentication store is not supported.

  Example: If you configured the first connection with 172.16.0.0, create subsequent connections with 172.16.0.0. If you configured the first connection with comp001@example.com, create subsequent connections with comp001@example.com.

SSL Considerations

If you are configuring the external administrator store connection over SSL, consider the following items:

- The directory server must be configured to communicate over SSL.

  Note: For more information about configuring the directory server for SSL, see your vendor–specific documentation.

- If you installed the Administrative UI using the stand–alone option, the Administrative UI is installed with an embedded certificate database.

- If you installed the Administrative UI to an existing application server infrastructure, implement a certificate database as required by your application server.

  Note: For more information about implementing a certificate database, see your vendor–specific documentation.

- Be sure to enter an SSL–enabled port when entering directory connection information. If you do not enter an SSL–enabled port, the Administrative Authentication wizard becomes unresponsive.

Gather Directory Server Information

If you are configuring a connection to a directory server, gather the following information:

- **Host name**—Identify the IP address or fully qualified domain name of the directory server host system.
- **Port**—Identify the port on which the directory server is listening.
- **Directory server user credentials**—Identify the user name and password of an account that has read/write permissions to the directory server.
- **Search root**—Identify the base DN of the directory server.
- **SSL certificate**—If the directory server is configured to communicate over an SSL connection, obtain the SSL certificate.

Gather Database Information

If you are configuring a connection to a database, gather the following information:

- **Host name**—Identify the name of the database host system.
- **Port**—Identify the port on which the database is listening.
- (Microsoft SQL Server) **Database name**—Identify the name of database.
- (Oracle) **Service name**—Identify the service name of the database.
- **Database user credentials**—Identify the credentials of a user account that has read/write permissions to the database.

  Important! If you are configuring a connection to Oracle, be sure to set the default schema for this user. The default schema must be the schema that is associated with the table that contains the administrative users. If you do not set the default schema for this user, the Administrative Authentication wizard cannot locate users in the database.

Deploy a JDBC Data Source

If you are configuring a connection to a relational database, the Administrative UI requires a JDBC data source to communicate with the administrator store. A utility is required to create the data source. If you installed the Administrative UI using the stand-alone option, the smjdbcsetup utility is provided for you.

Note: If you installed the Administrative UI to an existing application server, see your vendor-specific documentation for information about deploying a JDBC data source. If you are deploying a data source to WebSphere, verify that the JNDI name, under the datasource properties, is prefixed with the following text:

```
jdbc/
```

Example: If the datasource name is abc, then the JNDI name is jdbc/abc.

Follow these steps:

1. Log in to the Administrative UI host system.
2. (UNIX) Stop the SiteMinder Administrative UI service.

   Note: For more information about stopping the service, see the *Policy Server Installation Guide*.
3. Navigate to *administrative_ui_home*\CA\SiteMinder\adminui\bin.

   administrative_ui_home

   Specifies the Administrative UI installation path.
4. Run one of the following commands:

   - (Windows)

     ```
     smjdbcsetup.bat
     ```

     Important! Before running a SiteMinder utility or executable on Windows Server 2008, open the command line window with administrator permissions. Open the command line window this way, even if your account has administrator privileges.

   - (UNIX)

     ```
     smjdbcsetup.sh
     ```

   The utility prompts you for a unique identifier. The utility appends the identifier to the data source.
5. Type a value and press Enter.

   The utility prompts you for a database driver type. The driver types are prefixed with a number.
6. Type a number to select a driver type and press Enter.

   The utility prompts you for the name of the database host system.
7. Type the database host name and press Enter.

   The utility prompts you for the port on which the database is listening.

8. Type the database port and press Enter.
   - If you are configuring a connection to Microsoft SQL Server, the utility prompts you for the database name.
   - If you are configuring a connection to Oracle, the utility prompts you for the service name.
9. Type the database name or the service name and press Enter.

   The utility prompts you for the database user account name.
10. Type the database user account name and press Enter.

    Note: This user account must have read/write permissions to the database.

    The utility prompts you for the password of the database user.
11. Type the password and press Enter.

    The connection details appear.
12. Review the details and do one of the following steps:
    - To configure and deploy the data source, type y and press Enter.

      The utility deploys the data source to *admin_ui_home*\CA\SiteMinder\adminui\server\default\deploy and prompts you to restart the SiteMinder Administrative UI service.

      admin_ui_home

      Specifies the Administrative UI installation path.

      Note: Restarting the SiteMinder Administrative UI service is required before you can use the data source to create the connection.
    - To cancel the deployment, type n and press Enter.
13. Do one of the following steps:
    - To start the service automatically, do one of the following steps:
      - (Windows) Type y and press Enter.
      - (UNIX) You must manually start the service. For more information about starting the service, see the *Policy Server Installation Guide*.
    - To start the service manually, type n and press Enter.

    The data source is configured and the utility exits.

Configure an LDAP Administrator Store Connection

Configure the connection to change the source of administrator identities from the policy store to the external store.

To configure the external store connection with SiteMinder authentication

1. Click Administration, Admin UI.
2. Click Configure Administrative Authentication.
   - If you are configuring an external administrator store for the first–time, the Configure Administrative Authentication wizard appears.
   - If the Administrative UI is configured with an external administrator store, the connection details appear. Click OK to start the Configure Administrative Authentication wizard.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. (Optional) If you want to protect the Administrative UI with SiteMinder, select an agent from the drop-down list and click Next.

   Be sure to select an agent that is configured to function with a reverse proxy server.
4. Select a directory server vendor from the Directory type list and click Next.

   The wizard prompts you for connection details.
5. Do the following:
   a. Type the IP address or the fully qualified domain name of the directory server host system in the Host field.

      Important! If multiple Administrative UI instances are to use the same administrator authentication store, take note of the network identifier you enter. Mixing network identifiers for multiple Administrative UI connections to the same external administrator authentication store is not supported.

      Example: If you configure the first connection with 172.16.0.0, create subsequent connections with 172.16.0.0. If you configure the first connection with comp001@example.com, create subsequent connections with comp001@example.com.

   b. Type the port on which the directory server is listening in the Port field.

      Important! If you are configuring the connection over SSL, be sure to enter an SSL–enabled port. If you do not enter an SSL–enabled port, the Administrative Authentication wizard becomes unresponsive when you click Next.

c. (Optional) Select Use SSL and upload a Certificate Authority (CA) certificate to enable SSL communication between the Administrative UI and the administrator store.

Note: The directory server must be configured to communicate over SSL. For more information about configuring the directory server for SSL, see your vendor–specific documentation.

d. Type the common name and password of a directory server user in the respective fields.

Note: This user must have read/write permissions to the directory server.

e. Click Next.

The wizard prompts you for object class information.

6. Do the following:

a. Type the directory server search root in the Search Root field.

b. Use the shuttle controls to add and remove the object classes that apply to the SiteMinder administrators.

c. Click Next.

The wizard prompts you to specify the individual attributes required to map to your administrative users. The lists populate with the attributes in your directory server that are likely to identify each attribute.

7. Select the mnemonic attribute string that maps to each of the required attributes and click Next.

The wizard prompts you to search for a user.

Important! Do not point to *any* attribute that is used or written to by the LDAP or any other applications otherwise you may always be redirected to the /logout.jsp page and unable to log in to the Administrative UI.

8. Enter all or part of the user name in the Keywords field.

Users matching the search criteria appear.

9. Select a user and click Next.

Note: You can only select one user. The user you select becomes the superuser when the connection is configured.

A summary page appears.

10. Confirm the connection details and click Finish.

The connection to the external store is configured.

Important! After you configure an external administrator store, restart the application server manually before you log in with the new credentials of administrator.

Configure an RDB Administrator Store Connection

Configure the connection to change the source of administrator identities from the policy store to the external store.

To configure the external store connection with SiteMinder authentication

1. Click Administration, Admin UI.
2. Click Configure Administrative Authentication.
   - If you are configuring an external administrator store for the first–time, the Configure Administrative Authentication wizard appears.
   - If the Administrative UI is configured with an external administrator store, the connection details appear. Click OK to start the Configure Administrative Authentication wizard.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. (Optional) If you want to protect the Administrative UI with SiteMinder, select an agent from the drop-down list and click Next.

   Be sure to select an agent that is configured to function with a reverse proxy server.
4. Select one of the following from the Directory type list:
   - If your user schema includes a column that identifies the full–name of users, select the Relational Database (with full name column) template.
   - If the full–name of users must be calculated, select the Relational Database (without full name column) template.
5. Click Next.

   The wizard prompts you to select a data source.

   Note: If data sources do not appear, click Cancel and deploy a JDBC data source to the application server. You cannot create the connection without a deployed data source.
6. Select the data source and click Next.

   The wizard prompts you to select the user table that contains the SiteMinder administrators.
7. Select the user table and click Next.

   The wizard prompts you to specify the individual attributes required to map to your administrative users. The lists populate with the column names in the database that are likely to identify each attribute.

8. Do one of the following:
   - If you selected the Relational Database template, select the column name that maps to each of the required user attributes and click Next.

     The wizard prompts you to search for a user.
   - If you selected the Relation Database (no full name) template:
     a. Select the column name that maps to each of the required user attributes.
     b. Build the Get Full Name Query by replacing ##FIRST_NAME, ##LAST_NAME, and ##PRIMARY_KEY with the column name that maps to the respective user attribute.

        Note: Leave the question mark (?) at the end of the query.

        Example: select SmUser.FirstName + ' ' + SmUser.LastName from SmUser where SmUser.UserID = ?
     c. Click Next

     The wizard prompts you to search for a user.
9. Enter all or part of the user name in the User Keywords field.

   Users matching the search criteria appear in Search Results.
10. Select a user and click Next.

    Note: The user you select becomes the super user when the connection is configured.

    A summary page appears.
11. Confirm the connection details and click Finish.

    The connection to the external store is configured.

Important! After you configure an external administrator store, restart the application server manually before you log in with the new credentials of administrator.

Migrate Legacy Administrator Permissions

If a Legacy Administrator must continue using the Administrative UI or Policy Server tools after configuring a connection to an external administrator store, migrate the permissions.

Important! External administrator authentication does not let a single Legacy Administrator account retain rights to the Administrative UI, Policy Server tools, the Policy Management API, and Trusted Host privileges at the same time. If a Legacy Administrator must continue functioning in one or more of these roles, leave the Legacy Administrator unchanged. Be sure that the user is present in the external store and separately configure a new Administrator using the external user identity.

Follow these steps:

Note: Be sure that the administrator is present in the external store. Log in to the Administrative UI using the external super user.

1. Click Administration, Administrator.
2. Click Administrators.

   The Administrators page appears.
3. Specify search criteria using the full name of the user and click Search.

   Users matching the search criteria appear.
4. Click the name of the Administrator you want to modify.

   The View Administrator page appears. The user path points to the policy store.
5. Click Modify.

   The settings and controls become active.
6. Click Lookup in General.

   The Select a User page appears.
7. Specify search criteria and click Search.

   Users matching the specified criteria appear.
8. Select the user that you want and click Select.

   The user path is updated to point to the external store.
9. Click Submit.

   The Administrative UI authenticates the administrator using the external store. The administrator has the same level of access to the Administrative UI when the policy store was being used to store administrator identities.

Update External Administrator Store Credentials

If the credentials that the Administrative UI uses to connect to the external administrator store change, submit the new credentials to the Administrative UI or SiteMinder administrator authentication fails.

If you installed the Administrative UI using the stand–alone option, two utilities are provided for you:

- (LDAP) Use the smjndisetup utility to update the directory server user account credentials.

  Note: To update the directory server host system name or port information, use the Administrative UI to re–create the connection to the external administrator store. The smjndisetup utility cannot update host or port information.

- (RDB) Use the smjdbcsetup utility to update the database user account credentials.

  Note: To update the database host system name or port information, use the smjdbcsetup utility to re–deploy the JNDI data source.

If you installed the Administrative UI to an existing application server infrastructure, consider the following items:

- (LDAP) Use the Administrative Authentication Wizard to update the credentials that the Administrative UI is to use to access the directory server.

  Important! After you use the wizard to update the credentials, update the credentials on the directory server as soon as possible. Administrators cannot log in to the Administrative UI until the directory server credentials are updated to match the credentials you supplied using the wizard.

- (RDB) Use the database–specific tool to update the data source.

More information:

Deploy a JDBC Data Source (see page 73)

Update Directory Server Credentials

Use the smjndisetup utility to update directory manager credentials.

Note: The smjndisetup utility can only update connection details that were configured using the Administrative UI. You cannot use the smjndisetup utility to create the connection credentials.

To update directory server credentials

1. Log in to the Administrative UI host system.
2. (UNIX) Stop the SiteMinder Administrative UI service.

   Note: For more information about stopping the SiteMinder Administrative UI service, see the *Policy Server Installation Guide*.
3. Navigate to *administrative_ui_home*\CA\SiteMinder\adminui\bin.

   administrative_ui_home

   Specifies the Administrative UI installation path.
4. Run one of the following commands:
   - (Windows)

     ```
     smjndisetup.bat --reset-password
     ```

     Important! Before running a SiteMinder utility or executable on Windows Server 2008, open the command line window with administrator permissions. Open the command line window this way, even if your account has administrator privileges.
   - (UNIX)

     ```
     smjndisetup.sh --reset-password
     ```

     The utility prompts you for the user name.
5. Do one of the following operations:
   - Type the new directory user and press Enter.
   - Press Enter to accept the default user name.

     The utility prompts you for the password of the user.

6. Type the new password and press Enter.

   The utility verifies the credentials and prompts you to update the directory connection credentials.

7. Type y and press Enter.

   - If you ran the utility on Windows, the utility restarts the SiteMinder Administrative UI service. The utility also updates the new directory connection details.
   - If you ran the utility on UNIX, start the Administrative UI service to update the new directory connection details.

     Note: For more information about starting the Administrative UI service, see the *Policy Server Installation Guide*.

Update Database Credentials

Use the smjdbcsetup utility to update database user credentials in the JNDI data source.

To update database credentials

1. Log in to the Administrative UI host system.
2. (UNIX) Stop the SiteMinder Administrative UI service.

   Note: For more information about stopping the SiteMinder Administrative UI service, see the *Policy Server Installation Guide*.

3. Navigate to *administrative_ui_home*\CA\SiteMinder\adminui\bin.

   administrative_ui_home

   Specifies the Administrative UI installation path.

4. Run one of the following commands:

   - (Windows)

     ```
     smjdbcsetup.bat --reset-password
     ```

     Important! Before running a SiteMinder utility or executable on Windows Server 2008, open the command line window with administrator permissions. Open the command line window this way, even if your account has administrator privileges.

   - (UNIX)

     ```
     smjdbcsetup.sh --reset-password
     ```

   The utility prompts you to enter a unique identifier.

5. Enter the name of the deployed data source.

   Note: If you do not know the data source name, you can locate all deployed data sources in *administrative_ui_home*\SiteMinder\adminui\server\default\deploy.

   administrative_ui_home

   Specifies the Administrative UI installation path.

   The utility prompts you for the database user name.

6. Enter the user name and press Enter.

   The utility prompts you for the user password.

7. Enter the password and press Enter.

   The utility prompts you to verify the new data source credentials and verify that they can be updated.

8. Type y and press Enter to confirm the new data source credentials.

   The utility updates the data source.

   - (Windows) The utility prompts you to restart the SiteMinder Administrative UI service.
   - (UNIX) A message appears stating that the SiteMinder Administrative UI service must be manually started.

9. Do one of the following tasks:

   - (Windows) Type y and press Enter to use the utility to restart the SiteMinder Administrative UI service and deploy the updated data source.
   - (Windows) Type n and press Enter to start the SiteMinder Administrative UI service manually. The data source is deployed when the service is started.
   - (UNIX) Manually start the SiteMinder Administrative UI service to deploy the data source.

   Note: For more information about starting the SiteMinder Administrative UI service, see the *Policy Server Installation Guide*.

Modify the External Administrator Store Connection

Run the Administrative Authentication wizard again to change the external store to which the Administrative UI connects for administrator authentication.

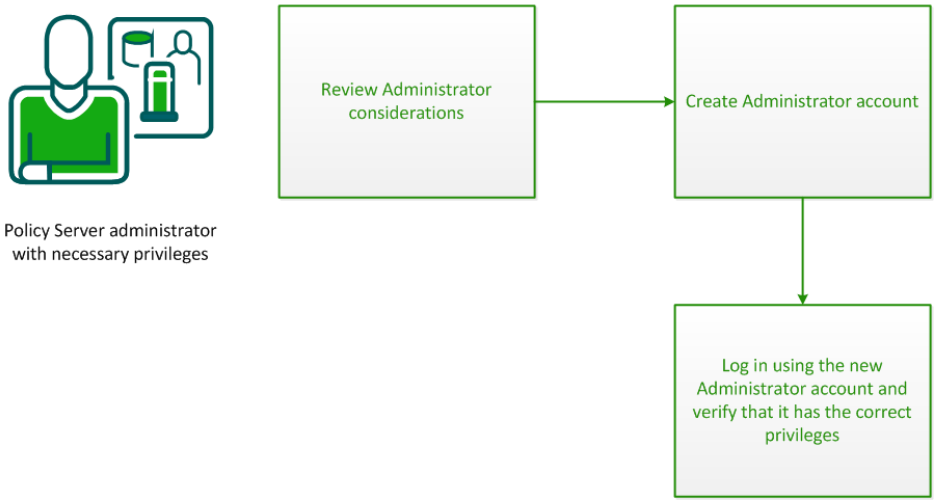
More information:

How to Configure an External Administrator Store (see page 70)

How to Create an Administrator

SiteMinder Administrator accounts can be configured with fine-grained privileges that determine the administrative capabilities available to that administrator.

SiteMinder Administrators are assigned rights to one or more security categories that define their administrative authority in the Administrative UI, such as managing authentication schemes. By default an administrator has access to every SiteMinder object related to an assigned security category.



1. Review the Administrator considerations (see page 85)
2. Create an Administrator account (see page 86)
3. Verify the privileges of the new Administrator account (see page 87)

Administrator Considerations

Before you configure an Administrator, review the following considerations:

- This process only applies if the Administrative UI is using an *external store* as the source of administrator identities. If you are using the policy store as the source of administrator identities, create a Legacy Administrator.
- The rights the Administrator requires to perform their job. Identifying these rights helps you delegate the appropriate security categories to the administrator.
- If the Administrators is responsible for application security policies.

Important! An Administrator can only create another Administrator with the same or lesser privileges. For example, if an Administrator has GUI and reports privileges, the Administrator can create another Administrator with GUI and reports privileges, but not with local API privileges. Similarly, an Administrator can only create another Administrator with the same or lesser scope (as defined by an assigned workspace).

Create the Administrator Account

Create an Administrator by creating an Administrator account.

Follow these steps:

1. Log in to the Administrative UI using the SiteMinder superuser or other administrator account with appropriate privileges.
2. Click Administration, Administrator.
3. Click Administrators.

   The Administrators page appears.
4. Click Create Administrator.

   The Create Administrator page appears.

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

   The Select a User page appears.
6. Specify search criteria and click Search.

   Users matching the specified criteria appear.
7. Select the administrator you want and click Select.

   The full name of the user appears in the Name field. The URL to the user in the external store appears in the User Path field.
8. Do one of the following:
   - To delegate all rights, select Super User and click Submit.
   - To delegate fine-grained privileges, go to the next step.
9. Specify how the administrator is permitted to interact with the Policy Server in the Access Methods section. Select as many methods as required for the administrator to perform tasks.

   Example: If an administrator is going to use the XPSImport and XPSExport tools, select Import Allowed and Export Allowed.
10. Click Add in the Rights section.

    The Create Permission: Select Security Categories page appears.

11. Select the security categories you want the administrator to manage and click OK.

    Note: Security categories comprise one or more tasks that correspond to specific SiteMinder objects. For more information, see the *Administrative UI online help system*.

    The Create Administrator page reappears; added security categories populate the Rights table.

12. Specify the permissions you want the administrator to have for each security category in the Rights table.

    Note: Only permissions applicable for security categories in the table are available. For more information about permissions, see the *Administrative UI online help system*.

13. Click Submit.

The Administrator is created.

More information:

Administrator Accounts (see page 68)
Limit Administrator Account Scope Using Workspaces Overview (see page 87)

Verify that the Administrator has the Correct Privileges

After creating an Administrator account, verify that it has the correct privileges.

Follow these steps:

1. Log in to the Administrative UI using the Administrator account.
2. Explore the Administrative UI to verify that only the security categories for which the account has rights are visible.

You have completed the required tasks to create an Administrator account.

Limit Administrator Account Scope Using Workspaces Overview

Administrator accounts are assigned rights to one or more security categories that define their administrative authority in the Administrative UI, such as managing authentication schemes. By default an Administrator account has access to every policy store object related to an assigned security category.

Workspaces define a subset of policy store objects. You can assign a workspace to one or more Administrator accounts to filter the objects that are available to them, further controlling the scope of their administrative authority. An Administrator account whose administrative authority is restricted by an assigned workspace is therefore known as a *scoped administrator*.

Note: You cannot assign workspaces to Legacy Administrator accounts — administrative scoping using workspaces is not related to domain scope limitations for Legacy Administrators.

Workspace Objects

A workspace object defines a subset of SiteMinder policy data that can be used to limit the scope of an Administrator to which it is assigned. A workspace can contain any top-level policy object (for example, a domain, authentication scheme, or host configuration object).

Note: The actual content of the workspace consists of the top-level contents plus any child objects (for example, realms under a domain) and some required objects, which are automatically included.

The contents of a workspace are dynamic:

- The contents of a workspace can be modified at any time. Any new objects are available immediately to all administrators assigned to the workspace. All removed objects are no longer available to administrators assigned to the workspace.
- If a scoped administrator has the right to modify an object type, that administrator can create objects in the workspace. These new objects are immediately available to all administrators assigned to the workspace.

More information:

Create the Administrator Account (see page 86)

Scoped Administrators

A scoped administrator is an Administrator account whose administrative authority in the Administrative UI is restricted to the subset of policy objects defined by an assigned workspace.

A scoped administrator cannot manage all the objects in the policy store for which they have rights. Instead, the Administrative UI appears and all policy management calls behave as if the policy store contains only the objects (and child objects) in the assigned workspace.

If a scoped administrator adds a new top-level object, that object immediately becomes available to all other similarly scoped administrators.

Scoped administrators that have the rights to create new administrators can only create administrators with the same or a more restrictive workspace than theirs. If they create new workspaces to further scope the new administrator, this new workspace object is added to their current workspace. The administrator can then assign their current workspace or the new workspace to the new administrator.

If the new administrator adds an object, the original administrator can also view it. This means that the effective set of objects that the original administrator can view includes any new objects added to workspaces that they created.

Note: Only Administrator accounts can be scoped using workspaces. Legacy Administrators cannot be scoped. However, Administrator accounts associated with Legacy Administrator records in the policy store can be scoped.

More information:

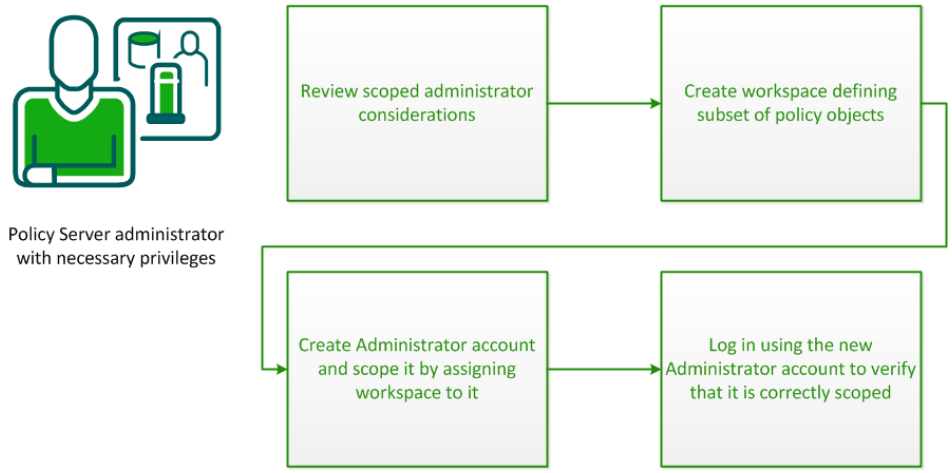
Create the Administrator Account (see page 86)

How to Create a Scoped Administrator

SiteMinder Administrator accounts can be configured with fine-grained privileges that determine the administrative capabilities available to that administrator.

SiteMinder Administrator accounts are assigned rights to one or more security categories that define their administrative authority in the Administrative UI, such as managing authentication schemes. By default an Administrator account has access to every SiteMinder object related to an assigned security category.

Workspaces define a subset of SiteMinder objects. Assign a workspace to one or more Administrator accounts to filter the objects that are available to them, further controlling the scope of their administrative authority. An Administrator account whose authority is restricted by an assigned workspace is known as a *scoped administrator*.



1. Review the scoped administrator considerations (see page 90)
2. Create a workspace defining a subset of SiteMinder objects (see page 91)
3. Create and scope an Administrator account (see page 92)
4. Verify the scope of the new Administrator account (see page 93)

Scoped Administrator Considerations

Before you configure a scoped administrator, review the following considerations:

- This process only applies for Administrator accounts (Legacy Administrators cannot be scoped using workspaces). However, Administrator accounts associated with Legacy Administrator records in the policy store can be scoped using workspaces.
- The scope of the Administrator. That is, whether the Administrator can manage all policy data or a subset of policy data defined in a workspace.
- The rights the Administrator requires to perform their job. Identifying these rights can help you delegate the appropriate security category to the administrator.
- Whether the Administrator is responsible for application security policies.

Important! An Administrator can only create another Administrator with the same or lesser privileges. For example, if an Administrator has GUI and reports privileges, the Administrator can create another Administrator with GUI and reports privileges, but not with local API privileges. Similarly, an Administrator can only create another Administrator with the same or lesser scope (as defined by an assigned workspace).

Create a Workspace

You create a workspace to define a subset of SiteMinder objects for which a scoped administrator has administrative privileges.

Follow these steps:

1. Log in to the Administrative UI using the SiteMinder superuser or other administrator account with appropriate privileges.
2. Click Administration, Administrator, Workspaces, Create Workspaces.

   The Create Workspace page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
3. Type the name and a description of the workspace in the fields in the General section.
4. Add objects that define the required subset of policy data to the workspace in the Members section:

   Note: Some commonly used objects are added to the workspace and appear in the Members list by default; you can remove them if necessary.

   a. Click Lookup.

      The Select Workspace Contents page appears.

   b. Select the type of objects that you want to add to the workspace from the Search for objects of type drop-down menu. Optionally, narrow the search to specific objects by Name or Description (or both).

   c. Click Search.

      A list of matching objects appears.

      Note: If the administrator account with which you are logged in is itself scoped, the list of matching objects is limited to those objects available to you.

   d. Select the object or objects you want to add to your workspace and click Select.

      The Create Workspace page reopens.
5. (Optional) Set administrator privileges for workspace members to read-only by setting the corresponding Read-Only check boxes.
6. Click Submit.

   The Create Workspace task is submitted for processing. SiteMinder verifies that the workspace is consistent (all required objects that are related to objects in the workspace are present in the workspace). If not, the missing objects are added and an information dialog appears indicating that some objects were automatically added to make the workspace consistent.

More information:

Limit Administrator Account Scope Using Workspaces Overview (see page 87)

Create an Administrator and Assign a Workspace

Create a scoped Administrator by creating an Administrator account and assigning a workspace that defines the scope of the objects that it can administer.

Follow these steps:

1. Log in to the Administrative UI using the SiteMinder superuser or other administrator account with appropriate privileges.
2. Click Administration, Administrator.
3. Click Administrators.

   The Administrators page appears.
4. Click Create Administrator.

   The Create Administrator page appears.

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

   The Select a User page appears.
6. Specify search criteria and click Search.

   Users matching the specified criteria appear.
7. Select the administrator you want and click Select.

   The full name of the user appears in the Name field. The URL to the user in the external store appears in the User Path field.
8. Select a workspace that defines the subset of objects to which the Administrator is scoped from the Workspace drop-down list.
9. Do one of the following:
   - To delegate all rights, select Super User and click Submit.
   - To delegate fine-grained privileges, go to the next step.
10. Specify how the administrator is permitted to interact with the Policy Server in the Access Methods section. Select as many methods as required for the administrator to perform tasks.

    Example: If an administrator is going to use the XPSImport and XPSExport tools, select Import Allowed and Export Allowed.

11. Click Add in the Rights section.

    The Create Permission: Select security categories page appears.

12. Select the security categories you want the administrator to manage and click OK.

    Note: Security categories comprise one or more tasks that correspond to specific SiteMinder objects. For more information, see the *Administrative UI online help system*.

    The Create Administrator page reappears.

13. Select the permissions (Read, Right, Modify, and Propagate) to apply to the security categories you added in the Rights section.
14. Click Submit.

The scoped Administrator is created.

More information:

Administrator Accounts (see page 68)
Limit Administrator Account Scope Using Workspaces Overview (see page 87)

Verify that the Administrator is Scoped

After assigning a workspace to an Administrator account, verify that it only has access to the scoped subset of objects.

Follow these steps:

1. Log in to the Administrative UI using the scoped Administrator account.
2. Explore the Administrative UI to verify that only the scoped subset of objects in the workspace are visible.

You have completed the required tasks to create a scoped Administrator account.

Administrator Use Cases

This administrator use case illustrates the following:

- How administrators are created
- How an existing administrator can create and grant privileges to a new administrator

This scenario involves three administrators

- SiteMinder Default Administrator (Superuser)
- Manager Admin
- Junior Admin

Using the three administrators, the following scenarios are described:

- Superuser creates Manager Admin
- Superuser creates a workspace and assigns it to scope Manager Admin
- Manager Admin creates Junior Admin
- Manager Admin creates a workspace and assigns it to further scope Junior Admin

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 administrator privileges extend to all policy objects or a subset of objects defined in an assigned workspace.

Permissions

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

Rights

Defines the complete set of privileges for the administrator. Rights are based on the security category, scope, and permissions.

Superuser Creates Manager Admin

The superuser is the administrator that was delegated system privileges when the connection to the external administrator user store was configured. The superuser can assign all categories, rights, and scope to any other Administrator.

From the Administrative UI, the superuser creates an administrator named Manager Admin. Initially, Manager Admin has no privileges until the superuser assigns them.

The superuser assigns the following to Manager Admin:

Access Method

GUI Allowed

Rights

Security Category	Permissions*
Agent Administration	V, M
Application Administration	V, M, P
Policy Administration	V, M, P

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

Note: The Propagate permission allows one manager to assign the category to another administrator.

At this stage, if Manager Admin logs in to the Administrative UI, they can view and modify all agents, applications, and domains in the policy store. If they create an administrator account, they can assign only Application Administration and Policy Administration security categories to that account.

Superuser Creates a Workspace and Assigns it to Scope Manager Admin

The superuser wants to limit the scope of Manager Admin so that they can only manage a subset of policy data. To do this, the superuser first creates a workspace named Workspace1 with the following members:

Type	Name
Agent	Agent1
Application	Application1
Application	Application2

Type	Name
Domain	DomainB

The superuser then edits the Administrator record for Manager Admin to assign Workspace1.

At this stage, if Manager Admin logs in they can now only view and modify Agent1, Application1, Application2, and policies in DomainB. Other agents, applications and domains in the policy store do not appear in the Administrative UI.

Manager Admin Creates Junior Admin

The second part of this use case shows how an administrator with a specific set of privileges creates another administrator.

Manager Admin creates an administrator named Junior Admin. When Manager Admin goes to add security categories for Junior Admin, only the following two (for which Manager Admin has the propagate permission) are available:

- Application Administration
- Policy Administration

Note: The propagate permission lets one administrator assign the category to another administrator.

Manager Admin proceeds to assign access methods and rights to Junior Admin as follows:

Access Method

GUI Allowed

Rights

Security Category	Permissions*
Application Administration	V
Policy Administration	V, M

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

At this stage, if Junior Admin logs in to the Administrative UI, they can view Application1 and Application2. They can view and modify DomainB (limited by default to the scope of Manager Admin).

Manager Admin Creates a Workspace and Assigns it to Further Scope Junior Admin

Manager Admin wants to further limit the scope of Junior Admin so that they can only manage a smaller subset of policy data. To do this, Manager Admin first creates a workspace named Workspace2 with the following members:

Type	Name
Application	Application2
Domain	DomainB

Manager Admin then edits the Administrator record for Junior Admin to assign Workspace2.

At this stage, if Manager Admin logs in they can now only view Application2, and view and modify policies in DomainB. Other applications and domains in the policy store do not appear in the Administrative UI.

How to Create a Legacy Administrator

Complete the following steps to create a Legacy Administrator:

1. Review the Legacy Administrator considerations (see page 97).
2. Create the Legacy Administrator record (see page 98).
3. Delegate Administrative UI permissions (see page 100).

Legacy Administrator Considerations

This process applies if one or more of the following criteria are met:

- A Legacy Administrator is required to manage SiteMinder objects using the Policy Management API
- A Legacy Administrator is required is to function as a Trusted Host administrator

Note: Legacy Administrators can also be used to access the Administrative UI if the policy store is configured as the source of administrator identities (the default). Once an external administrator store is configured, Legacy Administrator accounts can no longer be used to access the Administrative UI.

Create the Legacy Administrator Record

Create a Legacy Administrator record to store the Legacy Administrator identity in the policy store.

Follow these steps:

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

   The Legacy Administrators screen appears.
3. Click Create Legacy Administrator.

   The Create Legacy Administrator screen appears.
4. Click OK.

   The Create Legacy Administrator screen appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Do the following in the General section:
   a. Type a unique ID in the Name field.

      Consider entering a unique ID that adheres to your corporate standards. Adhering to your corporate standards does the following:

      - Prevents the administrator from having to remember a new set of credentials.
      - Facilitates a transition to an external administrator store because the Legacy Administrator ID matches the unique ID in the external store.

   b. Type the full name of the user in the Description field.

      The value appears as the display name when a user logs in to the Administrative UI. The display name identifies who is logged in.

      Example: If you enter Joe Smith, the Administrative UI display name appears as the following:

      Logged in as Joe Smith.
6. Leave the SiteMinder Database option button selected.
7. Enter the administrator password in the respective fields.
8. Do one of the following:
   - Click Submit if you are creating a Legacy Administrator to do only the following:
     - Use the Administrative UI
     - Use Policy Server tools

     The administrator record is created.

- Go to the next step if you are creating a Legacy Administrator to do any of the previous tasks and any one of the following:
  - Manage the Policy Management API
  - Function as a Trusted Host administrator

9. Select the System or Domain option from the Administrator Privileges section.

   Note: You delegate privileges to the Administrative UI after creating the Legacy Administrator.

   System

   An administrator has access to all policy domains in the Policy Management API. If you select System, a Task section appears.

   Domain

   An administrator has access to a specific subset of policy domains in the Policy Management API. If you select Domain, the Tasks and Scope section appears. The Tasks section lists the administrative tasks that can be performed. The Scope section lists the available domains that can be managed.

   Note: The Scope section on the Create Legacy Administrator screen is not related to Administrator account scoping using workspaces.

10. Select 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, do the following:
      a. Select one or more of the following tasks:
         – Manage Domain Objects
         – Manage Users
         – Manage Password Policies
      b. Select the domains that the administrator must manage in the Scope section.

11. Click Submit.

    You have created a Legacy Administrator.

    Note: An Administrator account, associated with the Legacy Administrator record in the policy store, is also created to provide Administrative UI access.

Delegate Administrative UI Permissions

When a Legacy Administrator is created, an Administrator account associated with the same record in the policy store is also generated. This Administrator account is configured with Administrative UI access privileges that correspond to the settings specified for the Legacy Administrator.

Changes to the Legacy Administrator settings that affect access privileges are automatically propagated to the associated Administrator account.

You can also directly modify the associated Administrator settings to take advantage of the fine-grained Administrative UI access privileges available to Administrator accounts.

Note: Changes to the associated Administrator account settings are *not* propagated back to the Legacy Administrator. Additionally, once changes are made to the associated Administrator, changes to the Legacy Administrator are no longer propagated to the Administrator account.

More information:

How to Create an Administrator (see page 85)
Limit Administrator Account Scope Using Workspaces Overview (see page 87)
How to Create a Scoped Administrator (see page 89)

Disable an Administrator

You can temporarily disable an Administrator without deleting the account. Disabling an account lets you reinstate the permissions without having to recreate the account.

To disable an Administrator

1. Click Administration, Administrator.
2. Click Administrators.

   The Administrators page appears.
3. Specify search criteria and click Search.

   Users matching the criteria appear.
4. Click the name of the user you want to disable.

   The View Administrator page appears.
5. Click Modify.

   The settings and controls become active.

6. Select Disabled.
7. Click Submit.

   The administrator is disabled.

You can enable the administrator at any time by repeating this procedure, clearing the Disabled option, and submitting the change.

Disable a Legacy Administrator

You can temporarily disable a Legacy Administrator without deleting the account. Disabling an account lets you reinstate the permissions without having to recreate the account.

Follow these steps:

1. Click Administration, Administrator.
2. Click Legacy Administrators.
3. Specify search criteria and click Search.
4. Click the name of the user you want to disable.
5. Click Modify to activate the settings.
6. Clear the tasks and click Submit.

   The Legacy Administrator no longer has access to the Policy Management API or to functions such as Trusted Host administrator.
7. (optional) If the Legacy Administrator has privileges in the Administrative UI, do the following:
   a. Click Administration, Administrator.
   b. Click Administrators.
   c. Specify search criteria and click Search
   d. Click the name of the Legacy Administrator from whom you removed privileges.
   e. Click Modify to activate the settings.
   f. Select Disabled and click Submit.

   The Legacy Administrator is disabled.

You can enable the Legacy Administrator by:

- Modifying the Legacy Administrator account to include system or domain–specific tasks.
- If applicable, clearing the Disabled check box from the respective Administrator record.

Restoring Administrator Access

An administrator with full access to the Policy Server can restore permissions when an administrator account becomes inadvertently modified or deleted. The two administrators with these permissions include the following:

- The default SiteMinder super user account (siteminder). If you are using the policy store to store administrator identities, use the siteminder account to restore permissions.
- The super user account that is established when the external administrator store connection is configured. If you are using an external store to store administrator identities, use the external super user to restore permissions.

If you cannot access the default administrator accounts, do the following:

- Use the smreg utility to change the password of the default SiteMinder super user account.
- Use the XPSSecurity utility to make any user in the external administrator store a super user.

  Note: For more information about using either utility, see the *Policy Server Administration Guide*.

Chapter 6: User Sessions

This section contains the following topics:

User Session Overview

SiteMinder maintains and administers consistent user session across multi-tiered applications. Maintaining user session information is critical as applications and Internet businesses become location-independent and the environments and applications technology and security needs vary.

Non-Persistent and Persistent Cookies

Standard SiteMinder session cookies are non–persistent. A non–persistent cookie is one that is maintained only in the memory of the web browser. When users close their browser, the session cookie is destroyed, effectively logging them out.

In addition to maintaining the cookie in the web browser memory, you can configure SiteMinder to have the cookie written to the hard disk. Maintaining a SiteMinder cookie in the web browser and on hard disk is known as a persistent cookie. When using persistent cookies, users that close and reopen their browser remain logged in.

Note: For more information about configuring a persistent cookie, see the *Web Agent Configuration Guide*.

Non-Persistent and Persistent Sessions

SiteMinder also provides the ability to configure a persistent session to provide Windows security context functionality and support for Federated Web Services. A persistent session is one in which a SiteMinder cookie is maintained in a SiteMinder session store, in the memory of the web browser, and optionally the hard disk.

Before you implement persistent sessions, consider the following:

- Persistent sessions are configured on a per realm basis.
- Persistent sessions should only be used where necessary. Using session services to maintain sessions has an impact on system performance.

Note: For more information about enabling and configuring session services, see the Policy Server Administration Guide.

Session Tickets

SiteMinder implements session management using session tickets. A session ticket contains basic information about a user and the authentication information for that user. The session ticket is used to identify the session of the user across all sites in a single sign–on SiteMinder environment. Session tickets are encrypted and only the Policy Server can read/validate them. SiteMinder web agents use session tickets to identify users and provide session information to the Policy Server.

The session ticket is handled differently depending upon whether the session is persistent or non–persistent.

Note: Non–persistent and persistent cookies are unrelated to the SiteMinder session of the user being non–persistent or persistent.

Non–persistent session

The web agent places the session ticket in a cookie. The cookie contains the user session data; no user-specific data is kept in the cookie itself. The web agent is responsible for validating the cookie and enforcing session timeouts.

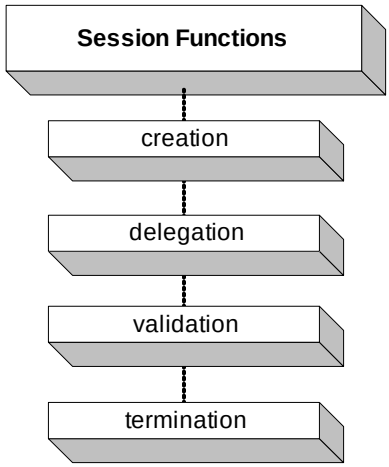
Persistent Session

The web agent places the session ticket in a session store database and, if possible, in an optional cookie on the client.

The session ticket data is used as an index into the cache of the web agent, which contains the user session data. If a cookie is written, no user–specific data is kept in the cookie itself. The web agent is responsible for validating the session and enforcing the session timeouts.

How SiteMinder Manages User Sessions

For the most part, SiteMinder manages user sessions automatically, performing a number of session management functions during the life cycle of a user session, as illustrated below.



Session creation

Establishing a session when a user successfully logs into an application. If a user fails to authenticate, no session is established.

Session delegation

Passing session information across an application environment. Delegating session information is necessary when an application's logic crosses several application tiers.

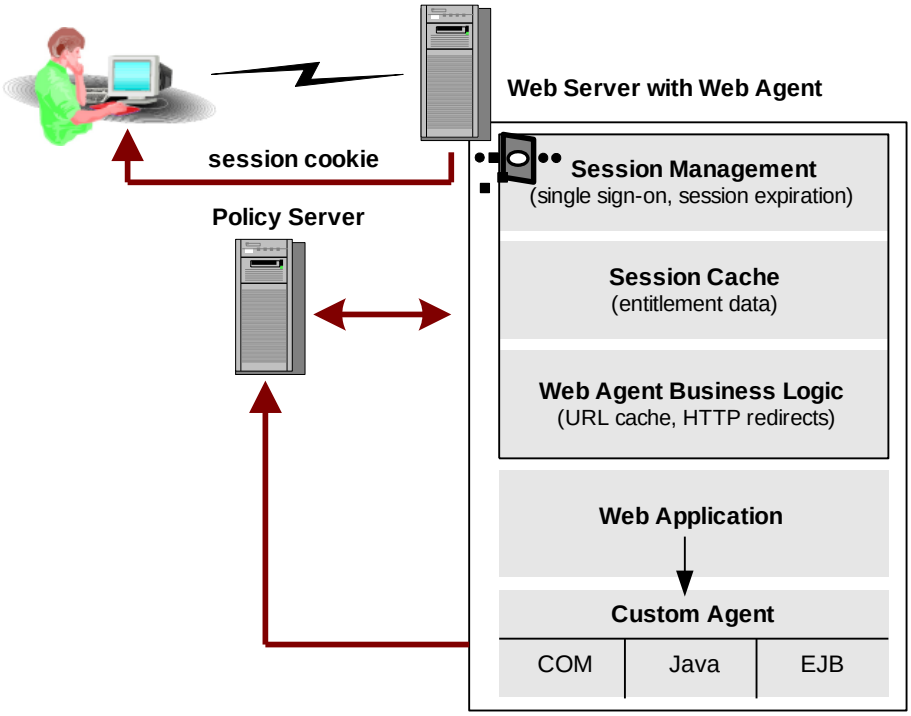
Session validation

Verifying the session ticket to make sure the user session is still active, that is, it has not expired or been terminated.

Session termination

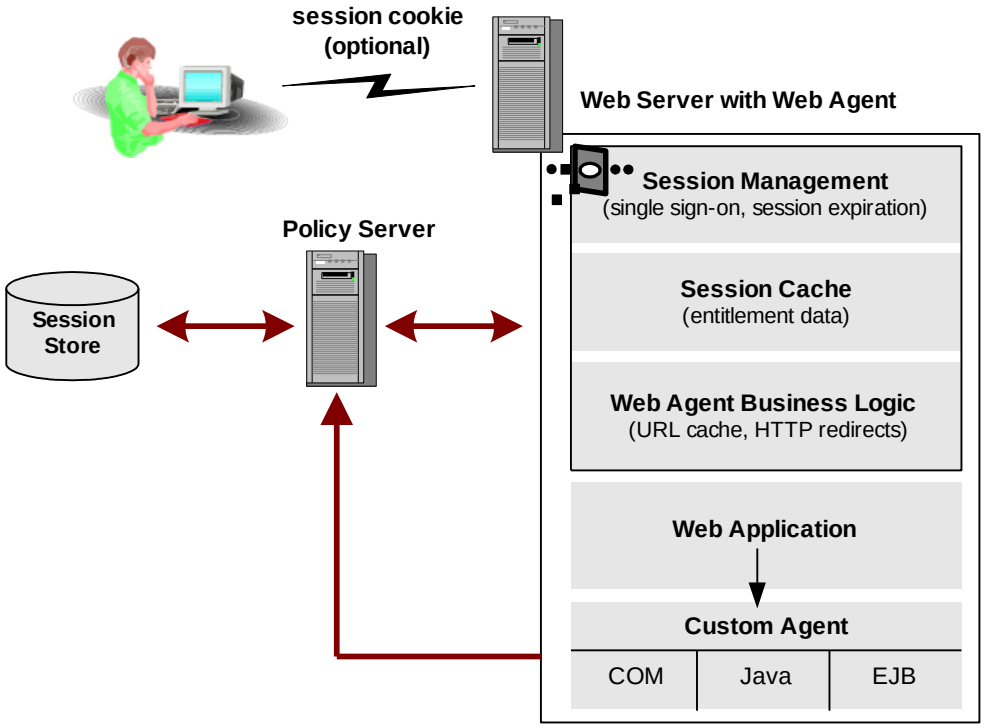
Ending a user session when a user logs out, when the configured session timeouts expire, or when a user is manually disabled by the SiteMinder System Manager. When a user logs out or the user session expires, they must log in again to create a new session. In the case of manual user disablement, the user can not re-initiate a session.

The following diagram illustrates how SiteMinder manages a non-persistent session.



The Web Agent passes the session ticket information across each application tier

The following diagram illustrates how SiteMinder manages a persistent session.



The Web Agent passes the session ticket information across each application tier

How a User Session Begins

A user session begins when a user logs in and is authenticated by SiteMinder. The Policy Server issues the session ticket, which is then required for all subsequent Agent requests for that user's session. If a user is forced to authenticate again during a session because of a higher protection level, the existing session is maintained.

A session is active until it is terminated when the user logs off, when the session expires, or when an administrator disables a user, thereby terminating the session.

How Sessions Across Realms Are Maintained

When a user requests access to a resource, his or her session is created within the context of the realm that contains that resource. An authentication scheme is also associated with a realm, and it determines the type of credentials that the user must present to gain access to the resource.

This authentication context is made available to all Web servers in the SiteMinder installation through SiteMinder's default HTTP headers that define components, such as the authentication scheme being used, the namespace the user is authenticating against, and other relevant information. In addition to the default headers, you can configure response attributes in a policy to communicate information for a user, such as a birth date or a phone number, that helps to further identify a user.

If the SiteMinder installation is configured for single sign-on, the authentication scheme may have a protection level assigned to it by an administrator. The level can be a number from 1 through 1000, with 1 being the least secure and 1000 being the most secure. These protection levels enable administrators to implement single sign-on with a higher level of security and flexibility.

An authenticated user of one realm can be validated for a session in another realm if the second realm is protected by an authentication scheme of an equal or lower protection level as the first. As long as the protection level is the same or lower, that user does not need to re-authenticate, which means that the user session remains valid. If a user tries to access a resource protected by an authentication scheme with a higher protection level, SiteMinder prompts the user to re-enter his or her credentials, thereby ending one user session and creating a new session.

To configure protection levels for your single sign-on environment, see Authentication Schemes (see page 301).

How Sessions Across Multiple Cookie Domains Are Maintained

SiteMinder supports single sign-on across multiple cookie domains in environments with heterogeneous Web server platforms. If a user visits companyA.com and then goes to companyB.com, his or her session information stays with them. To maintain session information across multiple cookie domains, the Web Agents must be configured for single sign-on. With single sign-on configured, the cookie that contains session information can be made available to all Agents and servers in the single sign-on environment.

The ability to pass session and identification information across multiple cookie domains enables a user to authenticate at a site in one cookie domain and then navigate to a site in another cookie domain without being re-challenged for information.

Single sign-on is accomplished using a cookie provider. The cookie provider is an extension of the Web Agents in the single sign-on environment.

To achieve cross-domain logout for resources in separate cookie domains, you can enable persistent sessions for the realms in separate cookie domains. When a user logs out in one domain, the Policy Server sends a logout event terminating the user session.

Note: Single sign-on across multiple cookie domains does not require that the same user directory be used across the single sign-on environment. However, user directory connections configured in the Administrative UI must share the same directory object name in each cookie domain.

How a User Session Is Validated

When a user requests access to a resource, the Web Agent validates the session by checking whether or not it is still active. The Web Agent first checks its session cache for session information. If the Web Agent reads the cookie and has the information in cache, it can validate the session. If it does not, then it contacts the Policy Server to verify the user's identity and collect any other session and authorization information.

When a user accesses a resource that has a higher protection level than the one used when establishing the session, the session information is maintained, even though another authentication takes place.

How Session Information Is Delegated

User sessions may be delegated between application tiers in a SiteMinder installation using the session ticket. The session ticket is the mechanism by which the user identity is passed from one application tier to another. Each application can make authorization calls to the Policy Servers after getting the session ticket.

If your SiteMinder installation uses custom Agents, the custom Agent must have access to the information in the session ticket to maintain session information.

In addition to using the session ticket for delegation, the Web Agent makes a set of default HTTP headers available for session management. These default headers can be passed across different business application tiers such as Enterprise Java Beans (EJB) and Component Object Model (COM) based tiers. Included in these headers is a unique session ID and optionally, a universal ID. The session ID identifies an active user session.

The universal ID identifies the user to an application in a SiteMinder environment. This ID is typically not the same as the user login ID, but is some other type of unique identifier like a telephone number or a customer account number. The universal ID helps facilitate identification between old and new applications. The universal ID delivers the user identification automatically, regardless of the application. In addition, the ID is built into applications. A built-in ID provides applications a user identification method that is separate from the user directory, which undergoes constant changes.

Both the session ID and the universal ID are shared among all the applications in a SiteMinder environment to maintain consistent user sessions.

Session Timeouts

Each user session includes session timeout information. The timeout values let you determine the length of an active session and the amount of session inactivity that can pass before a session is invalid. You configure session timeouts on a per-realm basis using the following timeout options.

Name	Purpose
Maximum Timeout (All sessions)	Specifies the maximum amount of time a user session can be active before the Web Agent challenges the user to re-authenticate. You can override this setting using the WebAgent-OnAuthAccept-Session-MaxTimeout response attribute.
Idle Timeout (All sessions)	Specifies the amount of time that a user session can be idle before the Web Agent terminates the session. If the session expires, a user must re-authenticate. **Note:** For persistent sessions, this value must be greater than that specified by Session Validation Period. You can override this setting using the WebAgent-OnAuthAccept Session-Idle-Timeout response attribute.
Session Validation Period (Persistent Sessions)	For persistent sessions only, specifies the maximum period between Agent calls to the Policy Server to validate a session. Session validation calls perform two functions: informing the Policy Server that a user is still active *and* checking that the user's session is still valid.

More information:

Realms (see page 501)

How Agent Key Management and Session Timeouts are Coordinated

SiteMinder Web Agents use a key to encrypt and decrypt any cookies that pass between Web Agents in a SiteMinder environment. All keys must be set to the same value for all Web Agents communicating with a Policy Server.

A SiteMinder installation can be configured to use dynamic Agent keys that change on a periodic basis. Dynamic key rollover lets you update dynamic keys at regular intervals to ensure the security of encrypted cookies. You specify when key rollovers occur in the Set Rollover Frequency dialog box of the Administrative UI. Key updates across a SiteMinder installation can take up to three minutes.

You must coordinate the updating of keys together with session timeouts or you may invalidate cookies that contain session information. This coordination is critical because the person designing policies in your organization may be different than the person configuring dynamic key rollover.

Session timeouts must be less than or equal to two times the interval configured between Agent key rollovers. If an administrator configures an agent key rollover to occur two times before a session expires, cookies written by the Web Agent before the first key rollover will no longer be valid. If a session timeout is greater than the specified rollover interval, a user may be re-challenged for their identification before their session terminates.

For example, if you configure key rollover to occur every three hours, you might want to set the Maximum Session timeout for 6 hours to ensure that multiple key rollovers do not invalidate the session cookie.

How a User Session Ends

A user session can end in one of three ways:

1. A user logs out.

   A user can terminate his or her own session by logging out.

2. The session timeouts expire.

   A user's session can expire because of configured session timeouts or policy-based response attributes, requiring the user to re-enter his or her credentials to begin a new session.

3. A user is disabled.

   A System Manager can disable a user account, flushing the session account and preventing that user from re-authenticating. For more information, see the *Policy Server Administration Guide*.

Windows User Security Context

In a Windows network, a security context defines a user identity and authentication information. Web applications such as Microsoft Exchange Server or SQL Server need a user security context to provide native security in the form of Microsoft access control lists (ACLs) or other access control tools.

Note: In a Windows security context, the user store must be Active Directory (AD).

The SiteMinder Web Agent can provide a Windows user security context for accessing web resources on IIS Web Servers (Windows 2000 platforms). By establishing a user security context, the server can use this identity to enforce access control mechanisms.

By providing a Windows user security context, SiteMinder offers the following benefits:

- Two levels of security

  You can protect web resources using all of SiteMinder's capabilities with the additional benefit of working with Microsoft security tools to protect applications.

- The Web Agent can communicate with various web browsers and still provide a Windows security context.

  SiteMinder can work with any browser that supports HTTP 1.x requests, HTTP cookies, and where appropriate SSL/TLS

- The Agent can collect user credentials over SSL connections, then allow subsequent requests over non-SSL connections. SiteMinder combines this mechanism for credential collection with the Windows security context when permitting access to Web resources.

- SiteMinder can implement single sign-on using forms-based authentication, and can pass on a user identity to back-office applications.

  Microsoft does not provide general-purpose forms credential collection.

How Persistent Sessions for User Security Contexts Are Maintained

For the web agent to provide a security context for specific resources, enable persistent sessions for all realms that include those resources. The session store maintains persistent sessions.

Note: For conceptual information about persistent sessions, see Persistent and Non–persistent Sessions (see page 103). For instructions on enabling persistent sessions for realms protected by web agents, see Configure a Realm (see page 507).

The session store, which resides on the same system as the Policy Server, stores the encrypted credentials of a user and associates the user with a session ID. When a SiteMinder session is established between a client and a web agent, the Windows user account is established and linked to the session. If the user session cache of the web agent becomes full and entries are purged, the agent can retrieve the credentials of the user from the session store and re–establish the session. The session store also stores the security context because this context must be propagated across a single sign–on environment.

How Sessions Are Revalidated

You can explicitly configure the web agent, working through the Policy Server, to contact the session store to revalidate a session. A session cookie stored in the browser of the user contains the session ID. The cookie uses the session ID to reacquire the credentials of the user from the session store.

Note: If the session cookie becomes invalid, the credentials associated with the ID also become invalid, and the user must reauthenticate.

A configurable value, validation period, determines the frequency at which the agent revalidates a session. The validation period defines how long the agent can keep a session active using the information in its cache before contacting the session store for updates.

If the validation period is too small, the agent goes back to the session store frequently, slowing down SiteMinder performance when processing requests. Therefore, you want to set the validation value to a high number. If the number of active sessions is lower than the maximum user session cache value of the agent, the agent uses the cached information instead of contacting the session store.

Note: If the session store is not operating, the validation period is infinite and the agent does not contact the session store.

For instructions on configuring the validation period, see Configure a Realm (see page 507).

Windows User Security Context Requirements

Your IIS Web server environment must meet the following requirements to use the Windows security context feature:

- All IIS servers have to be trusted in the domain in which the user is authenticating. You can establish trust relationships among servers that provide distributed services for groups of users. To set up trust relationships, see Microsoft server documentation.
- Users must have the privileges to log on locally to the Web server. If there are multiple servers, users need the right to log on locally to all servers.

  To enable a user to log on locally, configure this feature through the IIS Administrative Tools. See Microsoft documentation for instructions.
- If you are using a specific domain account and not the system account when starting the World Wide Web Publishing Service, you must set the user's account privileges to act as part of the operating system for the domain account running the service. This feature is configured through the Administrative Tools. See Microsoft documentation for detailed instructions.

Configuration Overview

Step	SiteMinder component	Supporting documentation
Designate a relational database as the session store.	Data tab of SiteMinder Policy Server Management Console	SiteMinder Policy Server Administration Guide
Enable the user directory containing the Windows users to run the security context.	Use Authenticated User's Security Context check box on the Directory Setup group box on the User Directory pane	User Directories (see page 157)
Associate one of the Supported Authentication Schemes (see page 115) with each realm	Resource group box of the Realm pane	Configure a Realm (see page 507)
Enable persistent sessions for each realm and set a high Validation Enabled Period	Session group box of the Realm pane	Configure a Realm (see page 507)

Step	SiteMinder component	Supporting documentation
If your IIS Web server is not in a physically secure location, set insecureserver to YES	Agent Configuration Object or WebAgent.conf	SiteMinder Web Agent Configuration Guide

Supported Authentication Schemes

The following authentication schemes are supported:

- Basic Authentication Schemes
- Basic Over SSL Authentication Schemes
- HTML Forms Authentication Schemes
- X.509 Client Certificate and Basic Authentication Schemes
- X.509 Client Certificate and HTML Forms Authentication Schemes

The Web Agent cannot use certificates alone because the user's credentials are not provided. Certificate authentication must be combined with basic or forms to collect the user credentials.

You may want to use the NTLM authentication scheme to provide access to resources in a particular realm, such as those on an intranet. CA does not support selecting the NTLM authentication scheme as part of the Windows User Security Context configuration described in this section. However, you can use NTLM for some resources on a Web server and the Windows User Security Context mechanism (with a supported authentication scheme) for different resources on the same Web server. In that case, the resources accessed using NTLM need to be in a different realm than the resources accessed through the Windows User Security Context mechanism described in this section.

More information:

Authentication Schemes (see page 301)

Active Directory and NetBIOS Names

Although Active Directory domains are named according to DNS naming standards, a NetBIOS name must still be defined when creating Active Directory domains.

For SiteMinder to support seamless Microsoft security context, NetBIOS names should be named the same as the DNS domain name as recommended by Microsoft.

When the Active Directory domain has a DNS name that is different from its NetBIOS name, the user domain cannot be established for the web server to provide the security context when SiteMinder is configured to use an LDAP user directory.

Single Sign-on in Security Context

SiteMinder will provide single-sign-on across all supported web servers while preserving the security context required for seamless Microsoft security context.

However, this requires that any realm that may cause the user to be authenticated (and establish the SiteMinder session) be configured as “Persistent”. If the user authenticates in a realm that is not persistent, the user will be challenged again when SiteMinder needs to persist the security context.

Chapter 7: Agents and Agent Groups

This section contains the following topics:

Trusted Hosts for Web Agents

A trusted host is a client computer where one or more SiteMinder Web 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. The following are the two available operation modes:

Failover Mode

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 Mode

Round robin mode is a dynamic load balancing mode in which Agents balance their requests among all the Policy Servers listed in the Host Configuration Object in a round-robin fashion.

In road robin 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.

We recommend this setting because dynamic load balancing produces better throughput when using multiple Policy Servers, resulting in more efficient user authentication and authorization. Dynamic load balancing also prevents a single Policy Server from becoming overloaded with requests. Failover still occurs if one of the load balancing Policy Servers is not available.

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 mode, set the EnableFailover parameter to YES.
   - To enable round robin dynamic 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 SiteMinder environment grows exists in the *Policy Server Administration Guide.*

Delete Trusted Host Objects

You delete a trusted host when you are reregistering it. You reregister a host using the smreghost registration tool. This tool is installed with the Web Agent.

Note: You can run the Web Agent Configuration Wizard to reregister a trusted host, but delete or rename the SmHost.conf file or the Wizard does not prompt you to register a trusted host. We recommend that you use the smreghost tool. For more information about registering and reregistering trusted hosts, see the *Web Agent Installation Guide*.

To delete a trusted host

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

   The Trusted Hosts page appears.
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

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

5. Click Delete Trusted Host.

   You are prompted to confirm the deletion of the host.

6. 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

To create a Host Configuration object, we recommend you 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 exists.

To copy a host configuration object

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

   The Host Configuration Objects page appears.

3. Click Create Host Configuration.

   The Host Configuration Search page appears.

4. Select Create a copy of an object of type Host Configuration and click OK.

   The Create Host Configuration page appears.

   Note: By default, the DefaultHostSettings Host Configuration Object is selected.

5. Enter a name and description.

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

6. In Configuration Values and Clusters, modify the properties that are different for the new object.

7. Click Submit.

   The Host Configuration Object is created.

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, add the Policy Servers to a Host Configuration object.

Note: If you are using a hardware load balancer to expose Policy Servers as multiple virtual IP addresses (VIPs), we recommend that you configure those VIPs in a failover configuration. Round robin load balancing is redundant as the hardware load balancer performs the same function more efficiently.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object.

To add a Policy Server to a Host Configuration object

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

   The Host Configuration Objects page appears.

3. Click Create Host Configuration and then click OK.

   The Create Host Configuration page appears.

   Note: By default, the Create a new object of type Host Configuration option is selected.

   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.
5. Specify the Host Configuration settings.
6. In the Clusters group box, click Add.

   The Add Cluster page appears.

7. Type the IP address and port number of the Policy Server that you want to add to the cluster.

   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 next to Port. To change the sequence of Policy Servers in the cluster, click the up and down arrows.

8. Click OK.

   The cluster is added.

   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 under Clusters, and repeat steps 6 and 7.

9. Enter the percentage of failover threshold.

   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 Host Configuration Object is updated with Policy Server details.

More information:

Operation Mode (see page 118)
Host Configuration Objects for Trusted Hosts (see page 121)

Configure Policy Server Clusters for a Host Configuration Object

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

Note: If you are using a hardware load balancer to expose Policy Servers as multiple virtual IP addresses (VIPs), we recommend that you configure those VIPs in a failover configuration. Clustering is redundant because the hardware load balancer performs the same function more efficiently.

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

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object.

To configure a cluster

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

   The Host Configuration Objects page appears.
3. Click Create Host Configuration and then click OK.

   The Create Host Configuration page appears.

   Note: By default, the Create a new object of type Host Configuration option is selected.

   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.
5. Specify the Host Configuration settings.
6. In the Clusters section, click Add.

   The Add Cluster page appears.
7. Type the IP address and port number of the Policy Server that you want to add to the cluster.

   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 next to Port. To change the sequence of Policy Servers in the cluster, click the up and down arrows.
8. Click OK.

   The cluster is added.

   Note: To modify a cluster, click the right-facing arrow in the first column. To delete a cluster, click the minus sign in the last column. To add another cluster, click Add, and repeat steps 7 and 8.

9. Type a percentage for the Failover Threshold.

   SM automatically calculates the Failover Threshold value that is based on both the number of Policy Servers configured in the cluster and the Failover Threshold Percent you specify. The Failover Threshold value (displayed in the column to the right of the list of servers in each cluster) is the total number of Policy Servers that must be active in the Cluster to consider the Cluster available based on your configuration. If the number of Active Servers in the Cluster falls below the displayed Failover Threshold value, the Cluster fails over to the next available Cluster in the list. If the calculation of the Failover Threshold value that is based on your configuration would result in a non-whole number, the Policy Server rounds the value up to the nearest integer.

   For example, consider a Cluster with 5 Policy Servers configured.

   With a Failover Threshold Percentage ...;

   ...in the range between [0% - 19%], the value of required active servers (Failover Threshold) is rounded up to 1 .

   ...of 20%, the calculated value for Failover Threshold is 1 .

   in the range between [21% - 39%], the value for Failover Threshold is rounded up to 2 .

   of 40%, the calculated value for Failover Threshold is 2 .

   in the range between [41% - 59%], the value for Failover Threshold is rounded up to 3 .

   of 60%, the calculated value for Failover Threshold is 3 .

   in the range between [61% - 79%], the value for Failover Threshold is rounded up to 4 .

   of 80%, the calculated value for Failover Threshold is 4 .

   in the range between [81% - 99%], the value for Failover Threshold is rounded up to 5 .

   of 100%, the calculated value for Failover Threshold is 5 .

   When you set the Failover Threshold Percentage, it applies to all clusters that use the Host Configuration Object.

   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 Host Configuration Object is configured with the Policy Server.

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

More information:

Duplicate Policy Server Objects (see page 54)

SiteMinder Agents Overview

An Agent in a SiteMinder environment is a network entity that acts as a filter to enforce network access control or Web access control. An Agent monitors requests for resources. When a user requests a protected resource, the Agent prompts the user for credentials based on an authentication scheme, and sends the credentials to a Policy Server.

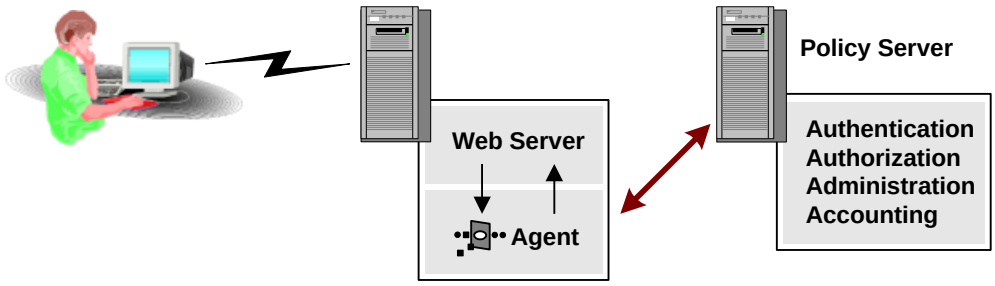
The Policy Server determines whether or not a user can be authenticated based on the credentials, and whether or not the user is authorized for the requested resource. The Policy Server then communicates with the Agent, which allows or denies access to the requested resource.

Web Agents, Affiliate Agents, EJB Agents, Servlet Agents, and RADIUS Agents are available by default. All other Agents are considered Custom Agents that must be created using the Agent APIs. Once created, you can configure Custom Agents in the Administrative UI.

Web Agents

Web Agents are SiteMinder Agents that operate with Web servers. When a user requests a page from the Web server, the Web Agent communicates with the Policy Server and processes authentication and authorization requests before the user can access the resource from a Web browser. In addition, the Policy Server can provide information that the Web Agent uses to provide personalized content based on a user's identity.

The following diagram illustrates the three most basic transactions that a Web Agent and Policy Server handle in order to provide access to a protected resource. These transactions can contain more detailed information to enable customized content and support other SiteMinder features, but the process is similar whenever a user attempts to access a resource through a Web server managed by a Web Agent.



1. Is the resource Protected? If yes, request the user's credentials.
2. Is the user authenticated? If yes, check policies for authorization.
3. Is the user authorized? If yes, allow access to the protected resource.

The previous figure assumes that a user requests a protected resource for which the user is authorized. The Web Agent checks with the Policy Server to determine if the resource is protected, and the Policy Server indicates that it is protected. The Web Agent gathers credentials from the user and communicates them to the Policy Server.

The Policy Server authenticates the user and informs the Web Agent that the user has been properly identified. Finally, the Web Agent checks with the Policy Server to determine if the user is authorized for the resource. The Policy Server verifies that the user is authorized for the resource, communicates this to the Web gent, and the Web Agent allows the Web server to display the protected resource requested by the user.

Agents that control the same resources and are of the same Agent type (all Web Agents, or all RADIUS Agents) can be grouped.

Note: If you plan to configure support for virtual Web servers, see the *Web Agent Configuration Guide*.

More information:

Agent Groups (see page 149)

SAML Affiliate Agents

The SAML Affiliate Agent is part of the legacy federation solution, which enables business partners to share user identity information.

Note: Legacy Federation and the SAML Affiliate Agent are packaged as separately licensed items.

The SAML Affiliate Agent is installed only at the consumer site. This Agent enables a site to consume assertions that are sent from a producer, facilitating seamless single sign-on between a producer and a consumer. The assertion confirms that a user has been authenticated at the producer. The consumer uses the information in the assertion to provide user information to a web server for access to resources and web applications.

If SiteMinder is at the producer, you configure legacy federation components that identify each partner for which SiteMinder can generate assertions.

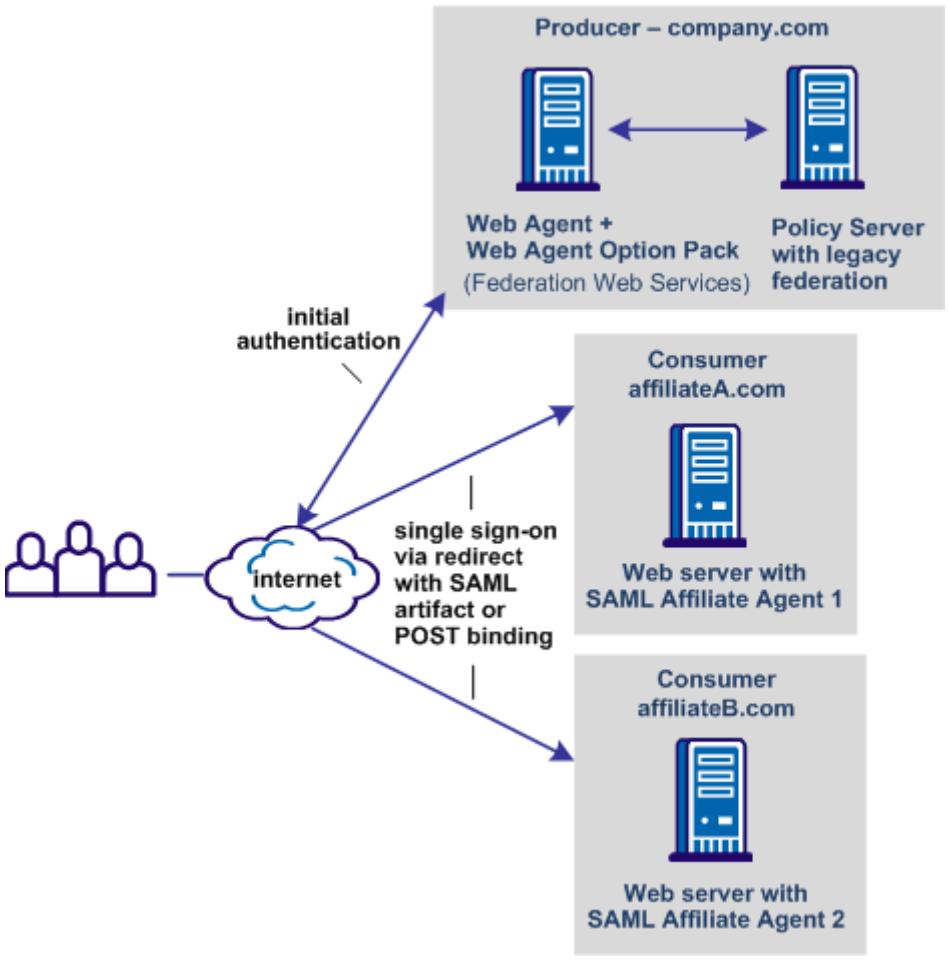
Note: In the Administrative UI, you can select an Affiliate Agent as the Agent type. This option is not for the SAML Affiliate Agent. The option is only applicable for 4.x Affiliate Agents. Do not select it.

If you install a SAML Affiliate Agent at an affiliate site, it is the only SiteMinder component that is required at that site for federated transactions. A full SiteMinder installation is not required at the consumer site because an application at the partner can enforce access control.

If SiteMinder is serving as the SAML 1.0 producer, the following producer-side components are required to support legacy federation:

- Policy Server
- Web Agent
- Web Agent Option Pack. The option pack provides the Federation Web Services application that is required to process federated transactions.

The following graphic shows a federated network with SAML Affiliate Agents.



In the pictured network, company.com has a full SiteMinder installation and can generate assertions. When a user requests a federated resource, the user initially authenticates at company.com. Company.com generates an assertion and redirects the user to the appropriate affiliate partner. At the partner, the SAML Affiliate Agent consumes the assertion and passes the necessary information to the web application. The user experiences a single sign-on transaction and gains access to the requested resource.

RADIUS Agents

Remote Authentication Dial-In User Service (RADIUS) is a protocol that enables you to exchange session authentication and configuration information between a Network Access Server (NAS) device and a RADIUS authentication server. The RADIUS protocol is often used by NAS devices that serve as proxy services, firewalls, or dial-up security devices.

A RADIUS Agent secures an entire application that communicates using the RADIUS protocol.

The Policy Server can be used as a RADIUS authentication server. RADIUS Agents allow the Policy Server to communicate with the NAS client devices.

More information:

Use the Policy Server as a Radius Server (see page 735)

Application Server Agents

The Application Server Agent is a collection of Java components that provide a full-featured SiteMinder Agent for securing WebLogic and WebSphere application server resources. The Application Server Agent integrates SiteMinder with the J2EE platform.

The Application Server Agent can protect the following components:

- Web Applications (including servlets, HTML pages, JSP, image files)
- JNDI lookups
- EJB components
- JMS connection factories, topics, and queues
- JDBC connection pools

The Application Server Agent is a single Agent, but from the perspective of the SiteMinder Policy Server, there are different Agent types that protect application server resources. The Agent types give the Application Server Agent the flexibility to protect servlets, and EJB components in two ways: using the Servlet, or EJB Agent respectively, or using the Web Agent.

Note: For more information on configuring SiteMinder to work with application server Agents, see the SiteMinder *Application Server Agent Guide* that applies to your platform.

CA SOA Security Manager SOA Agents

CA SOA Security Manager SOA Agents integrate with web and application servers to authenticate and authorize requests for access to web services resources hosted on those servers.

Note: More information about SOA Agents exists in the *CA SOA Security Manager Policy Configuration Guide*.

Web Agent Configuration Overview

The two options for configuring a Web Agent are:

Central configuration

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

The Agent configuration is specified in an Agent Configuration Object.

Note: Central configuration does not apply to RADIUS, EJB, Servlet, or Custom Agents—those Agents can only perform local configuration.

Local configuration

Indicates that the Web 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 Web Agent from the local Agent configuration file, not from the Policy Server. This is true whether you are configuring centrally or locally.

More information:

Web Agent Components (see page 132)
Combined Central and Local Configuration (see page 137)

Advantages of Centrally Configuring Web Agents

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

Compared with local configuration, central configuration provides:

Improved Usability When Using Central Agent Configuration

- Updating the configuration settings of multiple Web Agents is easier and faster. You can change the setting in one place and have it propagate to multiple Web Agents.
- Installing the Web 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 Web 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 Web 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.

Web Agent Components

On the Agent-side of a SiteMinder network, there are several main components involved in Web Agent operation:

SiteMinder Web Agent

Virtual interface to a Web Server; triggers rules and enforces policies

Trusted Host

A client computer where one or more Web 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 Web 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)

Web Agent Configuration File (WebAgent.conf or LocalConfig.conf)

Stored on the web server where the Agent resides, this file is used for local configuration. It holds the Agent configuration parameters for each Web Agent. All Web Agents use the WebAgent.conf file; however, the IIS 6.0 Web Agent uses WebAgent.conf file only for core settings needed for the Agent to start and connect to a Policy Server. For its configuration settings, the IIS 6.0 Web Agent uses the LocalConfig.conf file. There is a pointer to the LocalConfig.conf file in the IIS 6.0 WebAgent.conf file.

Host Configuration File (SmHost.conf)

Stored on the web Server where the Web 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.

More information:

Web Agent Configuration Overview (see page 131)

Policy Server Objects Related to Web Agents

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

Agent object

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

Agent Configuration Object

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

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.

More information:

Web Agent Configuration Overview (see page 131)

How to Configure a Web Agent

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

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

To configure a Web 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 Web 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 Web Agent. This setting is in the local Agent configuration file.

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

More information:

Host Configuration Objects for Trusted Hosts (see page 121)
Agent Configuration Object Overview (see page 142)
Enable a Web Agent (see page 147)

Configure Web Agents Centrally

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

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 Objects.

   The Host Configuration Objects page appears.
3. Click Create Host Configuration.
4. 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 Policy Server installation program installs the DefaultHostSettings object.

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

   The Create Host Configuration page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
6. Type the name and a description.
7. In Configuration Values, specify the Host Configuration settings.
8. Click Submit.

   The Host Configuration Object is created.

Configure Web Agents Locally

The Web 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 Web 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 Web 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 Web Agent configuration file (WebAgent.conf and/or LocalConfig.conf).

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

   For all Web Agents except IIS 6.0, there is a WebAgent.conf.sample file In the <*web_agent_home*>\config directory. You should modify this file, then save it under the name WebAgent.conf to the appropriate web server location.

   For IIS 6.0 Web Agents, this Agent uses the LocalConfig.conf file in <*web_agent_home*>\bin\IIS directory as its active configuration file. Modify this file if you want to make changes. The copy of the LocalConfig.conf file in <*web_agent_home*>\config is the original that you should not change.

   Note: If you are using an IIS 6.0 Web Agent, the main configuration file is called LocalConfig.conf. The WebAgent.conf file is still used, but only for core Agent settings that enable the Agent to start and connect to the Policy Server.

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

Combined Central and Local Configuration

When a Web 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 Web 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.

Example of Using Central and Local Configuration

Scenario:

You want to configure multiple cookie domain single sign-on across your SiteMinder network without having to configure each Agent individually.

The CookieDomain parameter in the Agent Configuration Object is set to acmecorp.com. However, you want to set the CookieDomain parameter to test.com for one Web Agent in your network, while continuing to use all of the other parameter values set in the Agent Configuration Object.

Solution:

To implement the example configuration

1. Configure an Agent Configuration Object with all the parameters applicable for your environment.
2. In the Agent Configuration Object, set the AllowLocalConfig parameter to yes.
3. For the single Web Agent, change only the CookieDomain parameter to test.com. Do not modify any other parameters.

The value for the CookieDomain parameter in the Agent configuration file overrides the value in the Agent Configuration Object, while the Agent Configuration Object determines the settings for all the other parameters.

Create an Agent Object to Establish a Web Agent Identity

To create a Web Agent identity, 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 Web server hosting the Web Agent and to associate policies with Web Agents correctly. Creating a Web Agent object and identity lets you associate the Web Agent with a realm.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object.

To create a Web Agent object and identity

1. Click Infrastructure, Agent.
2. Click Agents

   The Agents page appears.
3. Click Create Agent.

   Verify that the Create a new object of type Agent option is selected.
4. Click OK.

   The Create Agent page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Type the name and description of the Agent.

   Note: Web Agent names have the following 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.
6. Select SiteMinder as the Agent Style and Web Agent as the Agent Type.
7. Click Submit.

   The Web Agent Object is created.

More information:

Realms (see page 501)
Duplicate Policy Server Objects (see page 54)

Configure an Agent Object for a 4.x Web Agent Identity

To create a 4.x Web Agent identity, create an Agent object in the Administrative UI. The object name must match the Agent name in the local Web Agent configuration file. For descriptions of the configuration parameters, see the *Web Agent Configuration Guide*. Creating a Web Agent object and identity lets you associate the Web Agent with a realm.

Important! You will receive correspondence from CA Technologies regarding the end date for 4.x Web Agent support.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object.

To create a 4.x Web Agent object and identity

1. Click Infrastructure, Agent.
2. Click Agents.

   The Agents page appears.
3. Click Create Agent.

   Verify that the Create a new object of type Agent option is selected.
4. Click OK.

   The Create Agent page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Type the name and description of the Agent.

   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.
6. Confirm the following:
   - That the SiteMinder option is selected.
   - That Web Agent appears in the Agent Type drop-down list.
7. Select the Supports 4.x agents option.

   The Trust Settings page appears.

8. Enter the IP Address of the server on which the Agent resides.

   Note: Like a single server, virtual servers have defined names and IP addresses. Each Agent on a virtual server must have a unique Agent name.

9. Type and confirm a shared secret.

   Limits:

   - The secret that you supply must match the secret that was assigned when the Web 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.

   Note: Virtual servers on the same Web server must share the same secret. When a 4.x Agent attempts to connect to the Policy Server, the Agent and Policy Server use the shared secret for mutual authentication.

10. Click Submit.

    The 4.x Web Agent object is created.

More information:

Realms (see page 501)
Duplicate Policy Server Objects (see page 54)

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 Web 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 Web Agent, refer to the *Web Agent Configuration Guide* for parameter descriptions, the default values, and instructions on setting the parameters.

More information:

Agent Configuration Object Overview (see page 142)

Agent Configuration Object Overview

An Agent Configuration Object holds the parameters that define the Web Agent configuration. The configuration object is the Policy Server counterpart to Web Agent configuration file.

When you configure a Web Agent, you are prompted for the name of the Agent Configuration Object. The configuration object is associated with the Web Agent that you are configuring.

For more information about the initial configuration of a Web Agent and the required Web Agent parameters, see the *Web Agent Installation Guide*.

More information:

Required Agent Configuration Object Parameters (see page 144)

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 Web Agents that SiteMinder supports. After the new Agent configuration object is created, you can modify its list of parameters and their values.

Note: For more information about parameters and default settings, see the *Web Agent Configuration Guide*.

To copy an Agent Configuration Object

1. Click Infrastructure, Agent.
2. Click Agent Configuration Objects.

   The Agent Configuration Objects page appears.
3. Click Create Agent Configuration.

   The Create Agent Configuration page appears.
4. Select the Create a copy of an object of type Agent Configuration option, and do one of the following:
   - Select one of the default Agent Configuration Objects from the list.
   - Click Search and select one from the list of Agent Configuration Objects that is displayed.
5. Click OK.

   The Create Agent Configuration: page appears.

6. Enter a new name and description and click Submit.

   You are prompted to specify a parameter or continue entering a comment.

7. Enter a comment and click Yes.

   The Agent Configuration Object based on the default Agent Configuration Object is created.

Create an Agent Configuration Object

The Policy Server installer creates several Agent Configuration Objects for several supported web servers that contain default settings. These sample objects have names, such as IISDefaultSettings or iPlanetDefaultSettings. You can duplicate these default objects and use them as templates for your Agent configuration.

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, enter all of your parameters and their values manually.

To create an Agent Configuration Object

1. Click Infrastructure, Agent.
2. Click Agent Configuration Objects.

   The Agent Configuration Objects page appears.

3. Click Create Agent Configuration.

   Verify that the Create a new object of type Agent Configuration option is selected.

4. Click OK.

   The Create Agent Configuration page appears.

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

5. Type the name and a description of the Agent.
6. Click Add.

   The Create Parameter page appears.

7. Type the ame and value of the parameter.

8. (Optional) Do *one* of the following:
   - To encrypt the value of the parameter, select the Encrypted option.
   - To add multiple values for a parameter, do the following:
     a. Select the Multi-value option.
     b. Type a value for the parameter.
     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 page. 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.
9. Click OK.

   The parameter is added to the list.

   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.
10. Click Submit.

    The Agent Configuration Object is created.

Required Agent Configuration Object Parameters

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

- All Agents require DefaultAgentName
- IIS Agents require DefaultUserName and DefaultPassword
- Domino Agents require DominoDefaultUser and DominoSuperUser

Note: More information about these parameters exists in the Web Agent Installation Guide.

Specify the IIS Proxy User

Configuring an IIS Web Agent 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 SiteMinder. When users want to access resources on an IIS Web server protected by SiteMinder, they may not have the necessary server access privileges. The Web 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 SiteMinder.

Specify the Domino Users

Configuring a Web Agent 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 SiteMinder against another user directory, this is the name by which the Domino Web Agent identifies that user to the Domino server. This value can be encrypted.

DominoSuperUser

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

Specify the Agent Name

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

DefaultAgentName

Identifies the Agent identity that the Web 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 Web 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: For more information about parameters and default settings, see the *Web Agent Configuration Guide*.

To modify an Agent configuration object

1. Click Infrastructure, Agent.
2. Click Agent Configuration Objects.

   The Agent Configuration Objects page appears.
3. Specify search criteria, and click Search.

   A list of Agent configuration objects that match the search criteria appears.
4. Click the name of the Agent Configuration object that you want to modify.

   The View Agent Configuration page appears.

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

   The settings and controls become active.
6. Click the right arrow to the left of the parameter's name.

   The Edit Parameter page appears.
7. Edit the value of the parameter.

   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 page.
8. (Optional) Select the Encrypted option.

   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 page.

9. Click OK.

   The value of the parameter is updated.

   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 5 through 8.

10. Click Submit.

    The Agent configuration object is modified.

Enable a Web Agent

Regardless of whether you configure a Web Agent centrally or locally, you can only enable and disable a Web Agent from the WebAgent.conf 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 Web Agent, then you can enable it.

Note: More information about the WebAgent.conf file and enabling a Web Agent exists in the *Web Agent Configuration Guide*.

Configure a RADIUS Agent

To create a RADIUS Agent identity, create an Agent object in the Administrative UI. The object name must match the Agent name that was specified when the Agent was installed. If the Agent name is changed in the Web Agent Management Console or in the WebAgent.conf file, the object name must be changed also. The Policy Server uses the RADIUS Agent identity to communicate with the NAS client devices. Creating a RADIUS Agent object and identity lets you associate the RADIUS Agent with a realm.

Once you create a RADIUS Agent identity, you can install and configure a SiteMinder Agent on a RADIUS client or application server. You can configure only the RADIUS Agents locally.

Note: For more information, see the *Web Agent Installation Guide*.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object.

To configure a RADIUS Agent

1. Click Infrastructure, Agent.
2. Click Agents.

   The Agents page appears.
3. Click Create Agent.

   Verify that the Create a new object of type Agent option is selected.
4. Click OK.

   The Create Agent page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Type the name and a description of the Agent.

   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.
6. Select the Radius option, and then select a RADIUS vendor.

   The Trust Settings and Radius Settings fields appear.

   Note: You can select Generic RADIUS as the Agent Type to protect any type of RADIUS device. However, Generic RADIUS Agent types do not provide access to vendor-specific response attributes.
7. Specify the following trust settings:

   - Type an IP Address of the RADIUS client (NAS device).
   - Type and confirm an alphanumeric shared secret.
8. Type the number of the RADIUS realm hint.

   Note: For more information about realm hints, see the *Policy Server Administration Guide*.
9. Click Submit.

   The RADIUS Agent is configured.

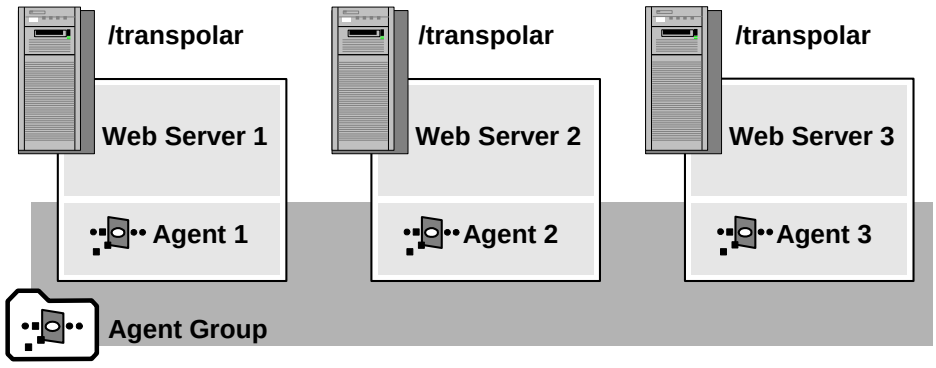
More information:

Duplicate Policy Server Objects (see page 54)

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 web servers/Web Agents. It also saves time because you define only one policy for all of the Web Agents. For example, you can use an Agent group to protect a group of Web Servers that use round robin processing to supply access to the same resources.

In the following figure, you can protect the Web farm with one set of policies, and create an Agent group that is bound to the set of policies.



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 SiteMinder, 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.

More information:

Policies (see page 561)

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 that you are creating an object. You can also copy the properties of an existing object to create an object.

To configure an Agent group

1. Click Infrastructure, Agent.
2. Click Agent Groups.

   The Agent Groups page appears.
3. Click Create Agent Group.

   Verify that the Create a new object of type Agent Group option is selected.
4. Click OK.

   The Create Agent Group page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Type the name and a description of the Agent group.
6. Select the Agent Style and Agent Type.

   Note: An Agent group can only contain one type of agent, for example, all Web Agents, all Affiliate Agents, or all RADIUS Agents.
7. Click Add/Remove.

   The Agent Group Members page appears.

   Note: Only Agents of the specified Agent type are listed under Available Members. For example, if the specified Agent Style is Radius and the specified Agent Type is 3-Com, only 3-Com Agents are listed. If the specified Agent Type is Generic Radius, all RADIUS Agents are listed.
8. Select one or more Agents from the list of Available Members, and click the right-facing arrow.

   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 the additional members. To select a block of members, click the first member and then hold down the Shift key while you click the last member in the block.
9. Click OK.

   The selected Agents are added to the Agent group.
10. Click Submit.

    The Agent group is configured.

More information:

Duplicate Policy Server Objects (see page 54)

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.
2. Click Agent Groups.

   The Agent Groups page appears.
3. Specify search criteria and click Search.

   A list of agent groups that match the search criteria appears.
4. Click the name of an Agent group that you want to modify.

   The View Agent Group page appears.
5. Click Modify.

   The settings and controls become active.
6. Click Add/Remove.

   The Agent Group Members page appears.

   Note: Only Agents of the specified Agent type are listed under Available Members. For example, if the specified Agent Style is Radius and the specified Agent Type is 3-Com, only 3-Com Agents are listed. If the specified Agent Type is Generic Radius, all RADIUS Agents are listed.
7. 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 the additional members. To select a block of members, click the first member and then hold down the Shift key while you click the last member in the block.
8. Click OK.

   The selected agents are listed under Group Members in the Modify Agent Group page.
9. Click Submit.

   The selected agents are added to the Agent group.

Custom Agents

You create a custom agent using either the C or Java version of the SiteMinder Agent API. The SiteMinder Agent API and associated documentation are provided by the Software Development Kit, which is available separately.

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. For example, if you developed a custom FTP Agent, you would then need to define an FTP Agent type.

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, 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 that you are creating an object. You can also copy the properties of an existing object to create an object.

To create an Agent object

1. Click Infrastructure, Agent.
2. Click Agents.

   The Agents page appears.
3. Click Create Agent.

   Verify that the Create a new object of type Agent option is selected.
4. Click OK.

   The Create Agent page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Type the name and a description of the Agent.

   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.
6. Select SiteMinder as the Agent Style, Web Agent as the Agent Type, and the Supports 4.x agents option.

   The Trust Settings fields appear.
7. Type the IP Address of the server on which the Agent resides.
8. Type and confirm a shared secret.

   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.
9. Click Submit.

   The Agent object is created.

More information:

Configure a Custom Agent Type (see page 152)
Duplicate Policy Server Objects (see page 54)

Resource Protection with a SiteMinder Agent

You associate a configured Agent with a realm, which is a collection of resources that you want to protect. Realms are protected by rules, which get included in an access control policy.

Agent Discovery Introduced

Agent Discovery lets you discover instances of different types and versions of SiteMinder agents. Once discovered, you can view agent-specific details such as version, state, and so on. You can also view a list of agents deployed on various hosts in your enterprise and delete the unwanted agent instance entries from the list.

Note: Agent Discovery does not support traditional agents.

Heartbeat messages help identify the r12.5 agents. The frequency with which an agent sends heartbeat messages to the Policy Server is known as heartbeat interval. The MinTimeBetweenAgentStatusUpdates parameter that denotes the heartbeat interval, controls the minimum time that has to elapse before a Policy Server updates the Agent Instance objects in the store. Any heartbeat message received by the Policy Server before the minimum time elapses, is ignored. The default value of this parameter is 24 hours and the minimum value is 1 hour. An Agent identifies itself on startup regardless of whether the time set for the MinTimeBetweenAgentStatusUpdates parameter has elapsed.

Agent Discovery identifies all agents that communicate with the Policy Server regularly, irrespective of the version of the agent. To identify agents prior to r12.5, Agent Discovery uses a combination of the IP address and trusted host of an agent. Any change in this combination for an agent results in multiple entries for the same agent.

Agent Discovery does not identify the ASA agents prior to r12.5, because these agents do not send any information to the Policy Server. The attributes of an ASA agent are displayed as unknown in the Agent Instances list. If the ASA agent host has another agent prior to r12.5 using the same host and trusted host combination, the ASA entry is hidden from the list.

You can view a list of agent instances that are deployed in your enterprise using Agent Instances under Infrastructure in the Administrative UI.

Note: For agents prior to r12.5, the IP address is displayed as the host name.

List Agent Instances

You can list the agent instances to know the number of agents deployed across your enterprise. You can also view a list of agent instances based on the search criterion you provide. The search criterion can be any attribute of an agent instance.

From the list, you can view all the relevant details of an agent instance and delete one or more agent instance entries. If you delete an agent prior to r12.5, none of the agent attributes are displayed in the list, except the trusted host. All the attributes of the deleted agent appear in the list again, whenever the agent restarts.

Note: Even after you delete an agent instance entry, the agent instance remains active so long as it communicates with the Policy Server regularly.

You can sort any attribute of an agent instance to view the list of agents instances that use the same attribute you selected.

To list agent instances

1. Click Infrastructure, Agent.
2. Click Agent Instances.

   All the discovered agent instances in your enterprise are displayed.
3. Click the icon under Show Details to view agent-specific details.

   The agent-specific details appear below the agent instances list table.

   Note: For agents prior to r12.5, the configuration mode is either FIPS Only or Compact. The Administrative UI displays the details based on the actual mode being used in server-agent communication.
4. (Optional) Specify the search criterion in the Search field.

   The agent instances list is filtered based on the search criterion.
5. (Optional) Click the arrow mark in the table heading of an attribute to sort the agent instances list based on the selected attribute.

Configure the Policy Server Heartbeat Interval

Heartbeat interval helps Agent Discovery determine the state of an agent. When the agent does not send any heartbeat message beyond 24 hours, the state of the agent instance changes to inactive. You can configure the heartbeat interval to change the default value, which is 24 hours.

To configure the Policy Server Heartbeat Interval using XPSConfig

1. Open a command prompt in the computer hosting the Policy Server.
2. Enter the following command:
   `XPSConfig`

   The Products Menu opens.
3. Enter SM.

   The Parameters Menu opens and lists the SiteMinder parameters.
4. Enter the value against the MinTimeBetweenAgentStatusUpdates option.

   The MinTimeBetweenAgentStatusUpdates Parameter Menu opens.
5. Enter C to change the value.
6. Enter the new value.

   Default: 24 hours

   Minimum: 1 hour
7. Enter Q three times to exit the MinTimeBetweenAgentStatusUpdates Menu, Parameters Menu, and SiteMinder Menu and return to the command prompt.

Chapter 8: User Directories

This section contains the following topics:

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.

You configure directory connections to resolve how the Policy Server establishes a context for user identities. The Policy Server uses these connections to verify user identities and retrieve user attributes contained in the user stores.

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

- LDAP
- ODBC
- Oracle
- Custom

A list of supported directories types exists in the SiteMinder platform support matrix.

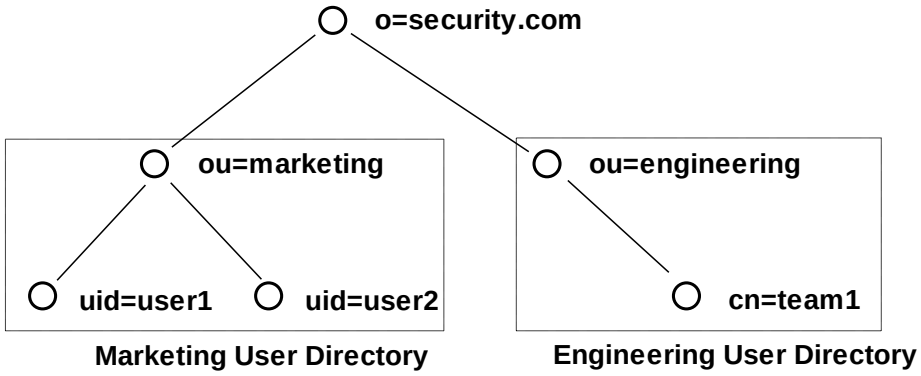
Note: If you are trying to configure or upgrade a SiteMinder store listed in the SiteMinder Platform Support Matrix and cannot find the procedures in this guide, see the *Directory Configuration Guide*.

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.

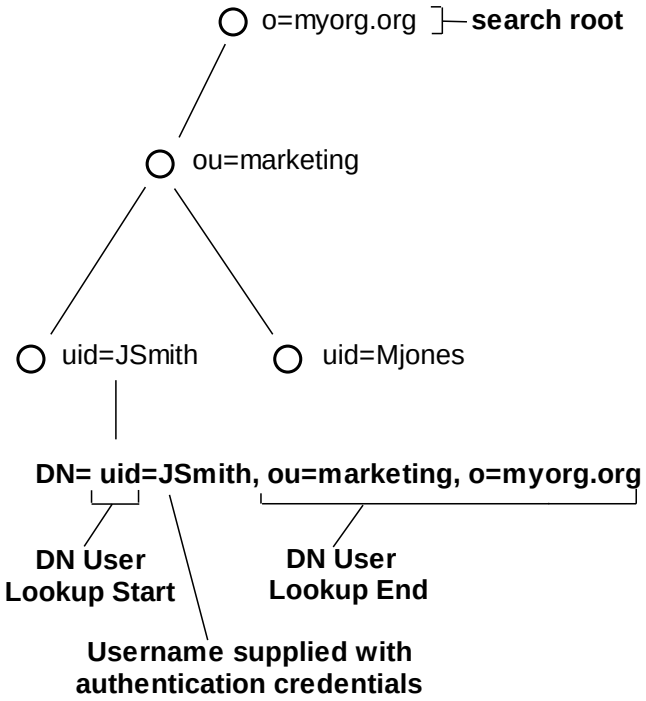
User Lookup by DN

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

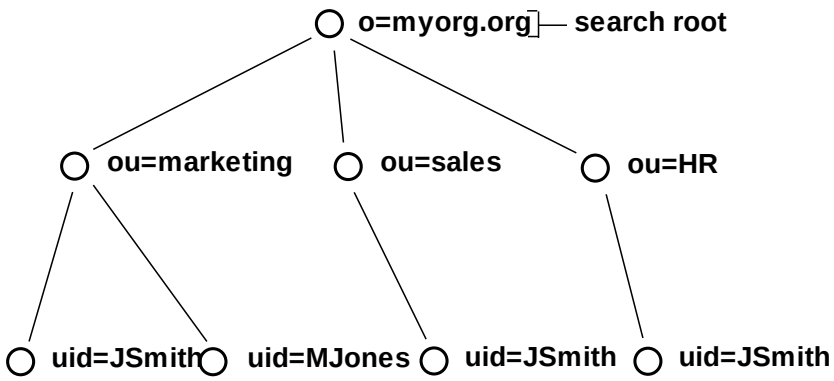
An LDAP directory server may contain numerous users in complicated DITs, and it may not be practical to create a large number of user directory connections. Your organization may have hundreds of organizational units and you may want to avoid having end users log in with detailed string representations. 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.
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.

Search Filter	Format	Description
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 SiteMinder 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

SiteMinder 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 SiteMinder 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 SiteMinder deployment, you can specify a connection to a slave LDAP directory. If you use any of SiteMinder's features that require data to be written to the LDAP directory, SiteMinder automatically detects referrals that point to a master LDAP directory. The information that SiteMinder 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, SiteMinder is able to use the read referrals to retrieve information from the associated directories.

More information:

Enhanced LDAP Referral Handling (see page 167)

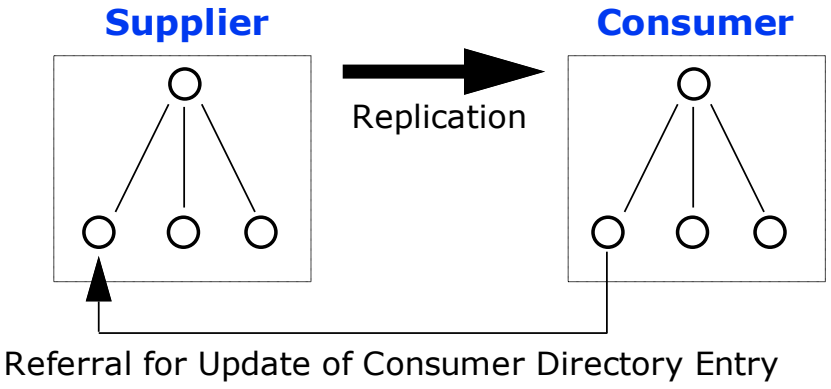
Directory Topology and LDAP Referrals

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:

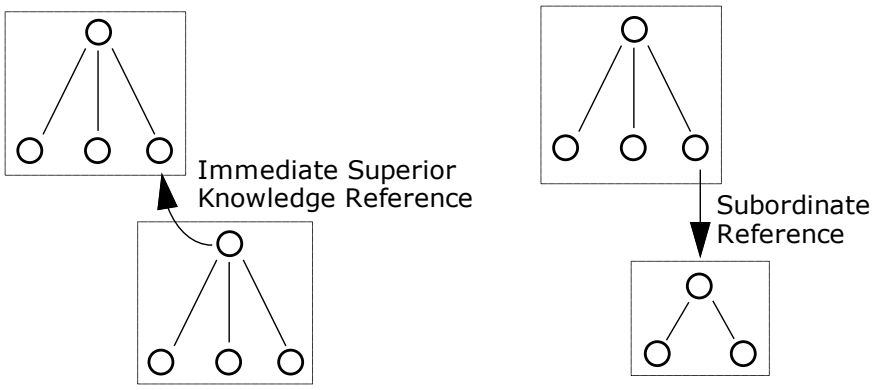
Replication Agreements

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.



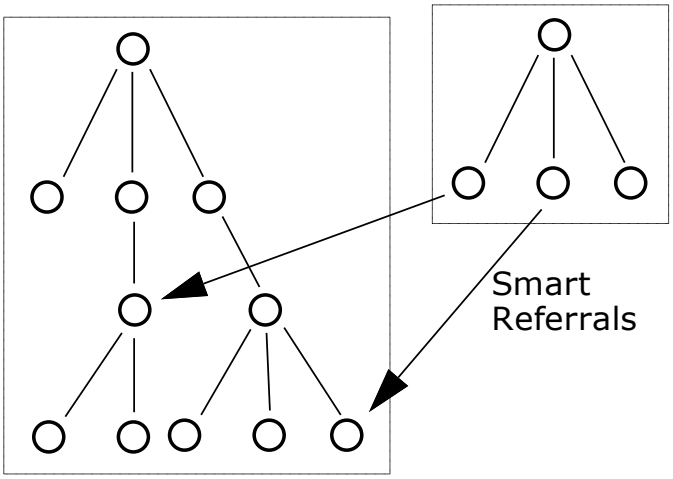
Knowledge References

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.

SiteMinder 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 SiteMinder to become saturated with requests.

For example, a SiteMinder 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

SiteMinder 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 SiteMinder SQL Query Scheme pane. This pane is accessible from the SiteMinder 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.

Active Directory Overview

SiteMinder supports user directories on the Microsoft Active Directory platform. Although the configuration for Active Directory (AD) and LDAP namespaces in the Administrative UI is 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.

Note: If the Policy Server is installed on a UNIX operating system, you cannot use the AD namespace for connecting to the AD user store.

Custom Directory Overview

The Administrative UI allows you to create custom user directory connections by creating shared libraries using the SiteMinder 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 SiteMinder APIs, you can configure a custom namespace on the User Directory pane.

Directory Attributes Overview

Some SiteMinder features require read or read/write access to seven SiteMinder 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 SiteMinder 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, SiteMinder 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 SiteMinder 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 SiteMinder 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 SiteMinder passes to protected applications to maintain a user's identity. This feature is a bridge between SiteMinder and legacy applications, which often use attributes to identify a user. The universal ID is also used in configuring Directory mapping.

Attribute Name	Data Type	Directory Types	Description
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 SiteMinder 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 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 243)
Password Policies (see page 667)
Anonymous Authentication Schemes (see page 307)
User Attribute Mapping (see page 258)

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.

Follow these steps:

1. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Complete the required connection information in the General and Directory Setup areas.

   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. Configure the LDAP search and LDAP user DN lookup settings in the LDAP Settings area.
6. Do the following in the Administrator Credentials area:
   a. Select Require Credentials.
   b. Enter the credentials of an administrator account.
7. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder in the User Attributes area.
8. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.
9. Click Submit.

   The user directory connection is created.

More information:

LDAP Load Balancing and Failover (see page 228)
Define an Attribute Mapping (see page 261)

Enable User Store DSA Parameters

SiteMinder 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.

Follow these steps:

1. Open the .dxi file for the user store DSA.
2. Define the following entries 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 .dxi 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 SiteMinder 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 SiteMinder 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.

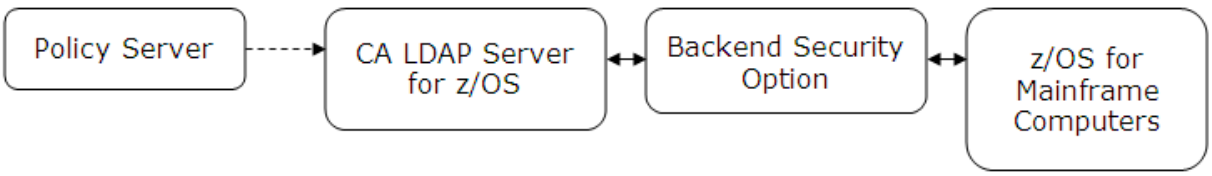
How to Configure a CA LDAP Server for z/OS User Directory Connection

You can use CA LDAP Server for z/OS as a SiteMinder user store. The following process lists the steps for configuring the user store connection to the Policy Server:

1. Review the following information:
   - CA LDAP Server for z/OS overview.
   - SiteMinder features not supported by CA LDAP Server for z/OS.
   - CA Top Secret r12 (TSS) backend security option and its objectclass hierarchy.
   - CA LDAP Server r15 for z/OS (ACF2) backend security option and its objectclass hierarchy.
   - CA LDAP Server r15 for z/OS (RACF) backend security option and its namespace hierarchy.
2. Configure the Policy Server registry entries for TSS, ACF2, or RACF.
3. Configure the user directory connection.

CA LDAP Server for z/OS Overview

You can configure a CA LDAP Server for z/OS as a user store by configuring a connection from the Policy Server to the LDAP Server. How you configure the connection from the Policy Server to the LDAP Server depends on the backend option that you are using to secure the LDAP Server:



CA supports the following backend security options for CA LDAP Server:

- CA Top Secret r12 (TSS)
- CA LDAP Server r15 for z/OS (RACF)
- CA LDAP Server r15 for z/OS (ACF2)

Become familiar with the objectclass hierarchy for these backend security options before configuring the connection from the Policy Server to the LDAP Server. Also, add the backend-related objectclasses to the Policy Server registries in the LDAP namespace.

Note: z/OS is an IBM operating system for mainframe computers.

SiteMinder Features Not Supported by CA LDAP Server for z/OS (TSS)

CA LDAP Server for z/OS does not support the following SiteMinder features:

Password Services

Password Services is not supported.

Anonymous Binds

When configuring a CA LDAP Server r15 for z/OS as a user store, provide values for the Administrator Credentials in the Create User Directory page.

Characters Not Supported in User Names

The following characters are not supported in user names:

- Space
- Single quote
- Opening parenthesis
- Closing parenthesis
- Comma
- Backslash

User Groups and Policies

Adding a user group to a policy and attempting to authorize a user in that group fails.

Load Balancing and Failover (For TSS)

Load balancing and failover is not supported.

LDAP Failover and Replication (For RACF and ACF2)

LDAP Failover and Replication is not supported.

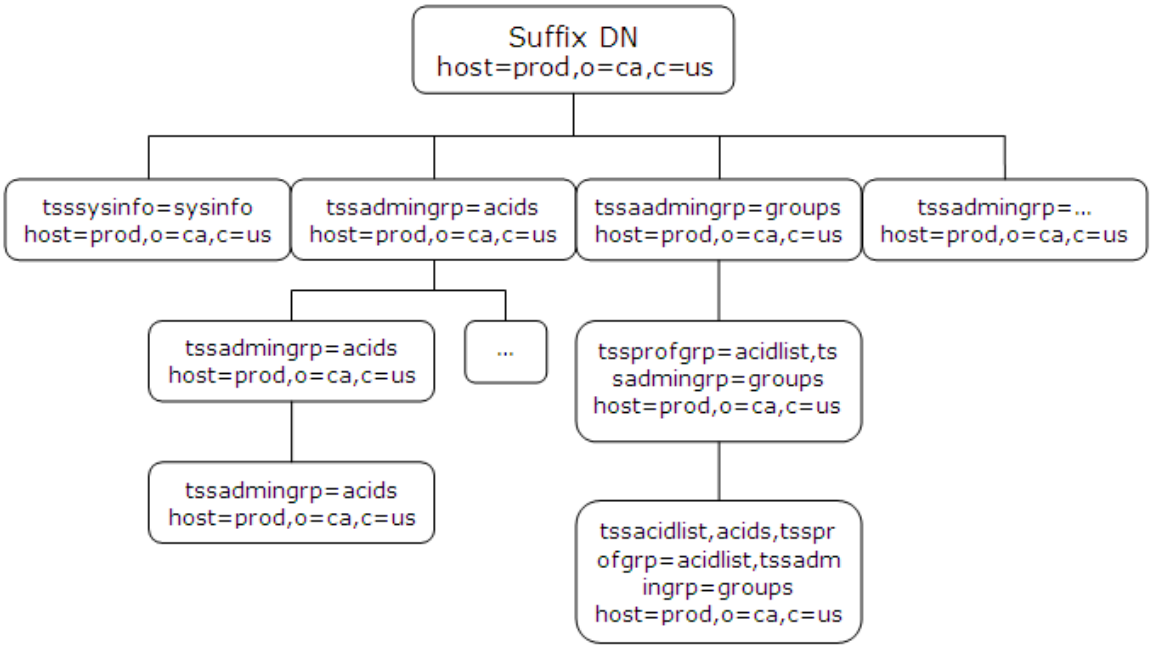
CA Top Secret r12 (TSS) Backend Security Option

When you are using TSS to secure the CA LDAP Server for z/OS, complete the following steps before configuring the connection from the Policy Server to the CA LDAP Server:

1. Become familiar with the TSS objectclass hierarchy.
2. Add the TSS objectclasses to the Policy Server registries in the LDAP namespace.

TSS Objectclass Hierarchy

The following diagram shows the hierarchy of objectclass entries in the CA Top Secret Directory Information Tree (DIT). Below the diagram is a description of each objectclass.



Objectclass host

Object class used to start access to the objectclass hierarchy for a CA Top Secret database.

Objectclass tsssysinfo

Object class used to create branches in the objectclass hierarchy below the host.

Objectclass tssadmingrp

Object class used to create branches in the objectclass hierarchy below the host.

Values:

- acids
- profiles
- groups
- departments
- divisions
- zones

Objectclass tssacid

Object class used to access the ACID record fields of all user types.

Objectclass tssacidgrp

Object class used to create the branches in the objectclass hierarchy below an acid.

Configure Policy Server Registry Entries for TSS

The CA LDAP Server for z/OS contains different object classes than other LDAP servers. Before configuring a connection from the Policy Server to the CA LDAP Server, add the TSS objectclasses to certain Policy Server registry entries in the LDAP namespace. Substitute the replacement values for the default values of the following Policy Server registry entries:

registry_entry_home

Specifies the following registry entry location:

HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Ds.

default_value

Specifies the default value of the registry entry.

replacement_value

Specifies a new value containing the TSS objectclasses for the registry entry.

- registry_entry_home\ClassFilters

  class_filters_default_value:

  organization,organizationalUnit,groupOfNames,groupOfUniqueNames,group

  class_filters_replacement_value:

  class_filters_default_value,eTTSSAcidName,tssacidgrp,tssadmingrp

- registry_entry_home\GroupClassFilters

  group_class_filters_default_value:

  groupOfNames,groupOfUniqueNames,group

  group_class_filters_replacement_value:

  group_class_filters_default_value,eTTSSAcidName,tssacidgrp,tssadmingrp

- registry_entry_home\PolicyClassFilters

  policy_class_filters_default_value:

  organizationalPerson,inetOrgPerson,organization,organizationalUnit, groupOfNames,groupOfUniqueNames,group

  policy_class_filters_replacement_value:

  policy_class_filters_default_value,eTTSSAcidName,tssacidgrp,tssadmingrp

- registry_entry_home\PolicyResolution

Add the following TSS object classes to this registry entry:

TSS Objectclass	Registry Key Type	Data
eTTSSAcidName	REG_DWORD	0x00000001(1)
tssacid	REG_DWORD	0x00000001(1)
tssacidgrp	REG_DWORD	0x00000002(2)
tssadmingrp	REG_DWORD	0x00000003(3)

Note: Some LDAP queries that the Policy Server issues (such as a full list of users) can take up to 60 seconds to complete. Under these conditions most of the queries from the Policy Server-side timeout. To improve connectivity, you can adjust this registry key entry as follows:

[HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Debug]
LDAPPingTimeout = 300; REG_DWORD

Configure CA LDAP Server for z/OS User Directory Connections

You configure the user directory connection to let the Policy Server communicate with the user store.

Note: Load balancing and failover are not supported for this LDAP server.

Follow these steps:

1. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Complete the required connection information in the General and Directory Setup areas.

   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. Configure the LDAP search and LDAP user DN lookup settings in the LDAP Settings area.

   Note: Be sure to enter 100 in Max Time. The Policy Server requires this amount of time to retrieve data from this LDAP Server.

6. Do the following in the Administrator Credentials area:

   a. Select the Require Credentials option.

   b. Type the full DN of an administrator in Username.

   c. Type the adminstrator password in Password.

   Note: TSS does not allow anonymous binds to the user store.

7. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.

8. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.

9. Click Submit.

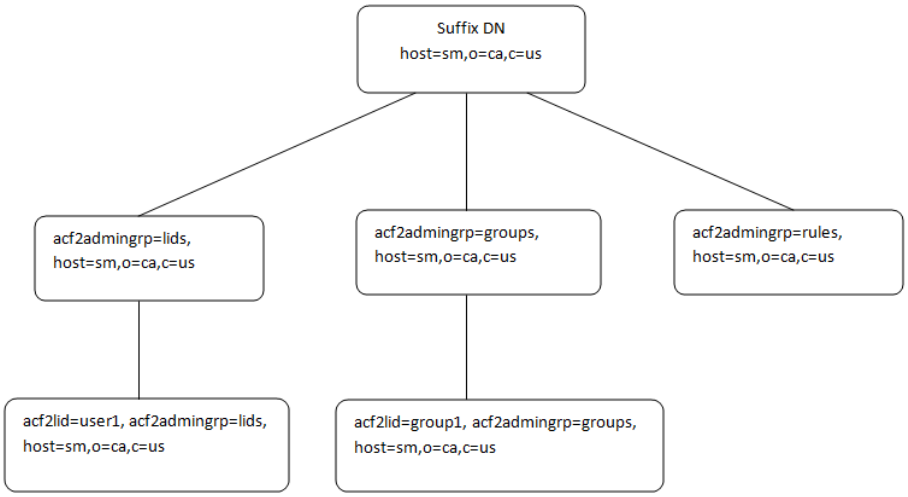
   The user directory connection is created.

CA LDAP Server r15 for z/OS (ACF2) Backend Security Option

This section describes the settings that are required to configure the CA LDAP Server r15 for z/OS (ACF2) as a user store with the Policy Server.

ACF2 Objectclass Hierarchy

The following illustration shows the hierarchy of objectclass entries in the CA ACF2 Directory Information Tree (DIT). The illustration provides a description of each objectclass.



Objectclass host

Object class that is used to start access to the objectclass hierarchy for a CA ACF2 database.

Objectclass acf2admingrp

Object class that is used to create branches in the objectclass hierarchy below the host.

Values:

- lids
- groups
- rules

Objectclass acf2lid

Object class that is used to access the LID record fields (user records). The acf2lid is the only objectclass that can be added, modified, and deleted. All other objectclass objects are read-only.

Objectclass acf2lidgrp

Object class that is used to emulate groups in CA ACF2.

Objectclass acf2ruletype

Object class that is used to group like rule types. The acf2ruletype objectclass is read-only and cannot be modified, added, or deleted.

Configure Policy Server Registry Entries for ACF2

The CA LDAP Server r15 for z/OS (ACF2) contains a different set of objectclasses than other LDAP servers. Before configuring a user directory connection from the Policy Server to the CA LDAP Server, add the ACF2 objectclasses to certain Policy Server registry entries in the LDAP namespace. Substitute the replacement values for the default values of the following Policy Server registry entries:

registry_entry_home

Specifies the following registry entry location:

HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Ds.

default_value

Specifies the default value of the registry entry.

replacement_value

Specifies a new value containing the ACF2 objectclasses for the registry entry.

- registry_entry_home\ClassFilters

  class_filters_default_value:

  organization,organizationalUnit,groupOfNames,groupOfUniqueNames,group

  class_filters_replacement_value:

  class_filters_default_value,*

- registry_entry_home\GroupClassFilters

  group_class_filters_default_value:

  groupOfNames,groupOfUniqueNames,group

  group_class_filters_replacement_value:

  group_class_filters_default_value,*

- registry_entry_home\PolicyClassFilters

  policy_class_filters_default_value:

  organizationalPerson,inetOrgPerson,organization,organizationalUnit,groupOfNames,groupOfUniqueNames,group

  policy_class_filters_replacement_value:

  policy_class_filters_default_value,*

- registry_entry_home\PolicyResolution

  Add the following ACF2 objectclasses to this registry entry:

ACF2 Objectclass	Registry Key Type	Data
acf2lid	REG_DWORD	0x00000001(1)
acf2admingrp	REG_DWORD	0x00000002(2)
eTACFLidName	REG_DWORD	0x00000001(1)

- registry_entry_home\Debug

  In UNIX, add the following ACF2 objectclass to this registry entry:

ACF2 Objectclass	Registry Key Type	Data
LDAPPingTimeout=	REG_DWORD	300;

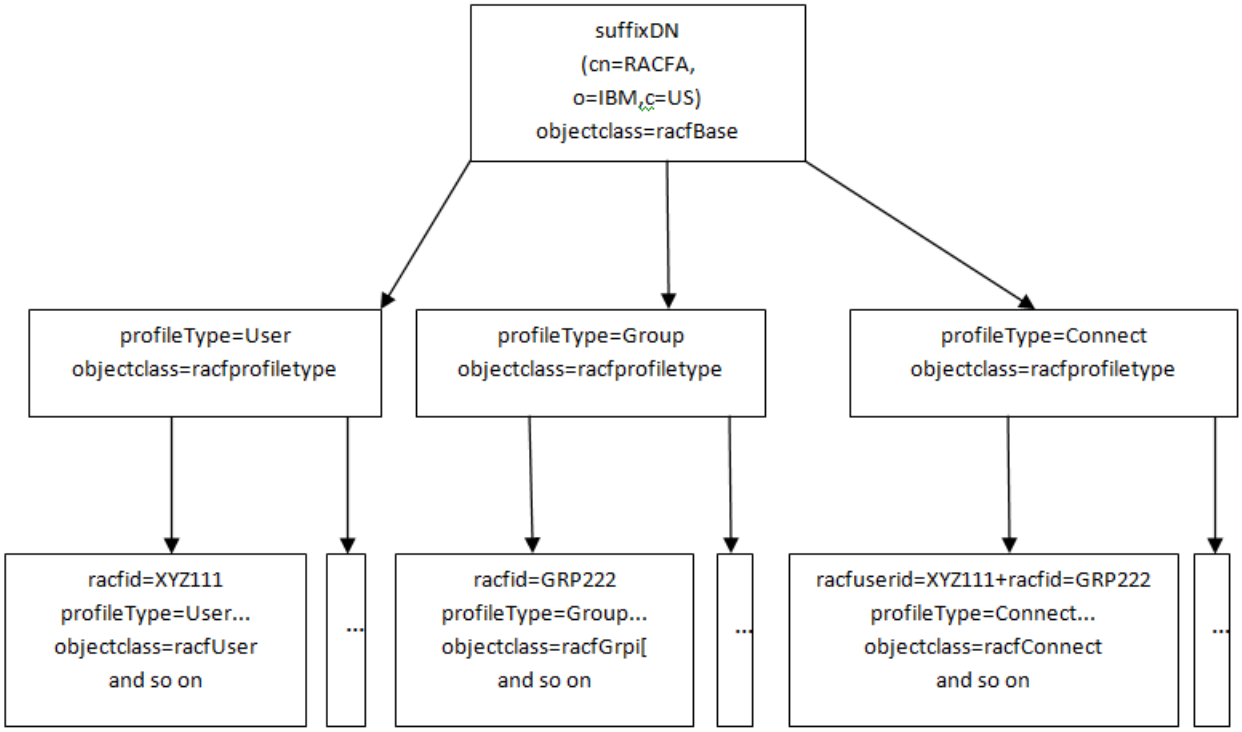
Note: The value of this registry key can be changed based on the response time of the CA LDAP Server r15 for z/OS (ACF2).

CA LDAP Server r15 for z/OS (RACF) Backend Security Option

This section describes the settings that are required to configure the CA LDAP Server r15 for z/OS (RACF) as a user store with the Policy Server.

RACF Namespace Hierarchy

The following illustration shows the hierarchy of namespace entries in the RACF Directory Information Tree (DIT). The illustration provides a description of each namespace. The RACF namespace hierarchy is similar to the ACF2 object class hierarchy.



Similar to the ACF2 Server, the top four entries in the hierarchy are reserved, read-only, and generated by the server. The purpose of these reserved entries is to enable a hierarchical representation of RACF users, groups, and connections.

Configure Policy Server Registry Entries for RACF

The CA LDAP Server r15 for z/OS (RACF) contains a different set of objectclasses than other LDAP servers. Before configuring a user directory connection from the Policy Server to the CA LDAP Server, add the RACF objectclasses to certain Policy Server registry entries in the LDAP namespace. Substitute the replacement values for the default values of the following Policy Server registry entries:

registry_entry_home

Specifies the following registry entry location:

HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Ds.

default_value

Specifies the default value of the registry entry.

replacement_value

Specifies a new value containing the RACF objectclasses for the registry entry.

- registry_entry_home\ClassFilters

  class_filters_default_value:

  organization,organizationalUnit,groupOfNames,groupOfUniqueNames,group

  class_filters_replacement_value:

  class_filters_default_value,*

- registry_entry_home\GroupClassFilters

  group_class_filters_default_value:

  groupOfNames,groupOfUniqueNames,group

  group_class_filters_replacement_value:

  group_class_filters_default_value,*

- registry_entry_home\PolicyClassFilters

  policy_class_filters_default_value:

  organizationalPerson,inetOrgPerson,organization,organizationalUnit,groupOfNames,groupOfUniqueNames,group

  policy_class_filters_replacement_value:

  policy_class_filters_default_value,*

- registry_entry_home\PolicyResolution

  Add the following RACF objectclasses to this registry entry:

RACF Objectclass	Registry Key Type	Data
eTRACUserid	REG_DWORD	0x00000001(1)
eTRACAdminGrp	REG_DWORD	0x00000002(2)

- registry_entry_home\Debug

  In UNIX, add the following RACF objectclass to this registry entry:

RACF Objectclass	Registry Key Type	Data
LDAPPingTimeout=	REG_DWORD	300;

Note: The value of this registry key can be changed based on the response time of the CA LDAP Server r15 for z/OS (RACF).

Configure a Connection from the Policy Server to CA LDAP Server for z/OS

To configure a directory connection from the Policy Server to the CA LDAP Server for z/OS (RACF) or CA LDAP Server for z/OS (ACF2), open an existing user directory object in the Administrative UI.

Follow these steps:

1. Open the User Directory Dialog.
2. In Directory Setup, select LDAP as the namespace.
3. Enter the connection information for your LDAP directory.

   Note: For more information, see the topic LDAP Namespace Directory Setup Tab in the *Policy Design Reference Guide*.

   Note: Failover is not supported for this LDAP Server.
4. In the LDAP Search section, in the Max Time field, specify a value of 300 seconds.

   Note: A greater timeout value is required, because the Policy Server takes more time to retrieve data from this LDAP Server.
5. In Credentials and Connection, specify administrator credentials that the Policy Server uses to connect to this LDAP Server.

   Important! Specifying administrator credentials is mandatory as anonymous binds to the user store are not allowed with CA LDAP Server r15 for z/OS (RACF) and CA LDAP Server r15 for z/OS (ACF2).

How to Configure an Oracle Directory Server Enterprise Edition User Directory Connection

You can use Oracle Directory Server Enterprise Edition to function as a SiteMinder user store. The following process lists the steps for creating the user store connection:

1. Ping the user store system.
2. Configure the Oracle Directory Server Enterprise Edition 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 Oracle Directory Server Enterprise Edition User Directory Connections

You configure a user directory connection to let the Policy Server communicate with the SiteMinder user store.

Follow these steps:

1. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Complete the required connection information in the General and Directory Setup areas.

   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. Configure the LDAP search and LDAP user DN lookup settings in the LDAP Settings area.

6. (Optional) Do the following in the Administrator Credentials area:

   a. Select Require Credentials.

   b. Enter the credentials of an administrator account.

      Note: Oracle recommends using an administrator account other than cn=Directory Manager. Using cn=Directory Manager can cause performance issues due to security policies applied to this account. Create a user with sufficient privileges to manage the directory and specify that user in Username.

7. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.

8. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.

9. Click Submit.

   The user directory connection is created.

More information:

LDAP Load Balancing and Failover (see page 228)
Define an Attribute Mapping (see page 261)

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.

Follow these steps:

1. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Complete the required connection information in the General and Directory Setup areas.

   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. Configure the LDAP search and LDAP user DN lookup settings in the LDAP Settings area.

6. (Optional) Do the following in the Administrator Credentials area:

   a. Select the Require Credentials option.

   b. Enter the credentials of an administrator account.

7. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.

8. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.

9. Click Submit.

   The user directory connection is created.

   Note: IBM Directory Server referrals are incompatible with SiteMinder.

More information:

LDAP Load Balancing and Failover (see page 228)
Define an Attribute Mapping (see page 261)

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. Be sure that the Domino user directory meets the following prerequisites before you configure it as a user store:

- 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, type *o=myorg.org* in the Root field on the User Directory screen.

- 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.

Follow these steps:

1. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Complete the required connection information in the General and Directory Setup areas.

   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. Configure the LDAP search and LDAP user DN lookup settings in the LDAP Settings area.

   Note: The value that you specify in Root 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 about load balancing and failover, see LDAP Load Balancing and Failover.

7. (Optional) Do the following in the Administrator Credentials area:

   a. Select the Require Credentials option.

   b. Enter the credentials of an administrator account.

8. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.

9. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.

10. Click Submit.

    The user directory connection is created.

More information:

User Disambiguation in an LDAP Directory (see page 159)
LDAP Load Balancing and Failover (see page 228)
Directory Attributes Overview (see page 170)
Define an Attribute Mapping (see page 261)

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 SiteMinder Administrator:

   - Special Access for the SiteMinder Administrator
   - Create a Novell eDirectory User Account for SiteMinder 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 SiteMinder, 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 170)

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 SiteMinder 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 SiteMinder.

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.

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 SiteMinder administrator (for example called SiteMinder_admin).
2. Give this user a password generated by the SiteMinder administrator which will be entered in the Administrative UI.

Create a Novell eDirectory User Account for SiteMinder Administration

You can give the SiteMinder Administration a user account using the NW Admin tool.

To create a Novell eDirectory user account for SiteMinder 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=SiteMinder_admin.

By default, set the access level to Read, which is sufficient for SiteMinder's basic functions. Customers whose use active APIs or some of SiteMinder'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.

Follow these steps:

1. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Complete the required connection information in the General and Directory Setup areas.

   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. Configure the LDAP search and LDAP user DN lookup settings in the LDAP Settings area.

   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) Do the following in the Administrator Credentials area:

   a. Select the Require Credentials option.

   b. Enter the credentials of an administrator account.

7. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.

8. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.

9. Click Submit.

   The user directory connection is created.

More information:

LDAP Load Balancing and Failover (see page 228)
Define an Attribute Mapping (see page 261)

How to Configure an Active Directory LDS User Directory Connection

You can use Active Directory LDS as a user store. Complete the following procedures:

1. Ping the user store system
2. Configure the 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 Active Directory LDS User Store Directory Connections

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

Follow these steps:

1. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Complete the required connection information in the General and Directory Setup areas.

   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. Configure the LDAP search and LDAP user DN lookup settings in the LDAP Settings area.

6. (Optional) Do the following in the Administrator Credentials area:

   a. Select the Require Credentials option.

   b. Enter the credentials of an administrator account.

7. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.

8. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.
9. Click Submit.

   The user directory connection is created.

More information:

LDAP Load Balancing and Failover (see page 228)
Define an Attribute Mapping (see page 261)

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 points:

Enhanced Active Directory integration

Active Directory 2008 has several user and domain attributes that are specific to the Windows network operating system (NOS) and that the LDAP standard does not require. If you configure the Policy Server to use Active Directory as a user store, enable Enhanced Active Directory Integration from the Policy Server Global Tools task available from the Administrative UI. This option improves the integration between the user management feature of the Policy Server and Password Services with Active Directory by synchronizing Active Directory user attributes with SiteMinder mapped user attributes.

Note: For more information, see the section Enable Enhanced Active Directory Integration in the *Policy Server Administration Guide*.

Multibyte Character Support

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

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

Active Directory namespace does not support paging

A search fails when the search results in more than 1000 users.

Authentication against a User Directory of an AD namespace

The Policy Server can authenticate a user against an Active Directory using SASL. To enable the use sasl bind, set the SASLBind registry key with a value of 1.

Note: When enabling this setting, set the administrator name on the user directory configuration to the AD login name, rather than the fully qualified distinguished name.

Create the registry key EnableSASLBind of type DWORD at the following location:

`HKLM\Software\Netegrity\SiteMinder\CurrentVersion\Ds\LDAPProvider`

EnableSASLBind

Disables or enables the SASL protocol while authenticating users. If you set EnableSASLBind to 1, the authentication occurs with SASL. If you set EnableSASLBind to 0, the authentication occurs with Simple Authentication mechanism.

Value: 0 (disabled) or 1 (enabled)

Note: The SASL authentication is specific to Windows-based Policy Servers.

Important! To configure an Active Directory User Directory using a secure (SSL) connection, set the Policy Server registry key EnableSASLBind to 0.

Administrator Credentials

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

LDAP Search Root Configuration

The Policy Server must identify the AD domain of an AD namespace to read account lock status. To configure the Policy Server to identify the AD domain, define 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. If the Policy Server cannot identify the AD domain, it cannot read the domain Windows lockout policy. 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 the following DN and lock two of these users:

```
ou=People,dc=clearcase,dc=com
```

The SiteMinder User Management dialog shows locked users as disabled only if the LDAP search root is configured as the DN of the AD domain, as follows:

```
dc=clearcase,dc=com
```

If you configure the LDAP search root as follows, the locked users are incorrectly shown as enabled:

```
ou=People,dc=clearcase,dc=com
```

Disable Password Services Redirect for Natively Disabled Unauthorized Users

By default, SiteMinder 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, SiteMinder 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/LDAP Provider

Values: 0 (disabled) or 1 (enabled).

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

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

SiteMinder 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 SiteMinder 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 SiteMinder Web Agent can run in a Windows user security context for accessing Web resources on IIS Web servers. Before SiteMinder can provide the Windows user security context, you must configure a session store and enable persistent sessions on a per realm basis (see How SiteMinder 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 164)

AD Namespace for an Active Directory Connection

SiteMinder 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.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

4. Microsoft Active Directory is an LDAP-compliant user directory. You can configure the connection using the AD namespace or the LDAP namespace. Do one of the following:

   a. Leave the default LDAP settings.

   b. Select AD from the Namespace list in the Directory Setup area.

5. Complete the remaining required connection information in the General and Directory Setup areas.

   Note: Consider the following:

   - For more information about an authenticated user security context, see How a Windows User Security Context is Obtained.
   - If you are using an SSL connection from the Policy Server to an Active Directory namespace, specify the FQDN and port number in the Server field in the Directory Setup area. 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.

6. (Optional) Click Configure in the Directory Setup area to configure load balancing and failover.

   Note: More information about load balancing and failover, see LDAP Load Balancing and Failover.

7. Do the following in the Administrator Credentials area:

   a. Select the Require Credentials option.

   b. Enter the credentials of an administrator account.

   Note: 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. Otherwise, user authentication can fail.

8. Configure the LDAP Search and LDAP User DN Lookup settings in the LDAP Settings area.

9. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.

10. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.

11. Click Submit.

    The user directory connection is created.

More information:

LDAP Load Balancing and Failover (see page 228)
Directory Attributes Overview (see page 170)
Define an Attribute Mapping (see page 261)

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, SiteMinder features that require writing to Active Directory, such as Password Services, are not supported, because Global Catalog does not support writes to Active Directory.

Follow these steps:

1. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Complete the required connection information in the General and Directory Setup areas.

   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. Configure the LDAP search and LDAP user DN lookup settings in the LDAP Settings area.

6. (Optional) Click Configure to configure load balancing and failover.

   Note: More information about load balancing and failover, see LDAP Load Balancing and Failover.

7. (Optional) Do the following in the Administrator Credentials area:
   a. Select the Require Credentials option.
   b. Enter the credentials of an administrator account.
8. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.
9. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.
10. Click Submit.

    The user directory connection is created.

More information:

Define an Attribute Mapping (see page 261)

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) Do the following in the Administrator Credentials area:
   a. Select the Require Credentials option.
   b. Enter the credentials of an administrator account.
7. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.
8. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.
9. Click Submit.

   The user directory connection is created.

More information:

LDAP Load Balancing and Failover (see page 228)
Define an Attribute Mapping (see page 261)

How to Configure OpenLDAP Server User Directory Connections

You can use OpenLDAP Server as a user store. Complete the following steps to create the user directory connection:

1. Create a user store.
2. Configure the OpenLDAP Server user directory connection.

Create a User Store

You can use an OpenLDAP directory server as a user store

To create a user store

1. Use an LDIF file to create ou=People under the root DN.
2. Create users under the organizational unit.

Configure OpenLDAP Directory Server User Directory Connections

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

Follow these steps:

1. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Complete the required connection information in the General and Directory Setup areas.

   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. Configure the LDAP search and LDAP user DN lookup settings in the LDAP Settings area.

6. Do the following in the Administrator Credentials area:

   a. Select the Require Credentials option.

   b. Enter the credentials of an administrator account.

7. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.

8. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.

9. Click Submit.

   The user directory connection is created.

More information:

LDAP Load Balancing and Failover (see page 228)
Define an Attribute Mapping (see page 261)

How to Configure a Red Hat Directory Server User Directory Connection

You can use Red Hat Directory Server as a user store. Complete the following steps to create the user directory connection:

1. Ping the user store system.
2. Configure the 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 Red Hat Directory Server User Directory Connnections

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

Follow these steps:

1. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Complete the required connection information in the General and Directory Setup areas.

   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. Configure the LDAP search and LDAP user DN lookup settings in the LDAP Settings area.

6. Do the following in the Administrator Credentials area:
   a. Select the Require Credentials option.
   b. Enter the credentials of an administrator account.
7. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.
8. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.
9. Click Submit.

   The user directory connection is created.

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 use a Microsoft SQL Server database for audit logs and caching is turned on, under heavy load, SiteMinder 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.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Select ODBC from the Namespace list.

   ODBC settings appear.

5. Complete the remaining required connection information in the General and Directory Setup areas.

   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. Select a SQL query scheme.

7. (Optional) Do the following in the Administrator Credentials area:

   a. Select Require Credentials.

   b. Enter the credentials of an administrator account.

   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 account must have read or read/write privileges for the user directory.

8. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder use in the User Attributes area.

9. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.

10. Click Submit.

    The user directory connection is created.

More information:

SQL Query Schemes (see page 233)
Configure ODBC Data Source Failover (see page 232)
Define an Attribute Mapping (see page 261)

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 SiteMinder 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.

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.

   Objects related to user directories appear on the left.

2. Click User Directories.

   The User Directories screen appears.

3. Click Create User Directory.

   The Create User Directory screen appears and displays the required settings to configure an LDAP connection.

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

4. Select Custom from the Namespace list.

   The settings to configure a Custom connection appear.

5. Complete the required connection information in the General and Directory Setup areas.

   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) Do the following in the Administrator Credentials area:

   a. Select the Require Credentials option.

   b. Enter the credentials of an administrator account.

7. (Optional) Click Create in the Attribute Mapping List area to configure user attribute mapping.

   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, create the user directory connection.

8. Click Submit.

   The user directory connection is created.

More information:

Directory Attributes Overview (see page 170)
Define an Attribute Mapping (see page 261)

How to Configure an LDAP User Directory Connection over SSL

Configuring an LDAP user directory connection over SSL requires that you configure SiteMinder to use your certificate database files.

Complete the following steps to configure the connection over SSL:

1. Perform steps required before you configure a connection over SSL.
2. Create the certificate database files
3. Add the root Certificate Authority (CA) to the certificate database
4. Add the server certificate to the certificate database
5. List the certifications in the certificate database
6. Configure the user directory connection for SSL
7. Point the Policy Server to the certificate database
8. Verify the SSL connection

Before You Configure a Connection over SSL

Review the following points before configuring an LDAP user directory connection over SSL:

- Verify that your directory server is SSL-enabled.

  Note: For more information about configuring your directory server to communicate over SSL, refer to the vendor-specific documentation.

- The Policy Server uses a Mozilla LDAP SDK to communicate with LDAP directories. As a result, the database files must be in the Netscape database version file format (cert8.db).

  Important! Do not use Microsoft Internet Explorer to install certificates into your cert8.db database file.

- (Active Directory) Considering the following points:
  - If the user directory connection is configured with the AD namespace, the SSL process that is documented in the subsequent topics 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:
    - Verify that the user directory connection is configured for a secure connection. For more information, refer to Configure the User Directory Connection for SSL.
    - On the computer hosting the Active Directory instance, verify that the root CA certificate and the server certificate are added to the service certificate store.

    Note: For more information about configuring Active Directory to communicate over SSL, refer to the Microsoft documentation.
  - If the user directory connection was configured with the LDAP namespace, complete the process to configure the connection over SSL.

Create the Certificate Database Files

Thecertificate database files must be in the Netscape database file format (cert8.db). Use the Mozilla Network Security Services (NSS) certutil application that is installed with the Policy Server 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.

Important! Before running a SiteMinder utility or executable on Windows Server 2008, open the command line window with administrator permissions. Open the command line window this way, even if your account has administrator privileges.

Follow these steps:

1. From a command prompt, navigate to the installation bin directory.

   Example: C:\Program Files\CA\SiteMinder\bin

   Note: Windows has a native certutil utility. Verify that you are working from the Policy Server bin directory, or you can inadvertently run the Windows certutil utility.

2. Enter the following command:

   `certutil -N -d certificate_database_directory`

   -N

   Creates the cert8.db, key3.db, and secmod.db certificate database files.

-d *certificate_database_directory*

Specifies the directory in which the certutil tool 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:

   - cert8.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

Add the root Certificate Authority (CA) to make it available for communication over SSL. Use the Mozilla Network Security Services (NSS) certutil application installed with the Policy Server to add the root CA.

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! Before running a SiteMinder utility or executable on Windows Server 2008, open the command line window with administrator permissions. Open the command line window this way, even if your account has administrator privileges.

Follow these steps:

1. From a command prompt, navigate to the Policy Server installation bin directory.

   Example: C:\Program Files\CA\SiteMinder\bin

   Note: Windows has a native certutil utility. Verify that you are working from the bin directory of the NSS utility, or you can 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

Add the server certificate to the certificate database to make it available for communication over SSL. Use the Mozilla Network Security Services (NSS) certutil application installed with the Policy Server to add the server certificate.

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! Before running a SiteMinder utility or executable on Windows Server 2008, open the command line window with administrator permissions. Open the command line window this way, even if your account has administrator privileges.

To add the server certificate to the certificate database

1. From a command prompt, navigate to the Policy Server installation bin directory.

   Example: C:\Program Files\CA\SiteMinder\bin

   Note: Windows has a native certutil utility. Verify that you are working from the bin directory of the NSS utility, or you can 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

List the certificates to verify that they were added to the certificate database. Use the Mozilla Network Security Services (NSS) certutil application that is installed with the Policy Server 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.

Important! Before running a SiteMinder utility or executable on Windows Server 2008, open the command line window with administrator permissions. Open the command line window this way, even if your account has administrator privileges.

Follow these steps:

1. From a command prompt, navigate to the Policy Server installation bin directory.

   Example: C:\Program Files\CA\SiteMinder\bin

   Note: Windows has a native certutil utility. Verify that you are working from the bin directory of the NSS utility, or you can 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.

   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 be sure that an SSL connection is used when the Policy Server and user store communicate.

To configure the user store connection for SSL

1. Login to the Administrative UI.
2. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

3. Click User Directories.

   The User Directories screen appears. The table lists the names of existing user directory connections.

4. Click the name of the user directory connection you want.

   The user directory settings appear as read–only.

5. Scroll down and click Modify.

   The settings become active.

6. Select the Secure Connection option in the Directory Setup area and click Submit.

   The user directory connection is configured to communicate over SSL.

Point the Policy Server to the Certificate Database

Point the Policy Server to the certificate database to configure SiteMinder to communicate with the user directory over SSL.

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. Use Administrator permissions even if you are logged in to the system as an Administrator. For more information, see the release notes for your SiteMinder component.

2. Click the Data tab.
3. Enter the path to the certificate database file in the Netscape Certificate Database File field.

   Example: C:\certdatabase\cert8.db

   Note: The key3.db file must be in the same directory as the cert8.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 be sure that the user directory and the Policy Server are communicating over SSL.

Follow these steps:

1. Login to the Administrative UI.
2. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

3. Click User Directories.

   The User Directories screen appears. The table lists the names of existing user directory connections.

4. Click the name of the user directory connection you want.

   The user directory settings appear as read–only.

5. Click View contents.

   If SSL is properly configured, the Directory Content screen appears and lists the contents of the user directory.

Configure an Oracle User Directory Connection Over SSL

SiteMinder lets you configure the connection between the Policy Server and Oracle database to communicate over SSL.

Note: A prerequisite for this communication is that the Oracle database must be enabled for SSL. For more information about enabling the database for SSL, see the Oracle documentation.

How the Policy Server connects to an Oracle Database over SSL

The following process describes how the connection is established between the Policy Server and the Oracle database over SSL:

1. When the Policy Server makes a connection request, the Oracle database server presents its public certificate. The Policy Server can be configured to validate the authenticity of the certificate that the Oracle database server presents.

   Optionally, you can configure the Policy Server to communicate with an Oracle database over SSL without configuring the Policy Server to validate the certificate.

   Note: The Policy Server uses a trust store to validate the certificate authenticity. The trust store can either be a single public certificate of the Certificate Authority (CA) or a PKCS12 trust store that contains a list of public certificates from trusted CAs. The public certificate is not password-protected, whereas, the PKCS12 trust store is encrypted and password-protected.

2. If the certificate that the Oracle database server presents matches a certificate in the trust store, an encrypted connection is established between the Policy Server and Oracle database. If the certificate that does not match a certificate in the trust store, the connection fails and the Policy Server generates an error.

Configure SSL on Windows

You can configure the Policy Server to communicate with Oracle over SSL using the ODBC Data Source Administrator Console.

Follow these steps:

1. Open the ODBC Data Source Administrator console.
2. In the System DSN tab, select a DSN for your CA SiteMinder Oracle database.
3. Click Configure.

   The ODBC Oracle Wire Protocol Driver Setup dialog appears.
4. Click the Security tab.
5. Specify the following encryption parameters:

   Encryption Method

   Specifies the encryption method the Policy Server uses to encrypt data that is sent between the Policy Server and the Oracle database server.

   Default: 0 – No Encryption

   Required Value: 1 – SSL Auto

   Validate Server Certificate

   (Optional) Specifies that the Policy Server validates the authenticity of the certificate that the Oracle database server presents.

   Default: Selected

   To configure SSL without requiring the Policy Server to validate the authenticity of the certificate that the Oracle database presents, clear the selection.

   Trust Store

   Defines the path name of the trust store file. Specify this value only if you require the Policy Server to validate the authenticity of the certificate that the Oracle database presents.

   Required Value: The trust store can either be the public certificate of the CA or a PKCS12 trust store that contains one or more certificates. The public certificate is a single certificate which is not password-protected. The PKCS12 trust store is password-protected.

   Trust Store Password

   Defines the password that is required to access the trust store.

   Host Name In Certificate

   Defines the hostname in the certificate. The hostname in the certificate must match the hostname that is used to connect to the Oracle database server. If the hostname does not match, the connection fails.

Note: The Key Store, Key Store Password, and Key Password parameters are not applicable for this connection.

6. Click OK.

Configure SSL on UNIX

Configure SSL for the Policy Server on UNIX to enable the Policy Server to communicate with Oracle over SSL.

Follow these steps:

1. Edit the system_odbc.ini file using an editor. The system_odbc.ini file is located in the /nete_ps_root/db directory.

   Note: For more information about the system_odbc.ini file, see the *Policy Server Installation Guide*.

2. Add the following parameters to the Oracle DSN that you want to connect over SSL:

   Note: For more information about the parameters, see Configure SSL on Windows (see page 227).

   ```
   ValidateServerCertificate=0 or 1
   ```

   Note: Specify 1, if you want to validate the Server Certificate. Specify 0, if you do not want to validate the Server Certificate.

   ```
   TrustStore=Path to the CA certificate or PKCS12 trust store
   TrustStorePassword=TrustStorePassword
   HostNameInCertificate=hostname.domain.com
   ```

   Example:

   ```
   ValidateServerCertificate=1
   TrustStore=\nete_ps_root\db\MyCAcert.cer or
              \nete_ps_root\db\MyCertTrustStore.p12
   TrustStorePassword=abcd
   HostNameInCertificate=mydbhost.abc.com
   ```

3. Save and close the system_odbc.ini file.

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 SiteMinder 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 SiteMinder SQL Query Scheme dialog.

Note: The "SM_" prefix in column names is reserved for additional special names required by SiteMinder. 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 that you are creating an object. You can also copy the properties of an existing object to create an object.

To configure a SQL Query Scheme

1. Click Infrastructure, Directory.

   Objects related to user directories appear on the left.

2. Click SQL Query Scheme.

   The SQL Query Scheme screen appears.

3. Click Create SQL Query Scheme.

4. Verify that the create new object option is selected and click OK.

   The Create SQL Query Scheme screen appears.

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

5. Type a name and description in the fields in the General area.

6. Update the contents of the query fields to correspond to your database schema.

   Configure each of the queries to work with your relational database. 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 SiteMinder 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

7. Select a query type and click Submit.

   The query is saved. You can associate the query scheme with a user directory connection

8. Restart the Policy Server using the Policy Server Management Console.

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

Stored procedures in Oracle functions must meet the following requirement:

- The first parameter must be the username, and the second parameter must be the password.

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.

SiteMinder supports asynchronous calls to the following databases:

- SQLServer
- Oracle 8 through 11g 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.

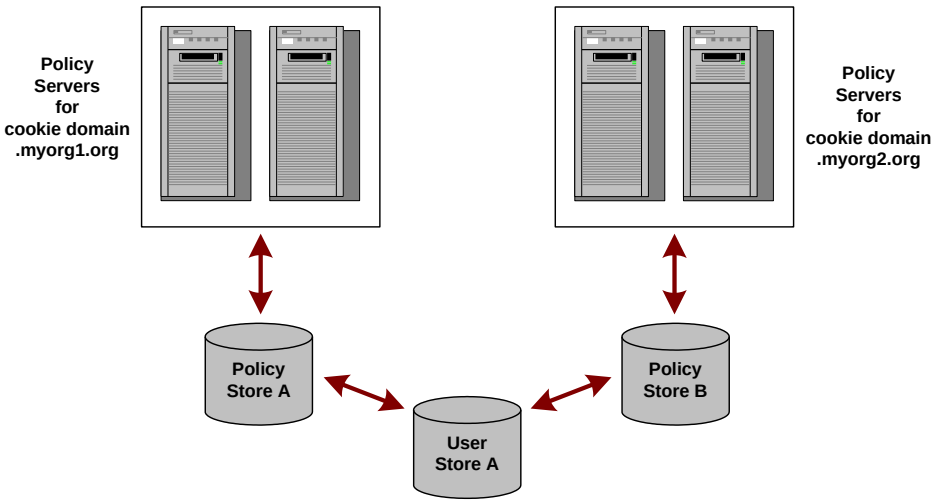
ODBC Connection Pooling

The following criteria apply to ODBC connection pooling:

- Connections are pooled by the ODBC data source. If one or more user directories use a data source, and the following criteria are met, the number of connections available can reach the total of the individual values specified for user directories and stores:
  - The policy store uses the data source.
  - Another store on the Policy Server uses the data source.
- 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 SiteMinder 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 SiteMinder 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 SiteMinder 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 SiteMinder 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, SiteMinder 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 172)

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 SiteMinder control. UIDs are often different from user login names.

UIDs allow SiteMinder 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 172)

How SiteMinder Uses UIDs

When you configure a user directory connection with a UID, once a user logs into SiteMinder, 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 SiteMinder 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. SiteMinder 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.

Named expressions are policy store objects that you reference by name and reuse in security policies defined in application objects. *Unnamed expressions* are stored in domain objects like responses and rules for use in traditional security policies.

Note: Named expressions can only be used in application objects. Named expressions *cannot* be used in traditional security policies defined using domain objects like responses and rules.

SiteMinder 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. SiteMinder 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 in security policies defined in application objects.

Note: Named expressions can only be used in application objects. Named expressions *cannot* be used in traditional security policies defined using domain objects like responses and rules.

SiteMinder evaluates named expressions to determine the values of calculated user attributes.

There are two types of named expressions:

- Virtual user attributes (see page 245)
- User classes (see page 248)

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
- SiteMinder built-in functions and expression syntax

Note: Named expressions can only be used in application objects. Named expressions *cannot* be used in traditional security policies defined using domain objects like responses and rules.

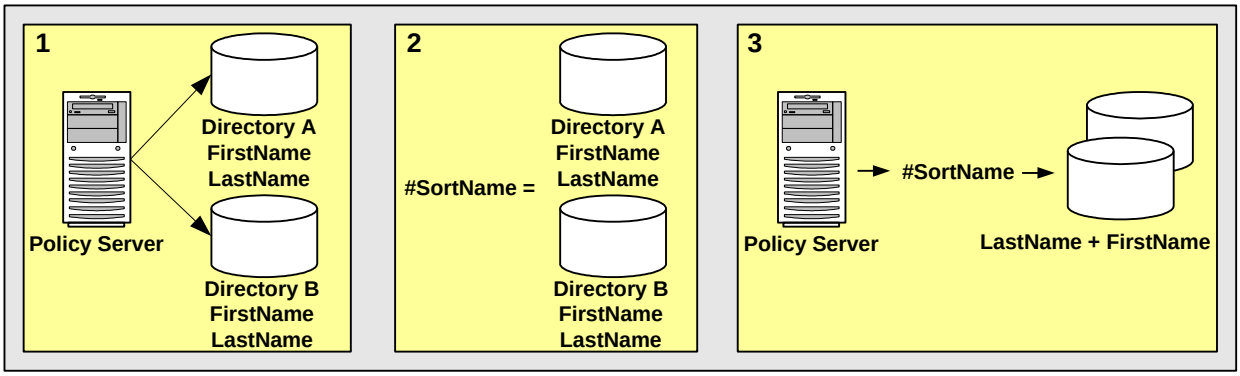
More information:

Expression Syntax Overview (see page 772)

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 that you are creating an object. You can also copy the properties of an existing object to create an object.

To define a virtual user attribute

1. Click Policies, Expression.
2. Click Named Expressions.

   The Named Expressions screen appears.
3. Click Create Named Expression.
4. Verify that the create new object option is selected and click OK.

   The Create Named Expression screen appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Select the Virtual User Attribute option and enter a name for the expression in the General area.
6. Type the expression in the Expression field in the Add Named Expression area.
7. (Optional) Select the Disabled option in the Add Named Expression area to disable the expression. A disabled expression is not listed in the expression editor and another named or unnamed expression cannot call it.
8. (Optional) Select the Private option in the Add Named Expression area. Only other named expressions can call a private expression. An unnamed expression cannot call a private expression.

9. (Optional) Click Edit in the Add Named Expression area to open the Expression Editor.
10. Click Submit.

    The named expression is created.

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
- SiteMinder built-in functions and expression syntax

Note: Named expressions can only be used in application objects. Named expressions *cannot* be used in traditional security policies defined using domain objects like responses and rules.

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.

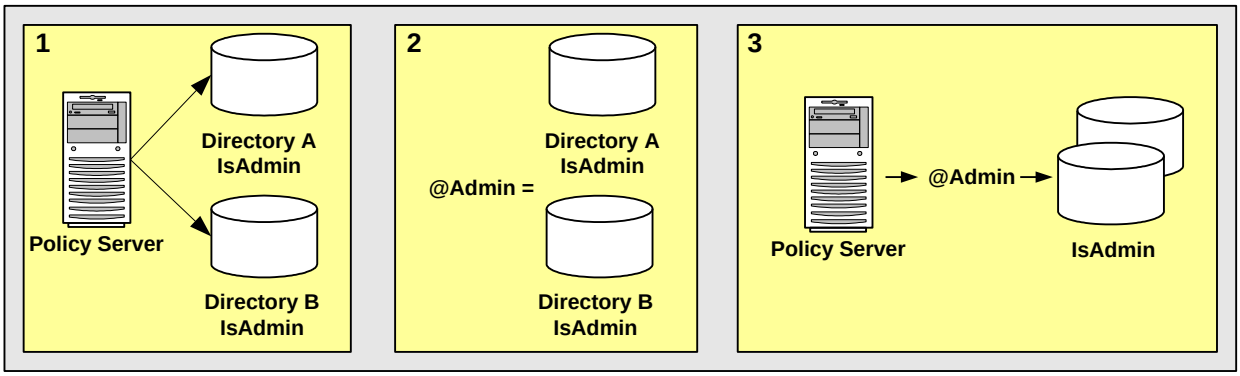
More information:

Expression Syntax Overview (see page 772)

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 SiteMinder 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 that you are creating an object. You can also copy the properties of an existing object to create an object.

To create a user class

1. Click Policies, Expression.
2. Click Named Expressions.

   The Named Expressions screen appears.
3. Click Create Named Expression.
4. Verify that the create new object option is selected and click OK.

   The Create Named Expression screen appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Select the User Class option and enter a name for the expression in the General area.
6. Type the expression in the Expression field in the Add Named Expression area.

   Note: The expression must be a Boolean expression.
7. (Optional) Select the Disabled option in the Add Named Expression area to disable the expression. A disabled expression is not listed in the expression editor and any other named or unnamed expression cannot call it.
8. (Optional) Select the Private option in the Add Named Expression area. Only other named expressions can call a private expression. An unnamed expression cannot call a private expression.
9. (Optional) Click Edit in the Add Named Expression area to open the Expression Editor.
10. Click Submit.

    The named expression is created.

How to Use the Expression Editor

You can use the expression editor to:

- Look up SiteMinder 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 SiteMinder 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 SiteMinder function and add the condition to an expression in the expression editor.

To create a condition containing a built-in SiteMinder 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 SiteMinder operation and add the condition to an expression in the expression editor.

To create a condition containing a built-in SiteMinder 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 253)
   - Delete a Condition from an Expression (see page 253)
   - Group the Conditions in an Expression (see page 254)
   - Change a Boolean Operator in an Expression (see page 255)
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 SiteMinder 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 SiteMinder 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 769)

User Attribute Mapping

When you configure a connection from the Policy Server to a user directory, you can map SiteMinder'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 SiteMinder 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

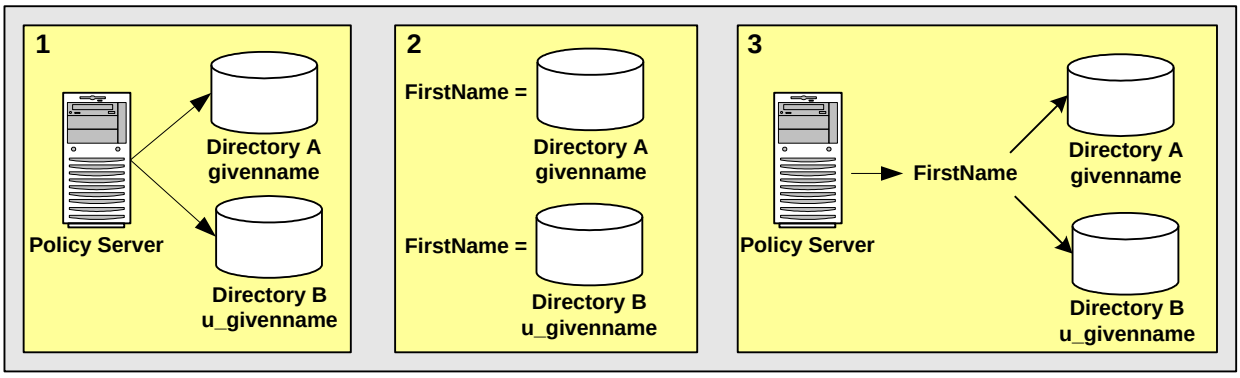
User directories store user information such as organizational information, user and group attributes, and individual credentials. Multiple user directories in a SiteMinder environment often store the same user information, but use different underlying schema and user attribute names to identify them. This results in a disparate view of the same user information from a SiteMinder perspective.

The purpose of user attribute mapping is to create a common view of the same user information by defining a universal schema. SiteMinder 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 SiteMinder 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. SiteMinder 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 262)
- Group Name (see page 264)
- Mask (see page 267)
- Constant (see page 271)
- Expression (see page 273)

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:

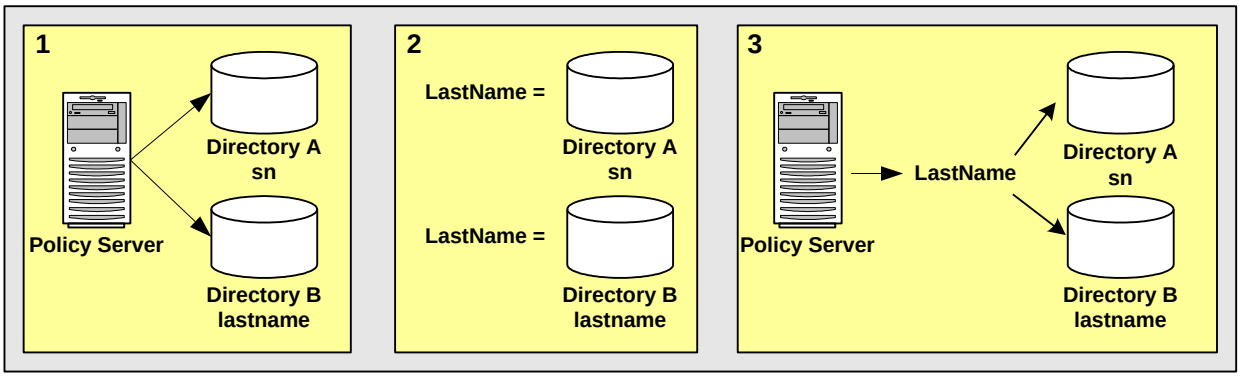
- string
- number
- Boolean

Alias Attribute Use Case

This use case shows two LDAP user directories, which identify the last name of users, but the directories have different underlying schema.

Note: Review the advanced user attribute mapping examples, which detail how to 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 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 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.

LastName results in a common view of the same user information. Use LastName when defining policies, expressions, or other objects that require the last names. The system has no concern for the directory-specific schema because the directories are operationally identical.

More information:

Named Expressions (see page 244)
Advanced User Attribute Mapping Examples (see page 276)

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.

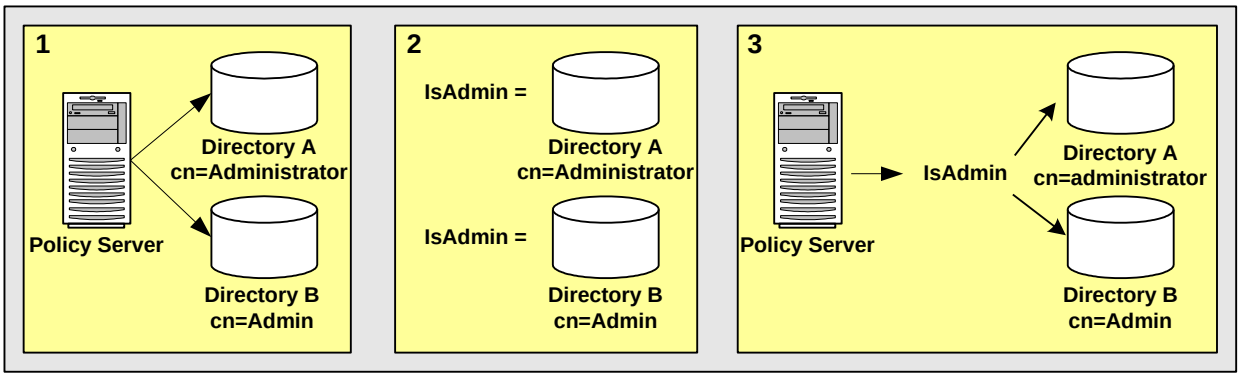
Note: For more information about user classes, see the Named Expressions.

Group Name Use Case

This use case shows two LDAP user directories, which use different underlying schema to identify users that belong to an Administrator group.

Note: Review the advanced user attribute mapping examples, which detail how to 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 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 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=groups,o=acme.com in Directory B.

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. The system has no concern for the directory-specific schema because the directories are operationally identical.

More information:

Named Expressions (see page 244)
Advanced User Attribute Mapping Examples (see page 276)

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.

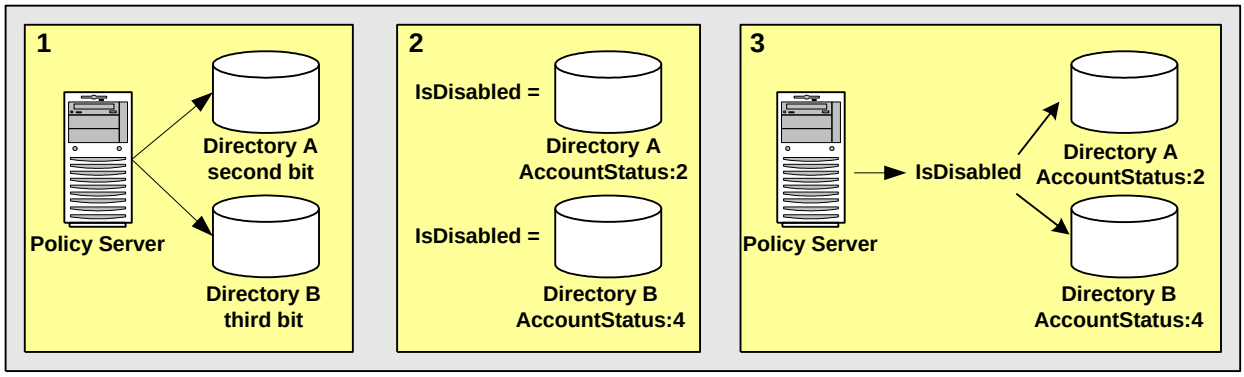
Mask Use Case

Some directory implementations use individual bits in an attribute to provide information about that attribute, such as the state of an account. You can apply a bit mask to an attribute.

This use case shows two Active Directory user stores that identify disabled user accounts. Each account has a different underlying schema.

Note: Review the advanced user attribute mapping examples, which detail how to 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 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 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, the bit mask 2 (decimal) determines whether the second bit of AccountStatus is set and the account is disabled.
   - In Directory B, bit mask 4 (decimal) determines whether the third bit of AccountStatus is set and the account is disabled.

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. The system has no concern for the directory-specific schema because the directories are operationally identical.

More information:

Named Expressions (see page 244)
Advanced User Attribute Mapping Examples (see page 276)

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.

   SiteMinder 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

A bit mask attribute mapping tests 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. The attribute AccountStatus stores the states of the following three flags in a bit pattern:

Bit Pattern	Flag
00?	account disabled?
0?0	password expired?
?00	gold member?

When a bit equals one, the flag is TRUE. The table shows the results:

Bit Pattern	Account Status
000 (0)	no flags are TRUE

Bit Pattern	Account Status
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
```

A bitwise AND operation on AccountStatus is performed on the bit mask and tests whether the result is equal to the bit mask. An equal result means the value of the tested bit is one and the flag is TRUE. The following table shows the results:

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
```

A bitwise AND operation on AccountStatus is performed on the bit mask and tests whether the result is equal to the bit mask. An equal result means the value of both tested bits is one and both flags are TRUE. The following table shows the results:

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:

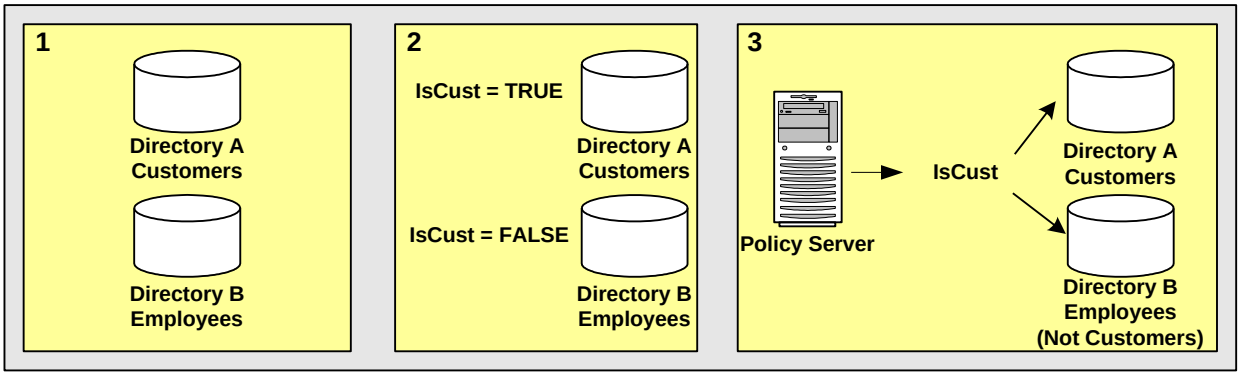
- string
- number
- Boolean

Constant Use Case

This use case represents a scenario in which one user directory stores only customers, while another user directory stores only employees.

Note: Review the advanced user attribute mapping examples, which detail how to 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. Reference IsCust when defining policies, expressions, or other objects. The common name lets the system determine whether a user is a customer, without regard to the particular directory in which the user is stored. The mapping indicates 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 244)
Advanced User Attribute Mapping Examples (see page 276)

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 SiteMinder 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 SiteMinder 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.

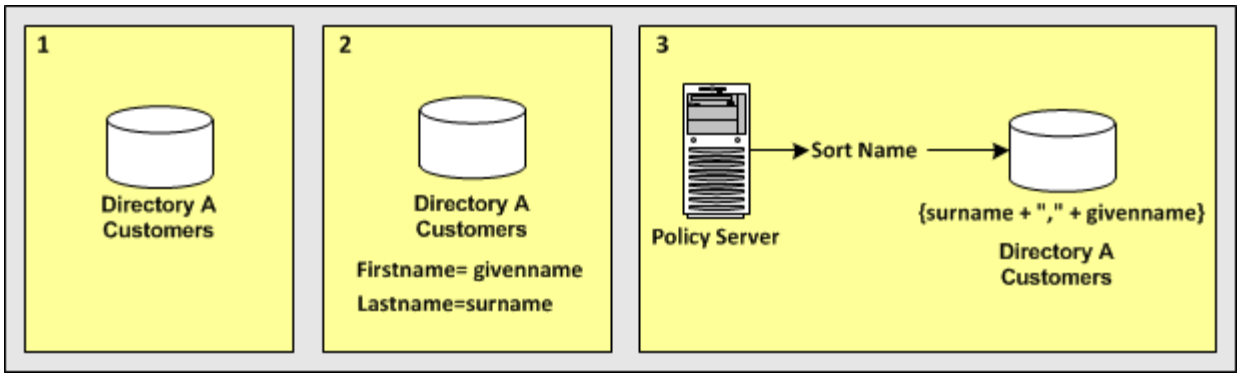
More information:

Named Expressions (see page 244)

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. A protected resource needs the *sort name* of each user (last name,first name). The user directory does not uniquely reference this attribute. Instead, the directory does store the last name of each user as surname and the first name of each user as givenname.

The following illustration details how an expression attribute mapping can create a common view of the same user information.



In the single user directory, a common name is mapped to an expression that creates the sort name using the user attribute names in the directory.

- Directory A contains all user records.
- The name of the mapping is **SortName**.
- The expression that defines SortName is:

  ```
  {surname + "," + givenname}
  ```

  Note: The expression conforms to the syntax rules of a SiteMinder expression.
- SortName is the common name that is mapped to the expression that includes the surname and the givenname attributes.

Reference SortName when defining policies, expressions, or other objects that require the sort name of users without concern for the directory-specific schema.

More information:

Named Expressions (see page 244)
Expression Syntax Overview (see page 772)
Advanced User Attribute Mapping Examples (see page 276)

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 SiteMinder 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.

Advanced User Attribute Mapping Examples

The following examples show more complex user attribute mapping configurations.

The example deployment is a retail clothing company that uses two user directories of different types:

Directory A

An internal LDAP user directory for employees only.

Directory B

An ODBC user directory for customers only.

Each user attribute mapping is specific to the user directory for which it is defined.

The following table details how Directory A and Directory B identify the same user information. The accompanying use cases explain how to use different attribute mappings to define a common view of the same user information. The common view serves as a universal schema, which makes the directories operationally identical.

Attribute Description	Directory A Attributes (LDAP)	Directory B Attributes (ODBC)
First name of each user	givenname	u_first_name
Last name of each user	surname	u_last_name

Sort name of each user (last name, first name)	The user directory does not uniquely store the user attribute.	sort_name
User as a customer	group:cn=customer,ou=groups,o=acme.com	Users are always customers.
Status of a user account	AccountStatus attribute (a set of flags). Second bit is a disabled account.	u_disabled

Map a First Name Attribute with an Alias Mapping Type

Use two alias attribute mappings to represent the first name user attribute in Directory A and Directory B.

Deployment

User Directory A identifies the first name of users with givenname. Directory B identifies the first name of users with u_first_name.

Solution

1. Create an alias attribute mapping for Directory A.

   Name

   FirstName

   Mapping Type

   Alias

   Definition

   givenname

2. Create an alias attribute mapping for Directory B.

   Name

   FirstName

   Mapping Type

   Alias

   Definition

   u_first_name

When referencing users in Directory A, the FirstName is mapped to givenname. When referencing users in Directory B, the FirstName maps to u_first_name.

Map a Last Name Attribute with an Alias Mapping Type

Use two alias attribute mappings to represent the last name user attribute in Directory A and Directory B.

Deployment

User Directory A identifies the last name of users with surname. Directory B identifies the last name of users with u_last_name.

Solution

1. Create an alias attribute mapping for Directory A.

   Name

   LastName

   Mapping Type

   Alias

   Definition

   surname

2. Create an alias attribute mapping for Directory B.

   Name

   LastName

   Mapping Type

   Alias

   Definition

   u_last_name

When referencing users in Directory A, the common view determines that the last name of users is identified by surname. When referencing users in Directory B, the common view determines that the last name is identified by u_last_name.

Map a Sort Name Attribute with Expression and Alias Mapping Types

Use an expression attribute mapping and an alias attribute mapping to represent the sort name of a user in Directory A and Directory B.

Deployment

- Directory A does not uniquely identify a the sort name for each user. For each user, Directory A stores the first name as givenname and a last name as surname for each user.
- Directory B identifies a sort name using sort_name.

Solution

1. Create an expression attribute mapping for Directory A:

   Name

   SortName

   Mapping Type

   Expression

   Definition

   ```
   (surname + "," + givenname)
   ```

   Note: The expression must conform to the syntax rules of an expression.

2. Create an alias attribute mapping for Directory B:

   Name

   SortName

   Mapping Type

   Alias

   Definition

   sort_name

When referencing users in Directory A, the sort name is calculated based on the specified expression. When referencing users in Directory B, the sort name is represented by the attribute sort_name.

Map Customers with Group and Constant Mapping Types

Use a group and a constant attribute mapping to identify customers in Directory A and Directory B.

Deployment

- Directory A stores employee. An employee of the company can also be a customer, so Directory A identifies customers as those employees that belong to the following group:

  `cn=Customers,ou=Groups,o=acme.com`

- 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 group attribute mapping for Directory A.

   Name

   IsCustomer

   Mapping Type

   Group

   Definition

   `cn=Customers,ou=Groups,o=acme.com`

2. Create a constant attribute mapping for Directory B.

   Name

   IsCustomer

   Mapping Type

   Constant

   Definition

   TRUE

When referencing Directory A, a user is considered a customer if they belong to cn=Customers,ou=Groups,o=acme.com. When referencing Directory B, every user is a customer.

Map the Account Status with the Mask and Expression Mapping Types

Use a mask attribute mapping and an expression attribute mapping to identify user accounts that are disabled in Directory A and Directory B.

Deployment

- Directory A identifies disabled accounts with a user attribute named AccountStatus, which is a set of flags. The second bit indicates a disabled account.
- 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

   IsDisabled

   Mapping Type

   Mask

   Definition

   AccountStatus:2

   The definition indicates that the bit pattern is stored in AccountStatus, and the bit mask is 2 (decimal).

2. Create a expression attribute mapping for Directory B.

   Name

   IsDisabled

   Mapping Type

   Expression

   Definition

   ```
   (u_disabled = "y")
   ```

   u_disabled is a Boolean expression.

When referencing Directory A, the bit pattern determines if a user is disabled. When referencing Directory B, the expression determines if a user is disabled.

Chapter 9: Directory Mapping

This section contains the following topics:

Directory Mapping Overview

SiteMinder assumes that a user will be authenticated and authorized against the same user directory. Although this default behavior is sufficient in many cases, SiteMinder 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.

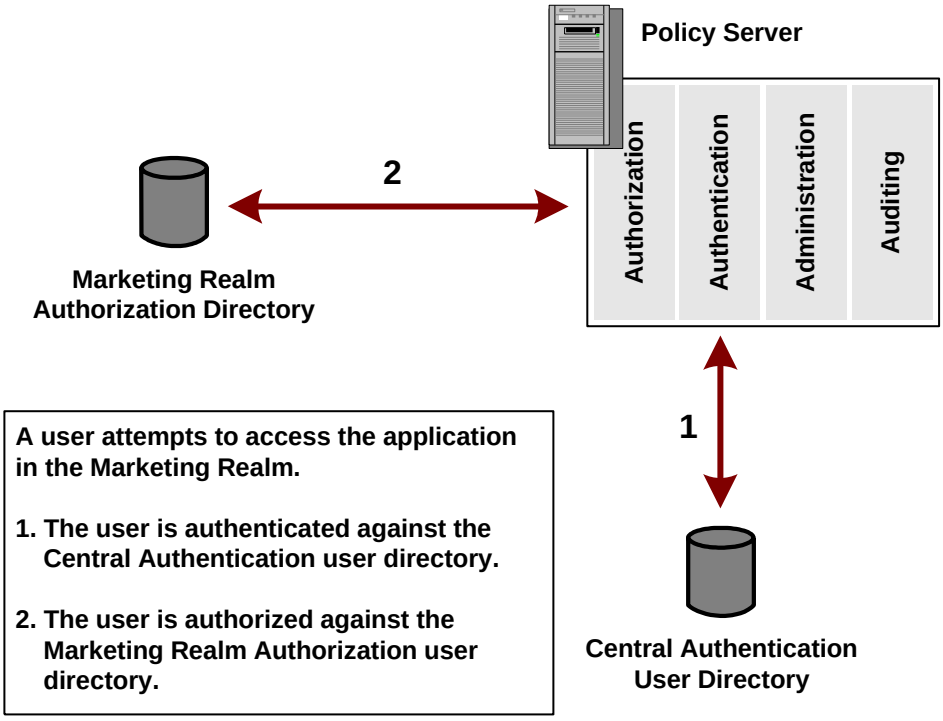
2. Configure a Directory Mapping

   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 172)
Realms (see page 501)
Directory Mapping Methods (see page 285)

Authorization Identity Mappings

Authorization Identity Mapping is the process of configuring a directory mapping to authenticate users against one directory and authorize users against a separate directory.

Assigning the configured authorization identity mapping to a realm lets the Policy Server authenticate a user in one directory and authorize a user in a separate directory.

Validation Identity Mappings

Validation Identity Mapping is the process of configuring a directory mapping to authenticate users against one directory and validate users against a separate directory.

With Validation Identity Mapping, you can map an authentication user directory that is connected to a Policy Server to a validation user directory that is connected to a different Policy Server.

Directory Mapping Methods

SiteMinder provides the following methods for mapping authentication directories with authorization and validation directories:

Identity Mappings

Introduced in SiteMinder r12.5, Identity Mappings provide greater flexibility than the legacy directory mapping methods. Identity Mappings let you configure multiple target user directories and use custom search criteria.

Legacy (Auth/Az and AuthValidate) Directory Mappings

The legacy Auth/Az and AuthValidate directory mapping methods available in previous releases are still available in this release. Existing mappings of these types continue to work in the same way.

More information:

Identity Mappings (see page 286)
Legacy Directory Mapping Methods (see page 292)

Directory Mapping Requirements

Legacy Directory mapping requires that the user directory connections to the Policy Server must already exist for the authentication directory and the authorization or validation directory.

Identity mapping does not require that the user directory connections to the Policy Server must already exist for authentication directory and validation directory.

More information:

User Directory Connections Overview (see page 157)

Identity Mappings

SiteMinder Identity Mappings provide an enhanced method of mapping users from a Source Directory to a Target Directory using custom search criteria. You can use Identity Mappings for both user authorization and user validation.

Identity Mapping provides the following two methods of mappings:

- Authorization Mapping (the default)
- Validation Mapping.

Identity Mappings enable custom search and let you control the order of mapping rules using different identity mapping entry objects. SiteMinder attempts to locate a user using the mapping mechanism defined in an Identity Mapping first. Only if the mapping fails, SiteMinder defaults to the session user directory. SiteMinder defaults to the session directory only if the session directory is present in the policy store.

Note: For validation mapping, the authentication directory need not be available in the local store.

Supported Directories for Identity 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
LDAP	Identical DN Universal ID Custom Search	Universal ID Custom Search
AD	Identical DN Universal ID Custom Search	Universal ID Custom Search
Relational Database	Universal ID Custom Search	Identical DN Universal ID Custom Search

Identity Mapping Entry Types

An Identity Mapping can contain one or more identity mapping entries. An identity mapping entry defines a rule that specifies how to find a user in the target directory. The Policy Server uses search criterion based on the session ticket information to find the user in the target directory.

An identity mapping can contain more than one target user directory. You can add identity mapping entries for different target user directories in the same identity mapping object. The identity mapping entries are processed as an ordered list of mappings.

The following are the two types of identity mapping entries:

Authorization Identity Mapping Entry

Specifies the links to the source and target directories. If the links are not available, the Authentication Directory is used for authorization. You can select either an Identical DN, a Universal ID, or specify custom search criteria.

Validation Identity Mapping Entry

Specifies the source directory name as a text string and a link to the target directory. You can provide the name of the source directory or select the default value, SMSESSION User Directory. The default value denotes that the user directory within the session ticket is used for validation. If there is no link to the target directory, the Authentication Directory is used for validation. You can select an Identical DN, a Universal ID, or specify a custom search attribute.

Using Complex User Search Expressions

You can map an authentication directory to an authorization directory or a validation directory using complex user search criteria. A user search criterion is a combination of attributes. The attribute can be from a source or target directory.

Typically, the user search criterion is user directory-specific. For example, an ODBC-based user search criterion can be different from an LDAP-based user search criterion.

To support user directories in different namespaces, define the search criteria for each user directory. You can also define a User Directory Attribute Mapping for the target user directory. Each user directory is then required to define its own specific search criterion for the attribute mapping. User Directory Attribute Mapping lets you define user directory-specific search criteria.

How to Configure an Authentication and Authorization Identity Mapping

Configuring an authentication-authorization identity mapping is a two-step process:

1. Configure an authorization identity mapping
2. Assign an authorization identity mapping to a realm

Configure an Authorization Identity Mapping

You can configure an identity mapping to authenticate users against one directory and authorize users against another directory. In addition to Identical DN and Universal ID, you can specify a custom search expression for the authorization identity mapping.

To configure an authorization identity mapping

1. Click Infrastructure, Directory.
2. Click Identity Mappings.

   The Identity Mappings page appears.
3. Click Create Identity Mapping.

   The Create Directory Mapping page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Specify a name and description for the mapping.
5. Select the mapping type as Authentication-Authorization.
6. Click Create Entry.

   The Create Identity Mapping Entry page appears.

7. Specify a name for the identity mapping entry.
8. Select the source and target directories from the respective lists.
9. Select a user search criterion from the following:

   Identical DN

   Maps the distinguished name (DN) of a user exactly from the authentication directory to the validation directory.

   Universal ID

   Matches the value of the Universal ID attribute from the authentication directory with the value of the Universal ID field from the validation directory to identify the user.

   Custom Search

   Specifies the attributes from the target directory and source directory. The source directory attribute can be a user-specified attribute or a SiteMinder session attribute.

10. Click OK.

    The identity mapping entry is added to the authorization identity mapping object.

11. Click Submit.

    The authorization identity mapping object is configured.

Assign an Authorization Identity Mapping to a Realm

You can assign an authorization identity mapping to a realm so that the Policy Server authenticates a user in one directory and authorizes a user in another directory. The Policy Server uses the authorization directory specified in the realm to authorize users.

To assign an authorization identity mapping to an existing Realm

1. Click Policies, Domain, Realms.

   The Realms page appears.

2. Select the realm that you want to modify.

   The View Realm page appears.

3. Click Modify.

   The settings and controls become active.

4. Select the identity mapping to use as the authorization directory from the Authorization Mapping list.
5. Click Submit.

   The Authorization identity mapping is assigned to the selected realm.

More information:

Configure Advanced Policy Components for Applications (see page 490)

How to Configure an Authentication and Validation Identity Mapping

Configuring an authentication and validation identity mapping is a two-step process:

1. Configure a validation identity mapping
2. Assign a validation identity mapping to a realm

Note: You can create validation mappings for directories within the same store. The Source Directory need not necessarily be in the local store.

Configure a Validation Identity Mapping

You can configure an identity mapping to authenticate users against one directory and validate users against another directory. In addition to Identical DN and Universal ID, you can specify a custom search expression for the authorization identity mapping.

To configure a validation identity mapping

1. Click Infrastructure, Directory.
2. Click Identity Mappings.

   The Identity Mappings page appears.
3. Click Create Identity Mapping.

   The Create Directory Mapping page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Type the name and description.
5. Select the mapping type as Authentication-Validation.
6. Click Create Entry.

   The Create Identity Mapping Entry page appears.
7. Type the name.
8. Specify a Source Directory if the directory is not from within the session.
9. Select the target directory.

10. Select a user search criterion.

    If you select Custom search, specify the attributes from the Target Directory and Source Directory.

    The Source Directory attribute can either be a user-specified attribute or a SiteMinder session attribute.

11. Click OK.

    The identity mapping entry is added to the validation identity mapping object.

12. Click Submit.

    The validation identity mapping object is configured.

Assign a Validation Identity Mapping to a Realm

You can assign a validation identity mapping to a realm so that the Policy Server authenticates a user in one directory and validates a user in another directory. The Policy Server uses the validation directory specified in the realm to authorize users.

To assign a validation identity mapping to an existing Realm

1. Click Policies, Domain, Realms.

   The Realms page appears.

2. Select the realm that you want to modify.

   The View Realm page appears.

3. Click Modify.

   The settings and controls become active.

4. Select the identity mapping to use as the authorization directory from the Validation Mapping list.

5. Click Submit.

   The validation identity mapping is assigned to the selected realm.

More information:

Configure Advanced Policy Components for Applications (see page 490)

Configure a Default Global Validation Directory Mapping

You can configure a single validation identity mapping to serve as the global default for validation mapping. Setting a global validation identity mapping saves you time by not having to set one for every realm. However, you can override the global validation identity mapping with a local mapping.

Follow these steps:

1. Click Policies, Global.
2. Click Select Global Validation Directory Mapping.

   The Select Global Validation Directory Mapping page appears.
3. Select a validation identity mapping object from the corresponding list.
4. Click Submit.

   The selected validation identity mapping object is set as the global default for validation mapping.

Legacy Directory Mapping Methods

Legacy directory mapping methods map the authentication directory to an authorization or a validation directory using an Identical DN or a Universal ID.

The following are the two types of legacy directory mapping methods:

- Auth/Az (Authorization mapping)
- AuthValidate (Validation mapping).

If an Auth/Az or AuthValidate mapping is configured, SiteMinder first attempts to use the session user directory to locate a user; uses the specified mapping mechanism only if the user is not found in the session user directory.

How to Configure an Authentication and Authorization Directory Mapping

Configuring an Auth/Az 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.

   The Auth/Az Mapping page appears.
3. Click Create Directory Mapping.

   The Create Directory Mapping page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Select the authentication and authorization directories from the respective lists.
5. Select the Identical DN or Universal ID.

   Important! The directory mapping is successful only if the Universal ID points to a single entry in the authorization directory.
6. Click Submit.

   The Create Directory Mapping task is submitted for processing.

More information:

Universal IDs (see page 243)

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.

   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.

   Important! You can map only one authorization directory per realm.
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.

More information:

Configure a Realm (see page 507)

How to Configure an AuthValidate Directory Mapping

AuthValidate Directory Mapping is an extension of Authentication and Authorization Directory Mapping. Both types of directory mapping allow users to authenticate against one user directory and authorize against another user directory. In both cases, the directory mapping type can be further specified as Identical DN or Universal ID.

AuthValidate directory mapping extends Authentication and Authorization directory mapping in three ways:

- With AuthValidate directory mapping, you can map an authentication user directory that is connected to one Policy Server to a validation user directory that is connected to another Policy Server. The user directories are located by OID and directory name.
- With AuthValidate directory mapping, the user directories can be of different types.
- With AuthValidate directory mapping, user lookup by Universal ID is attempted if user lookup by DN fails.

Configure an AuthValidate Directory Mapping

You can configure an AuthValidate directory mapping to authenticate users against one directory and validate users against another directory.

Note: AuthValidate mappings are global.

To configure an AuthValidate Directory Mapping

1. Click Infrastructure, Directory.
2. Click AuthValidate Directory Mappings.

   The AuthValidate Directory Mappings page appears.
3. Click Create AuthValidate Directory Mapping.

   The Create AuthValidate Directory Mapping page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Type the name of the directory that is used to authenticate users in the Authentication Directory field.
5. Select the directory that is used to validate users from the Validation Directory list.
6. Select a mapped DN from the available options.
7. Click Submit.

   The AuthValidate Directory Mapping task is created.

Remove Directory Mappings from Realms

Dissociate the directory mappings from the existing realms, to update, or delete a directory mapping of type authorization or validation.

To remove a directory mapping from an existing realm

1. Click Policies, Domain, Realms.

   The Realms page appears.
2. Select the realm that you want to modify.

   The View Realm page appears.
3. Click Modify.

   The settings and controls become active.
4. Select the directory mapping you want to dissociate from the corresponding list.

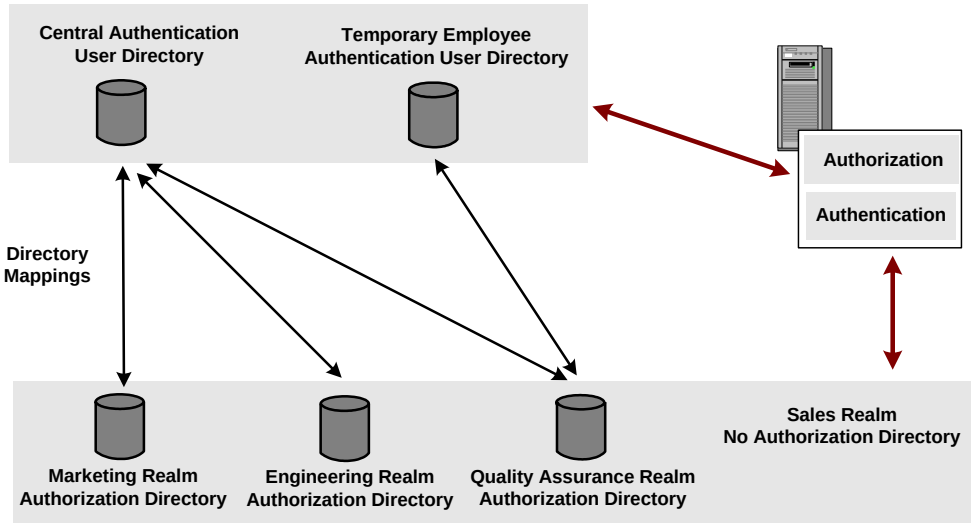
5. Click Submit.

   The directory mapping is removed from the selected realm.

You can follow the same procedure to dissociate the identity mappings of type authorization and validation, from a realm.

Directory Mapping Examples

Multiple directory mappings are required to authenticate and authorize users that have access to network resources. The following figure illustrates a simple sample case where multiple directory mappings are 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. If the requested resource is in the Marketing, Engineering, or Quality Assurance realms, the Policy Server uses one of the authentication directories based on the directory in the session ticket information.

In addition to mapping the directories using Universal ID and Identical DN, you can use Identity Mapping to map the directories using complex user search criteria.

More information:

Realms (see page 501)

Employee Accesses an Engineering Realm Resource

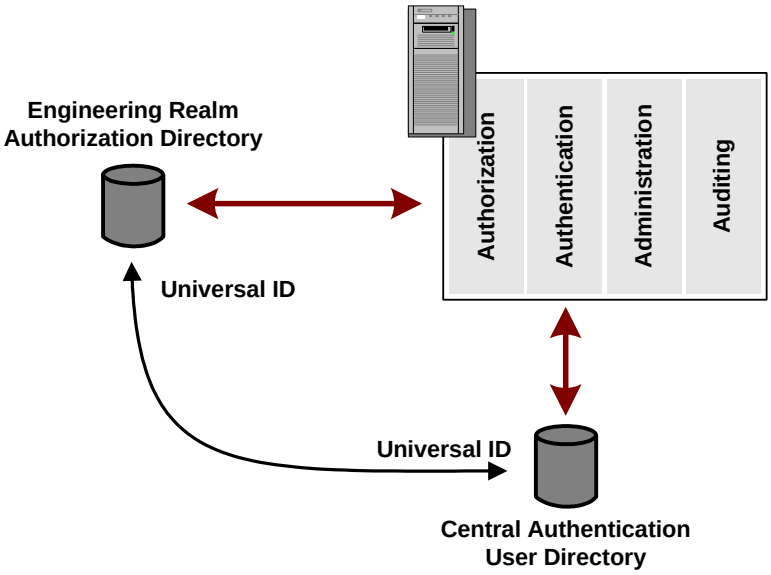
In the SiteMinder protected network described in the previous figure, a regular employee's authentication data resides in the Central Authentication user directory, and the employee attempts to access a resource in the Engineering Realm. When the employee is properly authenticated, the Policy Server 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 SiteMinder 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. SiteMinder 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, SiteMinder attempts to authorize the employee using information in the same directory where the employee was authenticated.

Directory Mapping by Universal ID

In the SiteMinder 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 SiteMinder 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.

Identity Mapping by Complex User Search Criterion

Identity Mapping locates a user based on the session ticket information and how the identity mapping entries are constructed.

The XPS function IDENTITY_MAP finds the Identity Mapping object by name. On finding a user in the target user directory by resolving its entries, returns the value of the attribute specified by the second parameter.

Example:

To get the last name of the user in the target user directory defined by the Identity Mapping named “target”, provide the following:

```
IDENTITY_MAP (“target”, last_name);
```

Directory Mapping and Responses

SiteMinder 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 535).

Chapter 10: Authentication Schemes

This section contains the following topics:

Authentication Schemes Overview

In most cases, when a user attempts to access a network resource, the owner of the network wants to verify the identity of the user. Company employees should be identified to determine which resources they can use. Customers should be identified for personalization of content as they access resources. Even anonymous users should be tracked uniquely, so that their history can be used to provide a quality experience when they once again access the network. To identify a user, SiteMinder employs authentication schemes.

Authentication schemes provide a way to collect credentials and determine the identity of a user. SiteMinder supports a variety of authentication schemes. These schemes range from basic user name/password authentication and HTML forms-based authentication to digital certificate and token authentication. Simple schemes can be used for low risk network resources, while complex schemes may be employed to ensure added security for critical network resources.

Authentication schemes must be configured using the Administrative UI. During authentication, SiteMinder Web Agents communicate with the Policy Server to determine the proper credentials that must be retrieved from a user who is requesting resources.

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.

Supported Authentication Schemes and Password Policies

Some authentication scheme types support Password Policies, while others do not. You can view whether a particular type of authentication scheme supports Password Policies by opening the Authentication Scheme Properties dialog in the Administrative UI. To view a particular authentication scheme type, select it from the drop-down list on the Scheme Common Setup group box and observe the Password Policies Enabled for this Authentication Scheme check box. If the authentication scheme does not support Password Policies, the check box description is dimmed and the check box is unavailable.

To view supported authentication scheme types and whether they support Password Policies without accessing the Administrative UI, see the following table:

Authentication Scheme Type	Type Supports Password Policies?
Anonymous	No
Basic	Yes
Basic over SSL	Yes
Custom	Yes
HTML Forms	Yes
Impersonation	No
MS Passport	No
OpenID	No
RADIUS CHAP/PAP	Yes
RADIUS Server	Yes
SafeWord	No

Authentication Scheme Type	Type Supports Password Policies?
SafeWord and HTML Forms	No
SecurID	No
SecurID and HTML Forms	No
X.509 Client Certificate	No
X.509 Client Certificate and Basic	Yes
X.509 Client Certificate or Basic	Yes
X.509 Client Certificate and HTML Forms	Yes
X.509 Client Certificate or HTML Forms	Yes
Windows Authentication	Yes

Limit Policy Server Search to One User Store during Authentication

A single user can be stored in more than one user directory or database associated with a policy domain. This user has the same password in each user store. During authentication, if the Policy Server finds that the user is disabled in one user store, then by default, it continues searching for the user in all stores associated with the policy domain. The user fails authentication only if the Policy Server finds the user disabled in all associated user stores. The user is authenticated if it is enabled in any associated user store.

This default Policy Server behavior is configurable. To configure the Policy Server to stop searching when it first finds the user disabled in a user store, add the following registry key and set its value to one: ReturnOnDisabledUser.

To limit Policy Server search to one user store during authentication

1. Manually add the registry key ReturnOnDisabledUser:

   Windows

   Add the registry key ReturnOnDisabledUser to the following location:

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion
   \PolicyServer
   ```

   Solaris

   Add the following lines to the sm.registry file:

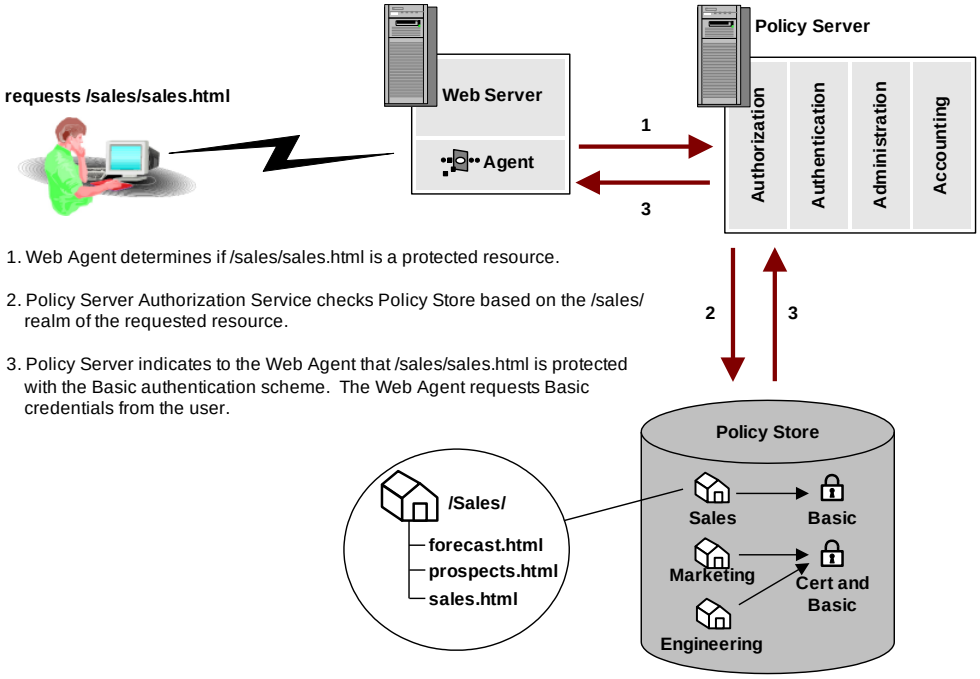
   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion
   \PolicyServer
   ReturnOnDisabledUser=0x1; REG_DWORD
   ```

2. Assign ReturnOnDisabledUser the value of one.

Authentication Scheme Processing

When a user attempts to access a protected network resource, the Policy Server uses the authentication scheme associated with the resource's realm to determine how to identify the user. The authentication scheme specifies the credentials that the user must supply for authentication, as well as the method used by the Policy Server to validate the user's identity.

You can use the Administrative UI to configure authentication schemes and assign the schemes to realms. The following diagram illustrates how an authentication scheme is called when a user attempts to access a protected resource.



In the example above, the user requests the protected resource sales.html from the /Sales/ realm. This realm requires Basic authentication. The Policy Server informs the Web Agent that the resource is protected and requests Basic credentials from the user via the Web Agent, which prompts the user for a user name and password.

More information:

Configure a Realm (see page 507)

Authentication Scheme Types

The Authentication Schemes supported by SiteMinder fall into a number of categories. These categories represent the general characteristics of available authentication methods. Details are provided in the sections of this chapter that discuss each specific authentication scheme.

Basic Authentication Schemes

Basic authentication identifies a user based on a user name and password. The user's identity is stored in a user directory. With a basic authentication scheme, the Policy Server locates a user in a directory based on the user name, then verifies that the password matches the one saved in the user directory (binds the user to the directory). If the user name and password supplied by the user match the data in the user directory, SiteMinder authenticates the user.

The Administrative UI provides authentication scheme templates for the following basic schemes:

- Basic (HTTP Basic)
- Basic over SSL

HTML Forms-based Authentication Schemes

SiteMinder supports the use of customized HTML forms to collect authentication information. In a forms-based authentication scheme, a user can be required to enter additional information such as a Social Security Number, organization, account number, etc. The Policy Server verifies the additional information against user directory attributes before authenticating the user.

Windows Authentication Scheme

Integrated Windows Authentication (IWA) is a proprietary mechanism developed by Microsoft to validate users in pure Windows environments. IWA enforces Single Sign-On by allowing Windows to gather user credentials during the initial interactive desktop login process and subsequently transmitting that information to the security layer. SiteMinder, using the Windows Authentication scheme, secures resources by processing user credentials obtained by the Microsoft Integrated Windows Authentication infrastructure.

Previous versions of SiteMinder supported Windows authentication through the NTLM authentication scheme. However, this support was limited to environments with NT Domains or where the Active Directory service is configured to support legacy NT Domains in mixed mode.

The Windows authentication scheme allows SiteMinder to provide access control in deployments with Active Directories running in native mode, as well as Active Directories configured to support NTLM authentication. The Windows Authentication scheme replaces SiteMinder's previous NTLM authentication scheme. Existing NTLM authentication schemes continue to be supported and can be configured using the new Windows Authentication scheme.

The NTLM authentication scheme can be used for resources that are protected by Web Agents on IIS Web servers, and whose users access resources via Internet Explorer Web browsers. This scheme relies on a properly-configured IIS Web server to acquire and verify a user's credentials. The Policy Server bases authorization decisions on the user's identity as asserted by the IIS server.

X.509 Client Certificate Authentication Schemes

SiteMinder supports the use of X.509 V3 client certificates. Digital certificates act as cryptographic proof of a user's identity. Once a certificate is installed on a client, that certificate can be used to verify the identity of a user who is accessing a resource. Certificate authentication uses SSL communication and can be combined with basic authentication to provide an even higher level of access security.

The Administrative UI provides authentication scheme templates for the following certificate-based authentication schemes:

- X.509 Client Certificates
- X.509 Client Certificates and Basic
- X.509 Client Certificates or Basic
- X.509 Client Certificates and HTML Forms
- X.509 Client Certificates or HTML Forms

Note: In the case of certificate-only authentication schemes, the web agent returns HTTP Error 403: Access Denied/Forbidden for any failed authentication or authorization attempt. This is because there is no way for the web agent to challenge the user for a new certificate.

Proxy Authentication Schemes

Proxy authentication schemes are schemes that use the Policy Server as a proxy or substitute for the server required by a third party authentication product. With a proxy scheme, the Policy Server performs the authentication function of the third party server using scheme specific libraries.

SiteMinder supports the following proxy authentication schemes:

- RSA ACE/Server (used with SecurID tokens)
- RSA ACE/Server (used with SecurID tokens) with HTML forms support for resetting passwords
- Secure Computing SafeWord Server
- RADIUS Server

Digest Authentication Schemes

A digest authentication scheme reads an encrypted user attribute string stored in a directory, then compares the string to the encrypted string it receives from the user. If the encrypted strings match, the Policy Server authenticates the user. A digest scheme compares a string encrypted on a client workstation to an encrypted string on a server without using an encrypted transmission.

SiteMinder supports the following digest authentication schemes:

- RADIUS CHAP
- RADIUS PAP

Anonymous Authentication Schemes

An anonymous authentication scheme allows non-registered users to access specific Web content. When a user accesses a resource that has anonymous authentication, SiteMinder assigns the user a Global User Identification (GUID). SiteMinder places this GUID in a persistent cookie on the user's browser so that the user can access specific resources without being challenged to authenticate.

Custom Authentication Schemes

If SiteMinder does not provide a method of authentication that you want to use, you can use CA's APIs to develop a custom authentication scheme.

Note: If you have installed the Software Development Kit, see the API Reference Guide for C or the API Reference Guide for Java for more information about creating custom authentication schemes.

Authentication over SSL

Authentication can be configured to run over a Secure Sockets Layer (SSL) connection. SSL is a method of establishing an encrypted connection between a client and a server using digital certificates to initiate the connection, and to establish a proof of identity.

The following authentication scheme types are configured to use an SSL connection:

- Basic*
- HTML Forms*
- X.509 Client Certificates
- X.509 Client Certificates and Basic
- X.509 Client Certificates or Basic
- X.509 Client Certificate or HTML Forms
- X.509 Client Certificate and HTML Forms

Note: An asterisk denotes that an SSL connection is optional.

Persisting Authentication Context Data

In addition to storing user data in the session ticket, you can optionally specify that SiteMinder persist the authentication context data in the session store. SiteMinder creates session variables and treats them as if they were session ticket fields that are named after the session variables. The Policy Server can access session variables in the session store, which can affect authentication decisions.

You can configure Responses and Policies to manipulate, store, or send back session context attributes available from the persisted authentication data. The information is retrieved from the session store and sent to the web agent. The web agent can store the data again, or can provide it to the authorization engine for evaluation. In addition, you can configure your own session variables and can use them for authorization.

To save the authentication context information, select the Persist Authentication Session Variables check box in the Scheme Setup section. This option is available for the Custom scheme, the SAML authentication schemes, and the X.509 Certificate schemes.

Important! Persisting authentication data in the session store creates a degradation in the authentication time. Only select this option when you really intend to use the variables at a later time for authentication decisions. Otherwise, you can possibly experience a significant performance impact with no benefit.

Protection Levels

Authentication schemes require a protection level. This level ranges from zero to 1000. A higher number indicates that the scheme provides higher level of protection. When users authenticate successfully against a scheme, they can access any resource with a protection level equal to or below the current authentication scheme. Users still require authorization for a resource to gain access to it.

Note: Anonymous authentication schemes always have a protection level of zero. Custom schemes have a protection level between zero and 1000. All other authentication schemes have a protection level between1 and1000.

For example, a set of resources is available to all network users, you can assign a Basic (user name and password) authentication scheme. For revenue information that is available only to corporate executives, you can assign an X.509 client certificate scheme with a higher protection level. A user who has already authenticated with a user name and password can authenticate a second time with a digital certificate to access the revenue information.

Sometimes the predefined protection level of the authentication scheme can be inadequate. For example, in a federation scenario, the asserting party can possibly require a different protection level to accommodate the relying party. In such cases, the administrator can specify that a protection value in the authentication scheme library overrides the protection level specified in the UI. In this case, SiteMinder writes the value in the library to the user session ticket. Select the Allow Protection Override check box in the Scheme Common Setup section of the Create Authentication Scheme dialog for Custom and SAML authentication schemes.

More information:

Domains (see page 493)
Policies (see page 561)

Authentication Schemes and Credential Requirements

The following table lists all supported authentication schemes and their credential requirements:

	Credential Requirements				
Authentication Schemes	**Directory User Name**	**Directory Password**	**Code from Token**	**X.509 Certificate**	**User Profile Attributes**
Anonymous					
Basic	yes	yes			
Basic over SSL	yes	yes			
Custom	optional	optional	optional	optional	optional
HTML Forms (over SSL optional)	custom credentials	custom credentials			optional
Impersonation	yes				optional

	Credential Requirements				
Authentication Schemes	**Directory User Name**	**Directory Password**	**Code from Token**	**X.509 Certificate**	**User Profile Attributes**
MS Passport	yes	yes			yes
NTLM or Windows	yes*	yes*			
RADIUS CHAP/PAP	yes	yes			
RADIUS Server	yes	yes			
SafeWord Server	yes	yes			
SafeWord and Forms	yes	yes			optional
SecurID	yes		yes		
SecurID and Forms	yes		yes		optional
TeleID	yes		yes		
X.509 Client Certificate				yes	
X.509 Client Certificate and Basic (uses SSL)	yes	yes		yes	
X.509 Client Certificate or Basic (over SSL optional)	yes for Basic	yes for Basic		yes for Certificate	
X.509 Client Certificate and HTML Forms	custom credentials	custom credentials		yes	optional
X.509 Client Certificate or HTML Forms	custom credentials for HTML Forms	custom credentials for HTML Forms		yes for Certificate	optional for HTML Forms

*For NTLM or Windows, when trying to access a resource, SiteMinder does not prompt the user to enter a username and password. This scheme relies on a properly-configured IIS Web server to acquire and verify a user's credentials. The Policy Server bases authorization decisions on the user's identity as asserted by the IIS server.

Set Up an Authentication Scheme Object in the Policy Server User Interface

To setup a new authentication scheme in the Administrative UI, components should be configured in the following order:

1. Web Server (only for certificate, SSL, and HTML forms-based schemes)
2. Policy Server (including Certificate Mapping for X.509 certificate schemes)

More information:

Certificate Mapping for X.509 Client Authentication Schemes (see page 403)

Web Server

In order for a SiteMinder Web Agent to support any SSL-based Authentication Scheme, a Web Server must be configured to support SSL.

Note: More information on Web server configuration exists in the Policy Server Installation Guide for instructions on Web server configuration.

Policy Server

After you configure your Web servers to support authentication schemes, configure the Policy Server to support the schemes.

More information:

Authentication Schemes Overview (see page 301)

Multiple Instances of a Single Authentication Scheme Configuration

You can configure multiple instances of most authentication schemes in the Administrative UI. For example, you might create multiple HTML forms-based schemes to process login, forgotten password requests, logout, etc. If you create multiple instances of a scheme type, be sure to set protection levels to reflect your security requirements.

Basic Authentication Schemes

The Policy Server installation process automatically configures a Basic authentication scheme. This scheme verifies a user's identity according to a user name and password that are passed to a user directory service for authentication. For LDAP user directories, this is done via an LDAP "Bind" operation. The HTTP Basic Authentication protocol is used to deliver credentials from the browser to the Web server protected by the SiteMinder Web Agent. Basic authentication schemes are supported only with ASCII characters.

When a user attempts to access a resource protected by Basic authentication, the SiteMinder Agent prompts the user to enter a user name and password. When the user enters a name and password, the Agent passes the credentials to the Policy Server over an encrypted connection, and the Policy Server matches the name against the users contained in the directories attached to the policy domain that contains the resource. When the Policy Server finds a matching user name, it compares the password in the user directory to the password supplied by the user. If the passwords match, the user is authenticated, and the Policy Server sends a message to the Web Agent indicating that the Agent may proceed. If the authentication fails, the user is challenged to re-enter credentials.

Note: This scheme does not provide encrypted credential delivery by default. Instead, the user name and password are delivered from the browser to the Web server protected by the Web Agent via the standard HTTP Basic protocol, unless every protected URL on the Web server is set up to require SSL. However, communication between the Web Agent and the Policy Server always takes place over an encrypted connection. For an encrypted authentication scheme based on simple user names and passwords, see Basic Over SSL Authentication Schemes (see page 314).

Realms that you create in the Administrative UI use the Basic authentication scheme by default. You can change the authentication scheme when you create a new realm or modify an existing realm.

For an additional level of security with Basic authentication, you can create password policies. This SiteMinder feature allows you to manage password rules.

More information:

Domains (see page 493)
Realms (see page 501)
Password Policies (see page 667)

Review Basic Scheme Prerequisites

Verify that the following prerequisites are met before configuring a Basic authentication scheme:

- Client user name and password information exists in a user directory.
- A directory connection exists between the Policy Server and the user directory.

More information:

User Directories (see page 157)

Configure a Basic Authentication Scheme

Configure a Basic authentication scheme in the Administrative UI to verify user identities against user names and passwords that exist in the user directory.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and protection level.
6. Select Basic Template from the Authentication Scheme Type list.
7. Click Submit.

   The authentication scheme is saved and can now be assigned to a realm.

Basic Over SSL Authentication Schemes

The Basic Over SSL Authentication Scheme verifies a user's identity by passing a user name and password credentials to a user directory in a process similar to Basic authentication. However, credential delivery is always done over an encrypted Secure Sockets Layer (SSL) connection even if the protected URLs are not setup to require SSL. The SiteMinder Web Agent accomplishes this by redirecting the user's browser to establish an SSL connection prior to credential delivery. After the credentials are delivered the Web Agent redirects the browser back to the original URL.

For an additional level of security with Basic over SSL authentication, you can create password policies. This SiteMinder feature allows you to manage password rules.

More information:

Basic Authentication Schemes (see page 305)
Password Policies (see page 667)

Verify That Basic over SSL Authentication Scheme Prerequisites Are Met

Verify that the following prerequisites are met before configuring a Basic over SSL authentication scheme:

- Client user names and passwords must exist in a user directory.
- A directory connection exists between the Policy Server and the user directory.
- An X.509 Server Certificate is installed and SSL configured on the SSL web server.
- The network must support an SSL connection to the client browser using the HTTPS protocol.
- A SiteMinder Web Agent must be installed on the web server to which requests are redirected for SSL. The Web Agent enables the server to handle the .scc MIME type required by the authentication scheme.

More information:

User Directories (see page 157)

Configure a Basic Over SSL Authentication Scheme

You can use a Basic Over SSL authentication scheme to verify user identities against the user names and passwords that exist in the user directory. Credential delivery is completed over an encrypted Secure Sockets Layer.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and protection level.
6. Select Basic over SSL Template from the Authentication Scheme Type list.

   Scheme-specific settings open.
7. Enter Server Name and Target information.
8. Click Submit.

   The authentication scheme is saved and may be assigned to a realm.

HTML Forms Authentication Schemes

HTML Forms authentication schemes provide a method for authentication based on credentials gathered in a custom HTML form. This flexible means of credential collection allows you to:

- Provide a “branded” look, perhaps including a company logo.
- Substitute custom labels for user name and password collection (for example, if users think in terms of an account number and a PIN rather than a name and password).
- Provide authentication based on credentials other than a user name and password (via user directory attributes). In this case, an authentication scheme library on the Policy Server machine maps the user’s data to a DN. By mapping the user to a DN, the Policy Server can match an attribute list to the appropriate values in a user directory. This process is called back-end mapping.

- Provide authentication based on credentials that include user attributes in addition to the user name and password. This is considered additional attribute verification. A custom authentication scheme library is not required for additional attribute verification.

  For example, a custom form can be used to collect a name and a secret phrase for users who forget their password.

- Provide multiple HTML forms for login, logout, forgotten passwords, etc.

  Note: HTML Forms authentication schemes are supported with multi-byte characters.

Multiple *Forms-based* Authentication Schemes can be configured in a Policy Server installation. Each scheme consists of the following components:

Forms Credential Collector (FCC)

The FCC process files are composed in a simple mark-up language that includes HTML and some custom notation.

Each HTML Forms scheme must have its own .fcc file. This file contains the custom form definition and additional information that the FCC uses to process HTML Forms authentication. The FCC extracts credentials that a user enters in the custom form generated from the .fcc file.

For the HTML Forms authentication scheme, the default extension for .fcc files is .fcc. If you want to use a different extension:

- For Apache or iPlanet Web servers, configure your Web server to use that extension.
- For Domino or IIS Web servers, specify that extension in the FCCExtensions parameter of your Web Agent configuration file or object. For more information on Web Agent configuration parameters, see the *Web Agent Configuration Guide*.

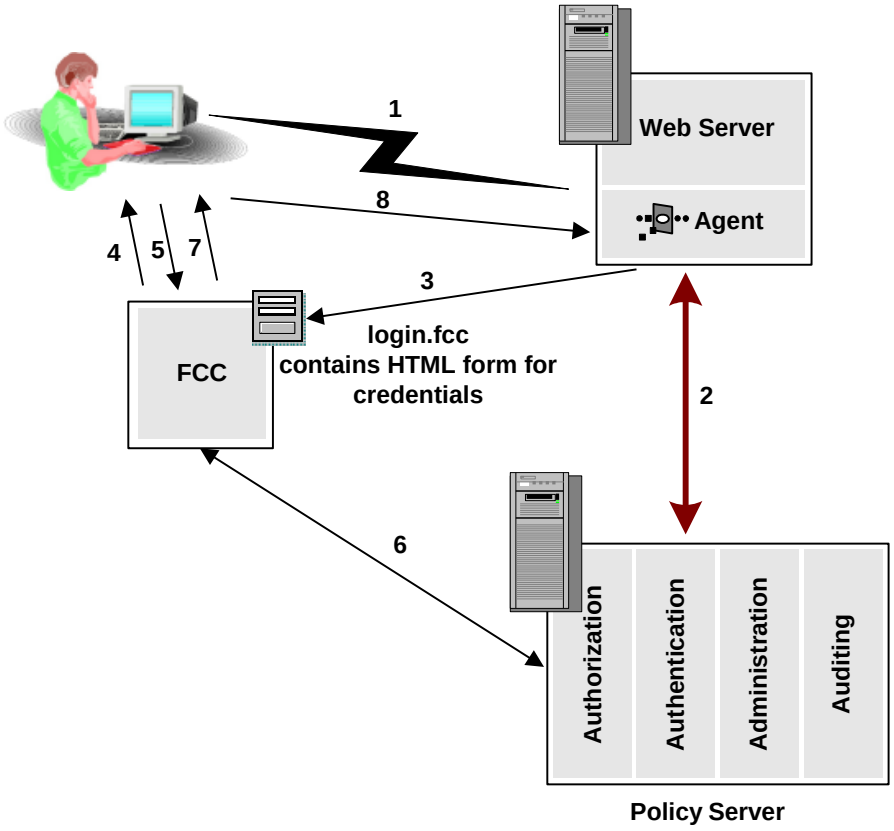
.unauth file

SiteMinder displays the contents of this file to users who exceed the maximum number of failed authentication attempts specified by the authentication scheme. A .unauth file should exist for each .fcc file. For example, if you have a login.fcc file on a Web server, you should also have a login.unauth file in the same location.

If an smerrorpage variable has been defined in the .fcc file, the .unauth file is not required.

Authentication Scheme Library

This is a shared library that runs on the Policy Server machine and performs authentications.



The previous diagram describes the process for HTML Forms authentication.

1. A user requests a resource contained in a realm protected by HTML Forms authentication.
2. The Web Agent contacts the Policy Server and determines that the user's request must be redirected to the credential collector.
3. The Web Agent redirects the request to the URL of the credential collectorfile.
4. The credential collector displays the form described in the .fcc file in the user's browser.
5. The user fills out the custom form and Posts (submits) the form. The credential collector processes the credentials.
6. The credential collector (FCC) logs the user into the Policy Server. The Policy Server returns user session data to the credential collector.

7. If the user is authenticated, the credential collector creates a session cookie, passes the session cookie to the browser and redirects the user to the resource that he or she originally requested.
8. The user uses the session cookie to authenticate. Then, the Web Agent handles user authorization.

More information:

SiteMinder FCC Files (see page 320)

Review the HTML Forms Scheme Prerequisites

Verify that the following prerequisites are met before configuring an HTML Forms authentication scheme:

- A customized .fcc file resides on a Web Agent server in the cookie domain in which you want to implement HTML Forms authentication. CA provides sample .fcc files under the Samples/Forms subdirectory where you installed your Web Agent.
- A customized .unauth file resides on the Web Agent server.

  Note: This file is not required if the .fcc file uses the smerrorpage directive.
- A directory connection exists between the Policy Server and the user directory.
- The default HTML forms library is installed. This library handles HTML Forms authentication processing:
  - SmAuthHTML.dll on Windows
  - smauthhtml.so on Solaris

  These files are installed automatically when you configure a Web Agent.
- (Sun Java Systems) If you are using a Sun Java Systems web server, increase the value of the StackSize parameter in the magnus.conf file to a value greater than 131072. Failing to change the value causes the web server to dump its core and restart each time an authentication request is made using forms.

More information:

SiteMinder FCC Files (see page 320)
User Directories (see page 157)

Custom Authentication Scheme Library Writing and Installation

The user name and password data that the FCC collects are passed to the Policy Server, which passes them to the Authentication Scheme library.

Unless back-end mapping is required, the SmAuthHTML Authentication Scheme library can be used. SmAuthHTML it is distributed with the Policy Server and already installed on the Policy Server system.

Note: Back-end mapping requires a custom Authentication Scheme library. If you have installed the software development kit, see the API Reference Guide for C.

If you have written a custom Authentication Scheme and you want to gather more data than the username and password, the FCC should pack that data into the username and password fields (each of which must be less than 511 characters long). The custom Authentication Scheme library must then be able to unpack the data and map it to the user name and password.

The FCC can be installed on the same system as the Policy Server.

HTML Forms Authentication Templates

The templates for forms authentication are text files that use an extended version of HTML. These files are referred to as .fcc files.

The default extension for .fcc files used in an HTML Forms authentication scheme is .fcc. However, you can use a different extension:

- For Apache Web servers, configure your Web server to use the extension you want.

  Note: More information exists in the *Web Agent Installation Guide*.

- For Domino or IIS Web servers, specify the extension that you want in the FCCExtensions parameter of your Web Agent configuration file or object.

  Note: For more information about Web Agent configuration parameters, see the *Web Agent Configuration Guide*.

The Web Agent includes sample English, French and Japanese .fcc files, which you can customize. These files are located in the following directories:

Language	Directory
English	Windows: Web Agent\Samples\Forms UNIX: webagent/samples/forms
French	Windows: Web Agent\Samples\Formsfr UNIX: webagent/samples/formsfr
Japanese	Windows: Web Agent\Samples\Formsja UNIX: webagent/samples/formsja

By default, the product uses the English .fcc files.

To use the sample .fcc files, create a form authentication scheme that specifies the appropriate target directory.

Secure HTML Forms Authentication Templates

Secure versions of the HTML forms authentication templates are also available. The secure HTML forms authentication templates differ from the standard versions in the following ways:

- Secure versions do not display the username in returned messages
- Secure versions include a Logout hyperlink in the top right side corner of the form template which logs out the user and redirects them to the custom logoff page
- Autocomplete is turned off for all text fields in secure versions

The secure template files are located in the following directories:

- Windows*:* webagent\secureforms
- UNIX: webagent/secureforms

To use the secure versions of the HTML forms authentication templates, copy the files from the secureforms directory to the following location, replacing the standard versions there:

- Windows*: weba*gent\samples\forms
- UNIX: webagent/samples/forms

Note: Localized versions of the secure HTML forms authentication templates are not available; only English language versions are supplied.

SiteMinder FCC Files

The SiteMinder Forms Credential Collector (FCC) incorporated into SiteMinder Web Agents reads template files called .fcc files. The .fcc files are written using standard HTML tags and a small amount of proprietary notation required by SiteMinder to verify attributes and take advantage of custom features described later in this section.

Important! If you create or edit an .fcc file on a Windows system and move that file to a UNIX system, your UNIX system may append ^M at the end of lines of text. These characters, which identify the file as a Windows text file, cause .fcc files to fail during authentication. When moving files from Windows text editors to UNIX systems, be sure to examine the files and remove the appended characters. To avoid this situation, create and edit .fcc files that will be used in a UNIX environment on a UNIX system.

For the HTML Forms authentication scheme, the default extension for .fcc files is .fcc. If you want to use a different extension:

- For Apache or iPlanet Web servers, configure your Web server to use that extension.
- For Domino or IIS Web servers, specify that extension in the FCCExtensions parameter of your Web Agent configuration file or object. For more information on Web Agent configuration parameters, see the Web Agent Guide.

When a user requests a resource protected by an HTML Forms scheme, the Web Agent redirects the user to an .fcc file. The .fcc file invokes the FCC on the Web server in order to collect credentials from the user via a customized form. The FCC generates a browser page and builds a user name and password based on the contents of the .fcc file. The file resides in a Web server's name space and is accessed like any HTML file. The .fcc file may contain two parts. Both parts are optional.

The first part of the FCC contains directives that are used when executing a POST operation on the .fcc file. The directives are never passed to the client. They must be at the beginning of the file and are of the form: @name=value

The name is the name of a variable. The value is the variable's value. The value may contain strings of the form: %name1%. This will be replaced by the value of the variable associated with name1.

The second part of the .fcc file contains HTML code that is returned when a GET operation is performed on the .fcc file. This part may include text in the form "$$name$$", including the quotation marks (") that will be replaced by the value associated with name. The name is not case sensitive.

The hidden inputs listed in the following figure are used to hold state for the credential collectors:

Name	To dynamically set, use the value:*	Data preserved
target	"$$target$$"	Resource that a user wants to access
smauthreason	"$$smauthreason$$"	Reason for a login failure
postpreservationdata	"$$postpreservationdata$$"	Data that a user submits through a post request.
smagentname	$$smagentname$$	Agent name used for logging user in.

*Be sure to enter the quotation marks (").

At a minimum, an .fcc file must collect the following:

- User name
- Password
- Target

Important! If users will be submitting post requests to a resource protected by an authentication scheme that uses a credential collector (see the following figure), use the postpreservationdata input. Otherwise, data that users attempt to post to the requested resource will be lost.

Schemes
Basic Over SSL Authentication Schemes
HTML Forms Authentication Schemes
X.509 Client Certificate Authentication Schemes
X.509 Client Certificate and Basic Authentication Schemes
X.509 Certificate or Basic Authentication Schemes
X.509 Client Certificate and HTML Forms Authentication Schemes
X.509 Client Certificate or HTML Forms Authentication Schemes

The following is an example of a valid (though simple) .fcc file:

Creates a user name based on form data

Collects the user's password

Collects the user's organization

```
@username=uid=%USER%,ou=%GROUP%,o=%ORG%
@smretries=3
<html>
<head><title>Sample Login Form</title><head>
<body>
<h3> Please enter your login credentials</h3>
<form method=post><table>
<tr>
  <td>User Name:</td>
  <td><input type=text name=USER></td>
 </tr>
 <tr>
  <td>Password:</td>
  <td><input type=password name=PASSWORD></td>
 </tr>
<tr>
  <td>Group: </td><td><input type=text name=GROUP></td>
</tr>
<BR>
<tr>
<td>Organization: </td>
<td><Select Name=ORG SIZE=1>
  <OPTION>Company A
  <OPTION>Company B
  <OPTION>Company C
  <OPTION>Company D
</SELECT></td>
 <input type=hidden name=target value="$$target$$">
 <input type=hidden name=smauthreason value="$$smauthreason$$">
<tr><td><input type=submit value=LOGIN></td></tr>
</table></form></body>
</html>
```

The file above is the usermap.fcc sample file included with the default installation of SiteMinder Web Agents.

The .fcc file above creates a distinguished name (DN) for the user based on the information the user enters in the User Name field and the Organization drop-down list of the HTML form. This DN is the user name authentication credential. The user's password is collected from the Password field of the HTML form. The hidden realm and target input values are also collected so that the user can be directed to the appropriate resource when authentication is complete.

More information:

HTML Forms Authentication Schemes (see page 315)

How Name/Value Pairs are Generated in FCC Files

The login.fcc template generates a namespace for use in $$*name*$$ expansions and for use by special name value pairs. If a name is entered more than once the last value wins. Name/value pairs are added to this namespace in the following order:

1. Name/Values from the query string (on Get only).
2. The name smtarget is created by copying the value of the target (on Get only).
3. Name/Values from the post data.
4. Name/Values from the cookies.
5. Name/Values from the @ directives (on POST only).
6. Responses named in the “smheaders” name value pair (on POST only).

Special Name/Value Pairs

An FCC can interpret a number of special name/value pairs (@directives) that invoke nonstandard processing. The special @directives and their meanings follow:

username

Name for the login user name.

password

Password to perform the login.

target

Resource to access after login.

smheaders

Colon separated list of response names to include in the namespace. The colon separated list must contain an entry for each header that you want to include in a transaction. For example, if you want to pass the value of header1 and header2 as part of a transaction, include the following line in your FCC:

@smheaders=header1:header2

smerrorpage

If there is an error on a POST to the custom form, the user browser is redirected to this page. If this special value is not specified in a .fcc file, the system uses the .unauth file that is associated with the .fcc file as the error page.

smretries

Specifies the maximum number of login attempts allowed. If you set this directive to 0, the number of retries is unlimited. If you set the number to 1 or greater, that is the number of retries allowed.

Note: If users log in using a POST to an .fcc form, it may appear that the user is given additional attempts to log in beyond the value of the smretries directive. However, the user is allowed access only if valid credentials are entered in the number of attempts that smretries specifies.

smpasswordfcc

Determines whether data is posted from the Password Services FCC file or from a different FCC file.

Default: 1

Important! We recommend that you use the default value. The SafeWord authentication scheme may not work properly if the default value is changed.

smusrmsg

Text that describes why the user was challenged / failed to login.

smauthreason

Reason code that is associated with a login failure.

smsavecreds

Set to Yes to save user credentials in a persistent cookie on the user browser.

smsave

Colon separated list of names to be saved as persistent cookies.

save

Another name for smsave.

smtransient

Colon separated list of names to be saved as transient cookies.

smagentname

Specifies the agent name that is supplied to the Policy Server when a user enters credentials and submits the form for authentication. If the Agent parameter, FCCCompatMode=NO, specify a value using this directive.

smlogout

Logs a user out of the system, similar to the LogoffUri parameter. By placing @smlogout=true in your .fcc template, the FCC logs a user out and redirect the user to the target. As such, the @smlogout directive is typically used with the @target directive *(@target=<yoururlhere>*).

urlencode(*name*)

Replaced by the URL encoded value of the named variable.

Note: If you expect the additional attributes or the Password to contain special characters (" . & = + ? ; / : @ = , $ %), URL-encode each additional attribute value in the .fcc template file. The template uses US-ASCII encoding.

urldecode(*name*)

Replaced by the URL decoded value the named variable.

Note: The “sm” prefix for name/value pairs is reserved for additional special names that the system requires. When creating names for your login page do not use the “sm” prefix.

Localization Name/Value Pairs

The .fcc template files include two localization parameters:

smlocale

Used to determine the language used in the HTML forms that collect user information or display status messages.

The value that is paired with **smlocale** corresponds to part of the name of a localization properties file. The localization properties file contains IDs mapped to text strings in the specified language.

smlocale values have the following format:

COUNTRY-LANGUAGE

For example, the value for **smlocale** for United States English is:

SMLOCALE=US-EN

smenc

Contains information that tells the browser what language encoding to use. Changing the default value for this variable overrides the encoding set in the following META tag:

<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">

Collect Additional Attributes

In addition to the username and password, you can collect additional attributes from users, such as an email address or job title.

To collect additional attributes

1. Specify the attribute or attributes by configuring the HTML forms authentication scheme and the associated .fcc template file.
2. Configure the HTML forms authentication scheme by specifying new attributes in the Additional Attribute List fields of the Scheme Setup dialog.

   Note: If you expect the additional attributes or the Password to contain special characters (" . & = + ? ; / : @ = , $ %), URL-encode each additional attribute value in the .fcc template file. The template uses US-ASCII encoding.
3. Modify the .fcc template file to collect additional attributes.

   Add the following line at the beginning of the file:

   ```
   @password=PASSWORD=%PASSWORD%&newattr1=%newattr1%&newattr2=%newattr2%
   ```

   If the additional attributes have special characters, the line should look like the following sample:

   ```
   @password=PASSWORD=%urlencode(PASSWORD)%&newattr1%=%urlencode(newattr1)%&newattr2=%urlencode(newattr2)%
   ```

   newattr1=%newattr1%

   Represents the first additional attribute.

   newattr2=%newattr2%

   Represents the second additional attribute.

   The value before the equal sign is the attribute name and the value between the percent sign (%) sign is the attribute value.

   The FCC parses the names of the new attributes from the attribute values.

   Note: Append additional attributes to the @password directive with the ampersand (&) character.

When you add attributes to the template file, consider the following points:

- The attribute name, which is identified by the %*attribute_name*% format, must match the input name entry that you also add to the file. For example, if you add the new attribute address:

  ```
  @password=PASSWORD=%PASSWORD%&mail=%address%
  ```

  or

  ```
  @password=PASSWORD=%urlencode(PASSWORD)%&mail=%urlencode(address)%
  ```

  you would add a line to the .fcc file similar to the following:

  ```
  <input name="address" type="text">
  ```

- The name of the additional attribute must match the name of the attribute in the user directory. For example, to collect email addresses from an LDAP directory, specify one of the following:

  @password=PASSWORD=%PASSWORD%&mail=%address%

  or

  @password=PASSWORD=%urlencode(PASSWORD)%&mail=%urlencode(address)%

  where mail is the name of the LDAP attribute that stores email addresses.

- Do not add additional spaces after the value you specify for the @password directive. Additional spaces may be interpreted as meaningful characters and may prevent users from being authenticated.

- The name of the attribute in the HTML forms authentication scheme must match the name of the additional attribute in the .fcc file.

  For example, to add the attribute mail (from the example above) to the authentication scheme, enter the following in the Additional Attributes List field:

  AL=PASSWORD,mail

  Note: The additional attribute names are case-sensitive.

More information:

HTML Forms Authentication Schemes (see page 315)

Tell Users Why Login Failed

The default behavior of forms-based authentication is to redirect unauthenticated or unauthorized users back to the original login form. You can configure the smretries directive (@smretries) to provide users with additional login attempts. However, the default behavior does not let you display a message that informs users why the login failed.

The Web Agent ships with the DynamicRetry.fcc and DynamicRetry.unauth files. This pair of .fcc files changes the behavior of the redirect. The login page (DynamicRetry.fcc) is configured to send users to the unauthorized page (DynamicRetry.unauth) after one failed login attempt. The unauthorized page is a different template file than the login file. As a result, the unauthorized page can contain a message stating why the login failed. By default, the unauthorized page is configured with a message that informs users that they have entered invalid credentials for the resource they are attempting to access.

Note: You can change this message by opening DynamicRetry.unauth and updating the text in between the h3 tags.

To tell users why login failed, specify the target path to the DynamicRetry.fcc file when configuring the authentication scheme. The default path to the DynamicRetry.fcc file is *agent_home*\samples\forms\DynamicRetry.fcc

agent_home

Specifies the Web Agent installation path.

Consider the following limitations when using the DynamicRetry pair of .fcc files:

- You cannot count user login attempts based on the SMTRYNO cookie.
- You cannot limit the number of login attempts.

  Note: You can use this method in combination with Password Services to disable users with a password policy after a specified number of failed attempts. More information about password policies exists in the Password Policies.

Configure an HTML Form Authentication Scheme

You can use an HTML Forms authentication scheme to authenticate users with a custom HTML form.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.

3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.

4. Click OK

   The Create Authentication Scheme page appears.

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

5. Enter a name and protection level.

6. Select HTML Form Template from the Authentication Scheme Type list.

   Scheme-specific fields and controls appear.

7. Enter Web Server Name, Target, and Additional Attribute List information.

   Note: Verify that the .fcc file you specify in the Target field complies with the guidelines listed in the prerequisites.

8. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

Enable Non-browser Client Support

You can configure HTML Form schemes that collect Basic (username and password) credentials to authenticate users using nonbrowser HTTP clients. These clients can be developed using Perl scripts, C++, and Java programs that communicate using HTTP protocol.

Custom clients must send the basic credentials with the initial request through an HTTP Authorization header or SiteMinder does not authenticate the users. If the credentials are not sent through an HTTP Authorization header, SiteMinder redirects to the HTML Form scheme without nonbrowser client support.

Follow these steps:

1. Open the HTML Form authentication scheme.
2. Select the Support nonbrowser clients check box.
3. Click Submit.

   Nonbrowser client support is enabled.

Windows Authentication Schemes

Integrated Windows Authentication (IWA) is a proprietary mechanism developed by Microsoft to validate users in pure Windows environments. IWA enforces Single Sign-On by allowing Windows to gather user credentials during the initial interactive desktop login process and subsequently transmitting that information to the security layer. SiteMinder, using the Windows Authentication scheme, secures resources by processing user credentials obtained by the Microsoft Integrated Windows Authentication infrastructure.

Previous versions of SiteMinder supported Windows authentication through the NTLM authentication scheme. However, this support was limited to environments with NT Domains or where the Active Directory service is configured to support legacy NT Domains in mixed mode.

The Windows authentication scheme allows SiteMinder to provide access control in deployments with Active Directories running in native mode, as well as Active Directories configured to support NTLM authentication. The Windows Authentication scheme replaces SiteMinder's previous NTLM authentication scheme. Existing NTLM authentication schemes continue to be supported and can be configured using the new Windows Authentication scheme.

Note: In some circumstances, you may want to combine Windows User Security Context functionality with other authentication schemes instead of using the Windows authentication scheme.

The Windows authentication scheme can be used for resources that are protected by Web Agents on IIS Web servers, and whose users access resources via Internet Explorer Web browsers. This scheme relies on a properly-configured IIS Web server to acquire and verify a user's credentials. The Policy Server bases authorization decisions on the user's identity as asserted by the IIS server.

More information:

Windows User Security Context Requirements (see page 114)

Review Kerberos Support Considerations

Kerberos is used for domain authentication in supported Windows platforms with user and computer principals stored in Active Directory. Kerberos provides a platform independent architecture for authentication and single sign-on.

SiteMinder supports Kerberos authentication. See this chapter (see page 871) for information about Kerberos configuration.

Verify that Windows Authentication Prerequisites Are Met

Verify that the following prerequisites are met before configuring a Basic over SSL authentication scheme:

- For legacy WinNT directories or Active Directory in mixed mode:
  - The user directory connection that you create in the Administrative UI specifies the WinNT namespace.
  - The requested resources can be located on any type of web server. However, the authentication server and the web agent protecting those resources must be on a Microsoft IIS web server.
- For Active Directories running in native mode:
  - User data resides in an Active Directory.
  - User directory connections must specify either an LDAP or AD namespace.
  - The requested resources can be located on any type of web server. However, the authentication server and the web agent protecting those resources must be on a Microsoft IIS web server.
  - Client and server accounts are enabled for delegation.
- Users must log in using an Internet Explorer web browser (4.0 or later).
- To work on IIS in Windows, the "Verify that file exists" option in the Wildcard Application Maps must not be set.
- Windows Authentication schemes also require that any virtual directory on the IIS web server that contains the creds.ntc file remain unprotected.
- Internet Explorer browser options are configured for automatic logon with the current username and password of the user.

Review Windows Authentication Scheme Considerations

The IIS web server, not the Policy Server, performs authentication-based on credentials it receives from the Internet Explorer web browser. Therefore, you cannot use the OnAuthAttempt authentication event to redirect users who do not exist in the user store.

Configure a Windows Authentication Scheme

You can use a Windows authentication scheme to authenticate users in a Windows environment.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and protection level.
6. Select Windows Authentication Template from the Authentication Scheme Type list.

   Scheme-specific settings appear.
7. Enter Server Name, Target, and User DN information. If your environment requires NT Challenge/Response authentication, obtain the following values from the agent owner:

   Server Name

   The fully qualified domain name of the IIS web server, for example:

   server1.myorg.com

   Target

   /siteminderagent/ntlm/smntlm.ntc

   Note: The directory must correspond to the virtual directory already configured by the installation. The target file, smntlm.ntc, does not need to exist and can be any name that ends in .ntc or the custom MIME type that you use in place of the default.

   Library

   smauthntlm
8. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

Information Card Authentication Schemes

Using the Information Card Authentication Scheme (ICAS) feature of SiteMinder, you can create multiple information card authentication schemes. Each one is configured as a custom authentication scheme.

Introduction to Information Cards

Information cards are like the physical cards that we carry in our wallets. Each information card represents a set of identity information. For example, an information card that represents a driver's license might contain the following sensitive identity information: photo, birth date, first and last name, and driver's license number.

Information cards let users manage their identity information. Users can view their information cards and the associated identity information. They can choose from the available cards for a given information exchange. And they can authorize the release of the identity information associated with a selected card.

Information cards are exposed by an Identity Selector.

Introduction to Identity Selectors

An Identity Selector is an application that lets users manage their identity information and their online relationships with Relying Parties and Identity Providers. A Relying Party (RP) is the web site, application or service that requires identity information to authenticate the user. The Identity Provider (IdP) is a third party that authenticates identity information and creates security tokens that the user can share with the Relying Party.

Identity Selectors allow users to access Web-based resources without having to manage a multitude of user names and passwords. Likewise, businesses no longer have to maintain a database of user identity information that can be inaccurate, out-of-date, and vulnerable to misuse, thus reducing risk and liability and enhancing agility.

Identity Selectors also give the user control over exactly what identity information is released to each Relying Party. And finally, Identity Selectors provide users with a consistent user interface and better user experience.

Windows CardSpace

Windows CardSpace, Microsoft's implementation of an Identity Selector, provides users with a consistent user interface for interacting with any Relying Party or Identity Provider. SiteMinder supports Windows CardSpace through a custom authentication scheme called Information Card Authentication Scheme (ICAS).

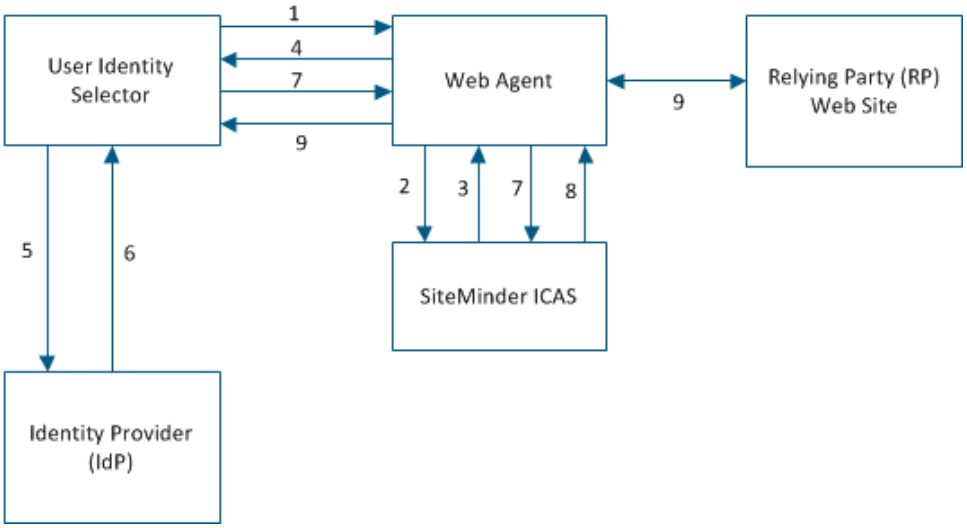
SiteMinder Information Card Authentication Scheme (ICAS)

SiteMinder Information Card Authentication Scheme (ICAS) is a SiteMinder authentication scheme that supports Windows CardSpace. Each instance of ICAS is configured as a custom authentication scheme in the Administrative UI and implemented like any other SiteMinder custom authentication scheme.

ICAS Overview

Authenticating a user with SiteMinder ICAS is a process that involves these components and steps:

- User
- Identity Selector
- Web Agent
- Relying Party (RP)
- Identity Provider (IdP)



1. A user wants to visit a SiteMinder-protected Web site or Relying Party (RP).
2. The Web agent intercepts the user's request and invokes ICAS.
3. ICAS sends the RP's policy requirements to the Web agent.
4. The Web agent instructs the user's browser to launch an Identity Selector on the user's computer and sends the RP's policy requirements.

5. The Identity Selector reads the policy requirements and highlights for the user those information cards that satisfy the requirements. The user selects one highlighted card. The Identity Selector collects the user's credentials and sends them to the Identity Provider (IdP) for authentication. The Identity Selector also sends the RP's policy requirements to the IdP and requests a token.

   Note: The user can select a card that contains optional claims not required by the RP.

6. The IdP authenticates the user and processes the policy requirements. It generates a token containing the required claims and sends it back to the Identity Selector.
7. The Identity Selector displays the claims, and the user approves release of the claims to the RP.
8. ICAS decrypts the token, verifies the token's authenticity and integrity, and associates the user's claims to a user's identity in the user database. SiteMinder then performs standard policy-based authorization and grants access to the user if authorized.
9. The user accesses the Web site.

ICAS Terms

The following terms are useful for understanding ICAS:

Identity Metasystem

An architecture that specifies how identity information can be shared by users, Relying Parties, and Identity Providers.

User

The person whose identity information is being shared. Sometimes, the user is called the subject.

Relying Party (RP)

The Web site that requests and consumes identity information.

Identity Provider (IdP)

A third party that authenticates identity information and shares the information with Relying Parties by creating security tokens. Credit card companies, banks, government agencies, employers, and insurance companies are all examples of Identity Providers.

Security Token Service (STS)

The technology used by Identity Providers to create security tokens. A Security Token Service:

- Authenticates users
- Creates security tokens that
  - Contain different subsets of identity information, depending on the requirements of the Relying Party and the restrictions of the user
  - Are of different types

    Note: SiteMinder supports SAML 1.0 and 1.1.
  - Are encrypted for security and signed for authenticity and integrity

Security Token

A cryptographically signed and encrypted set of claims.

Claim

An assertion of truth. Each token contains one or more claims about the user's identity. Examples of claims are first name, last name, email address, birth date, and so on. Claims can be made by the user or a third-party Identity Provider.

Information Card

A set of identity information. Information cards are comparable to the physical cards that we carry in our wallets. For example, an information card that corresponds to a driver's license might contain the following sensitive identity information: photo, birth date, first and last name, driver's license number, state, height, and sex.

Personal Card

An information card that contains claims that the user asserts about himself, but that are not corroborated by a third party. A personal card contains a Private Personal Identifier (PPID) that is generated when the card is created. Personal cards are appropriate for low-sensitivity identity information, such as an email address.

Note: Personal cards are also called self-issued cards.

Managed Card

An information card contains claims that the user asserts about himself and that are corroborated by a third party. A managed card contains a Private Personal Identifier (PPID) that is generated when the card is created and a pointer to the Identity Provider's STS. Managed cards are appropriate for sensitive identity information, such as a credit card number.

Identity Selector

An application that lets users manage their relationships with Relying Parties and Identity Providers and control how their identity information is shared and used. An identity selector:

- Enables sharing of information between parties that use multiple communication protocols and security technologies
- Provides a consistent user interface across multiple Identity Providers and Relying Parties using information cards
- Highlights those information cards that are available for each exchange of identity information
- Lets users view and consent to sharing identity information

Windows CardSpace

Microsoft's Identity Selector for the Windows operating system.

Information Card Authentication Scheme (ICAS)

Support for Windows Cardspace, Microsoft's Identity Selector, implemented in SiteMinder as a custom authentication scheme.

Private Personal Identifier (PPID)

Identifier generated by the Identity Selector when an information card is created.

ICAS Files

SiteMinder uses two files to configure each instance of ICAS: the fcc file and the properties file.

***filename*.fcc**

Specifies the authentication settings that SiteMinder requires and that can be customized for each instance of ICAS.

InfoCard.fcc

A sample fcc file that is shipped with the Web Agent kit

***filename*.properties**

Specifies how an instance of ICAS behaves.

InfoCard.properties

A sample properties file

Note: When configuring an instance of ICAS in the Administrative UI, the administrator specifies the path to the properties file.

ICAS Prerequisites

Before you can implement ICAS, the following conditions must be met:

Web Browser Configuration

One of the following web browsers must be used:

- Internet Explorer 7.0
- Firefox 2.x with CardSpace plug-in

Web Server Configuration

The Web Server must be configured for SSL communication. This configuration protects the fcc file.

Note: For more information, see the *Web Agent Configuration Guide*.

Web Agent Configuration

The InfoCard.fcc file that is shipped with the Web Agent kit must be customized for each instance of ICAS.

Note: For more information, see the *Web Agent Configuration Guide*.

Java Runtime Environment (JRE) Configuration

The Java Runtime Environment must be configured to decrypt security tokens that are encrypted with strong encryption algorithms.

Policy Server Configuration

Policy Server Configuration includes the following tasks:

- Configuring an ICAS properties file
- Configuring the certificate data store
- Configuring a user directory for ICAS
- Creating an instance of ICAS
- Configuring an active response that retrieves a claim value

Configure the Java Runtime Environment (JRE) for ICAS

Configure the Java Runtime Environment (JRE) by installing the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction policy files. These files are needed to decrypt security tokens that are encrypted with strong encryption algorithms.

Important! Back up the default policy files that are shipped with the JRE before installing the new policy files.

Follow these steps:

1. Download the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction policy files from http://www.oracle.com/technetwork/java/index.html.
2. Install them in the $NETE_JRE_ROOT\lib\security directory.
3. Add the following line to the $NETE_JRE_ROOT\lib\security\java.security file:

   ```
   security.provider.7=com.rsa.jsafe.provider.JsafeJCE
   ```

How to Configure the Policy Server for ICAS

Configuring the Policy Server for ICAS includes the following steps:

1. Configure an ICAS properties file.
2. Store claims for later use in active responses.
3. Configure the certificate data store for ICAS.
4. Configure a user directory for ICAS.
5. Create an instance of ICAS.
6. Configure an active response that retrieves a claim value.
7. Configure support for ICAM white list.

Configure an ICAS Properties File

An ICAS properties file specifies how an instance of ICAS behaves. When configuring an instance of ICAS in the Administrative UI, the administrator specifies the path to the associated properties file. To configure a new properties file, open a sample properties file with a text editor and edit the contents. Always save and rename the new properties file.

Note: Multiple instances of ICAS can share the same properties file.

Example: A properties file named InfoCard.properties contains the following properties and sample values:

fcc

Specifies the location of the Information Card fcc file.

Example: fcc=https://*web_server_home*/siteminderagent/forms/InfoCard.fcc

Note: To activate the Identity Selector, specify "https".

vppid_attribute

Specifies the VPPID attribute value. This attribute is required by the command UpdateUserStoreWithVPPID to update a user attribute.

Example: vppid_attribute=mail

vppid_claim

Specifies the claim to use to disambiguate the user.

Examples:

vppid_claim=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname

vppid_claim=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname

vppid_claim=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddres
s

alias

Specifies the key in the certificate data store that is used to retrieve the SSL certificate of the Relying Party.

Example: alias=rpssl

guestuser

If ICAS is configured in *anonymous* mode, specifies the guest user login.

Example: guestuser=guest

tokenPrim

Specifies the provider of the tokenPrim interface.

Example: tokenPrim=com.ca.sm.authscheme.infocard.higgins.TokenAdapter

storeclaims_attribute

Specifies the user attribute where the claims are stored when the command *StoreClaimsToUserStore* is executed.

Example: storeclaims_attribute=description

preprocessingchain

Defines a chain of commands to execute during user disambiguation.

Example: preprocessingchain=+vppidchain

Note: For more information, see Configure ICAS Command Chains (see page 342).

postprocessingchain

Defines a chain of commands to execute during user authentication.

Example:
postprocessingchain=+vppidchain;com.ca.sm.icas.command.StoreClaimsToContext

Note: For more information, see Configure ICAS Command Chains (see page 342).

errorprocessingchain

Defines a chain of commands to execute during error processing.

Note: For more information, see Configure ICAS Command Chains (see page 342).

whiteListLocation

(Optional) Defines the location of the XML file that contains the White List specified by Federal Identity, Credentialing, and Access Management (ICAM).

Example: whiteListLocation=C:\\Program Files\\ca\\siteminder\\config\\WhiteList.xml

Note: For more information about ICAM White List, see Configure Support for the ICAM White List (see page 351).

Configure ICAS Command Chains

Command chains are defined in the ICAS properties file. A chain is a a set of commands that executes in an orderly manner.

Named chain of commands include:

- preprocessing
- postprocessing
- errorprocessing

These named chains are present by default.

Consider the following:

- If the specific named chains are found in the ICAS properties file, then ICAS uses these named chains.
- You can define your own chains and place them in the ICAS properties file and refer them in the named chains.
- If any customized chain is not referred in any of these named command chains, that chain execution is ignored.

According to your requirements, configure the following class files:

- Key class files to configure ICAS (see page 343)
- Class files to check the values of claims (see page 344)
- Class files to debug (see page 345)

Key Class Files to Configure ICAS

The following key class files are used to configure ICAS:

com.ca.sm.icas.StoreClaimsToSessionStore

Stores extracted claims to the session store of Policy Server.

com.ca.sm.icas.command.ExecuteClaimChain

Executes a set of commands over a claim. For each claim name, which forms the last part of the namespace, ICAS looks for a chain defined as chainID.claimID in the properties file. If the chain definition is found, then ICAS executes it and passes the context with the specific claim.

Example: For executeClaimCommand1=com.ca.sm.icas.command.ExecuteClaimChain, define the email claim as follows:

executeClaimCommand1.emailaddress=com.ca.sm.icas.command.DumpClaimsCommand;

Processes the DumpClaimsCommand if a token is found with an emailaddress claim.

com.ca.sm.icas.command.HashVPPID

Generates a hash of the VPPID claim attribute.

com.ca.sm.icas.command.SetVPPIDAsAnonymous

Sets the VPPID claim attribute value to anonymous. This class file is used for logging in with information cards that are not linked to any user accounts (anonymous access).

com.ca.sm.icas.command.SetVPPIDFromToken

Sets the VPPID claim attribute value from the information card token to the context object.

com.ca.sm.icas.command.StopChain

Returns the value true, which stops the execution of the currently running chain.

com.ca.sm.icas.command.StoreClaimsToContext

Reads the claims from the information card token and store them to the application specific context.

com.ca.sm.icas.command.StoreClaimsToUserStore

Stores the claims read from the information card token to a specified user attribute (in the ICAS properties file) in the user directory.

com.ca.sm.icas.command.TransformScriptInClaim

Escapes the script tags (<, >) in the claims using HTML entities. For example, <td> becomes <td;&gt

com.ca.sm.icas.command.TransformSqlInClaim

Escapes the characters in a claim value so that it can be passed to an SQL query.

com.ca.sm.icas.command.TranslateErrorCode

Translates the exception string found within context. This translated string is stored back into the context.userText that is available in the FCC.

com.ca.sm.icas.command.TranslateErrorCodeFromParms

Translates the exception string found within the ICAS properties file. This translated string is stored back into the context.userText that is available in the FCC.

com.ca.sm.icas.command.UpdateUserStoreWithVPPID

Stores the claims read from the information card token to a specified user attribute (available in the ICAS properties file) from the user directory.

Class Files to Check the Values of Claims

The following class files are used to check the values of claims:

com.ca.sm.icas.StoreClaimsToSessionStore

Stores extracted claims to the session store of Policy Server.

com.ca.sm.icas.command.ErrorIfEmptyClaim

Generates an exception when the claim is empty.

com.ca.sm.icas.command.ErrorIfMultiValueClaim

Generates an exception when the given claim has multiple values.

com.ca.sm.icas.command.ErrorIfNotEmptyClaim

Generates an exception when the given claim is not empty.

com.ca.sm.icas.command.ErrorIfNotMultiValueClaim

Generates an exception when the given claim is not multivalued.

com.ca.sm.icas.command.ErrorIfNotNullClaim

Generates an exception when the given claim is not null.

com.ca.sm.icas.command.ErrorIfNotSingleValueClaim

Generates an exception when the given claim is not single valued.

com.ca.sm.icas.command.ErrorIfNullClaim

Generates an exception when the given claim is null.

com.ca.sm.icas.command.ErrorIfSingleValueClaim

Generates an exception when the given claim is single valued.

Class Files to Debug

The following class files are used for debugging purpose:

com.ca.sm.icas.command.debug.DumpChainID

Displays the chain ID on the console.

Note: To view the output, start the Policy Server in the console mode.

com.ca.sm.icas.command.debug.DumpClaim

Displays the VPPID claim details (name and value) on the console.

Note: To view the output, start the Policy Server in the console mode.

com.ca.sm.icas.command.debug.DumpClaims

Displays the details of all the claims in the token on the console

Note: To view the output, start the Policy Server in the console mode.

com.ca.sm.icas.command.debug.DumpContext

Displays the details of the context object on the console.

Note: To view the output, start the Policy Server in the console mode.

com.ca.sm.icas.command.debug.DumpParameters

Displays the details of the parameters specified in the ICAS properties file on the console.

Note: To view the output, start the Policy Server in the console mode.

com.ca.sm.icas.command.debug.DumpSystemVars

Displays the details of the system environment on the console.

Note: To view the output, start the Policy Server in the console mode.

com.ca.sm.icas.command.debug.DumpThreadStack

Displays the stack trace of the executing thread on the console.

Note: To view the output, start the Policy Server in the console mode.

com.ca.sm.icas.command.debug.GenerateClaim_claims_AsHTML

Generates a new claim "_claims_" that has an HTML table representing the known claims. You can use this claim in an active response to create a cookie or header to show the retrieved claims to a user.

com.ca.sm.icas.command.debug.GenerateClaim_parameters_AsHTML

Generates an HTML table representing the parameters specified in the ICAS properties file. You can use this HTML table in an active response to create a cookie or a header to show the retrieved claims to a user.

com.ca.sm.icas.command.debug.LogClaim

Writes the claim details (name and value) in the chain context to the Policy Server trace logs.

com.ca.sm.icas.command.debug.LogClaims

Writes the details of all the claims in the token to the Policy Server trace log.

com.ca.sm.icas.command.debug.LogDecryptedClaims

Logs the decrypted XML token to the Policy Server trace log.

com.ca.sm.icas.command.debug.LogEncryptedClaims

Logs the encrypted token obtained from the IDP to the Policy Server trace logs.

com.ca.sm.icas.command.debug.START

Displays a START message on the console.

Note: To view the output, start the Policy Server in the console mode.

com.ca.sm.icas.command.debug.STOP

Displays a STOP message on the console.

Note: To view the output, start the Policy Server in the console mode.

com.ca.sm.icas.command.debug.ThrowException

Generates an exception object that includes the claim value as a part of the message.

Store Claims for Later Use in Active Responses

You can store claims for later use in active responses. To store claims for later use, add the following property to the properties file:

postprocessingchain

Defines the chain of commands to execute during user authentication. This phase includes any claim transformation and storage commands.

Example:
postprocessingchain=com.ca.sm.authscheme.infocard.command.StoreClaimsToContext

Configure the Certificate Data Store for ICAS

Consider the following:

- The Relying Party must use SSL to protect the fcc file.
- The Relying Party must export the SSL certificate associated with the website to a pfx file.
- A SiteMinder administrator configures the certificate data store for ICAS by importing the SSL certificate from the Relying Party website.
- The imported certificate is associated with an alias, which is stored in the fcc file. The private key of the certificate is used to decrypt the security token and verify the digital signature.

To configure the certificate data store for ICAS

1. Use the Web Server Certificate Wizard to export the SSL certificate from an IIS web server to a pfx file.

   Note: For more information, see the Microsoft documentation. The password you provide when exporting the SSL certificate to the pfx file is used later by SiteMinder when importing the SSL certificate from the pfx file.

2. Use the Administrative UI to import the SSL certificate into the certificate data store.

Configure a User Directory for ICAS

Authentication of the user depends on finding a match between one of the claims presented to ICAS and a user attribute in the user database. During token disassembly, the specified claim value is used as a lookup value in the user directory. Therefore, the user directory must be configured so that the LDAP lookup string or SQL query scheme specifies the user attribute that corresponds to the specified claim. The following examples show how to configure an LDAP lookup string and SQL query scheme for an email address.

LDAP Example

LDAP User DN Lookup group box

Start

(mail=

End

)

SQL Example

SQL Queries group box

Get User/Group Info

SELECT EmailAddress, 'User' FROM SmUser WHERE EmailAddress = '%s' UNION SELECT Name, 'Group' FROM SmGroup WHERE Name = '%s'

Authenticate User

SELECT EmailAddress FROM SmUser WHERE EmailAddress = '%s' AND Password = '%s'

Create an Instance of ICAS

You can create an instance of ICAS by specifying a custom authentication scheme in the Administrative UI.

Limit: Each policy store can support up to ten instances of ICAS.

To create an instance of ICAS

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.
4. Select Create a new object of type Authentication Scheme, and click OK.

   The Create Authentication Scheme: *Name* page appears.
5. Type the authentication scheme's name and description.

6. Select Custom Template from the list of Authentication Scheme Types.

   The Scheme Setup group fields appear.

7. Enter a value in the Protection Level field.

8. Clear the Password Policies enabled for this Authentication Scheme Types.

   Note: ICAS does not support Password Policies.

9. Enter the values for the following fields in Scheme Setup:

   Library

   smjavaapi

   Note: The custom authentication scheme uses the Java Authentication API.

   Secret and Confirm Secret

   Leave these fields blank.

   Note: The custom authentication scheme does not use the shared secret.

   Parameter

   Type the following two parameters in the Parameter field and separate them by a space:

   com.ca.sm.icas.SmAuthInfoCard

   This is the fully qualified name of the class that implements the SmAuthScheme interface.

   policy_server_home\config\icas\InfoCard.properties

   This is the location of the properties file.

   Example:

   com.ca.sm.icas.SmAuthInfoCard
   policy_server_home\config\icas\InfoCard.properties

10. (Optional) Select the Store Auth Scheme Context to Session Store option in the Scheme Setup. This option specifies that the authentication context data is persisted.

11. Click Submit.

    The Create Authentication Scheme task is submitted for processing.

Configure an Active Response that Retrieves a Claim Value

You can use the custom class com.ca.sm.icas.GetClaimValue to configure an active response that retrieves a claim value after authentication is complete.

Note: Storing and retrieving claim values requires a session store. For more information about session stores, see the *Policy Server Administration Guide*.

To configure an active response that retrieves a claim value

1. Click Policies, Domain.
2. Click Responses.

   The Responses page appears.
3. Click Create Response.

   The Create Response: Select Domain page appears.
4. Select a domain, and click Next.

   The Create Response: Define Response page appears.
5. Type the name and a description of the response.
6. Specify Web Agent as the Agent Type.
7. Click Create Response Attribute in Attribute List.

   The Create Response Attribute: *Name* page appears.
8. Select WebAgent-HTTP-Header-Variable or WebAgent-HTTP-Cookie-Variable from the list of attributes in Attribute Type.
9. Select Active Response as the Attribute Kind in Attribute Setup.
10. Type the following values in the Attribute Fields.

    Cookie or Variable Name

    Specifies the name of the claim.

    Example: emailaddress

    Library Name

    Specifies the name of the library.

    Value: smjavaapi

    Function Name

    Specifies the name of the function.

    Value: JavaActiveExpression

Parameters

Specifies the custom ICAS command and the location of the file, claims.xsd, that defines standard claim types according to the Information Card model.

Example: com.ca.sm.authscheme.infocard.GetClaimValue http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress

11. Click OK.

    The response that retrieves a claim value is created.

Configure Support for the ICAM White List

The ICAS implementation supports the Federal Identity, Credentialing, and Access Management Identity Metasystem Interoperability 1.0 Profile (ICAM IMI 1.0 Profile) specifications, which is based on White List of issuers and LOA authorization.

Note: For more information about White List and LOA support, see *ICAM IMI 1.0 Profile Release Candidate*.

To enable White List processing, update the following parameters in the *Infocard.properties* file.

postprocessingchain

Defines the chain of commands to execute during user authentication.

Example: postprocessingchain=com.ca.sm.icas.whiteListChecker.whiteListChecker

whitelistlocation

Defines the location of the XML file that contains the White List specified by ICAM.

Example: whitelistlocation=C:\\Program Files\\ca\\siteminder\\config\\WhiteList.xml

The ICAS implementation supports the verification of three Level of Assurances (LOAs)

- LOA1
- LOA2
- LOA3

White List processing is mandatory to verify LOA1, LOA2, and LOA3. Append the following details to the postprocessingchain parameter to support LOA processing:

- For LOA1 processing

  postprocessingchain=com.ca.sm.icas.whiteListChecker.whiteListChecker;
  com.ca.sm.icas.whiteListChecker.ErrorIfLOA1ClaimMissing

- For LOA2 processing

  postprocessingchain=com.ca.sm.icas.whiteListChecker.whiteListChecker;
  com.ca.sm.icas.whiteListChecker.ErrorIfLOA2ClaimMissing

- For LOA3 processing

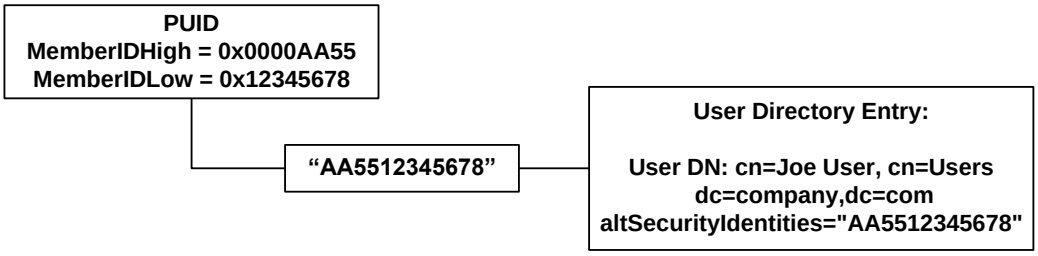
  postprocessingchain=com.ca.sm.icas.whiteListChecker.whiteListChecker;
  com.ca.sm.icas.whiteListChecker.ErrorIfLOA3ClaimMissing

MS Passport Authentication Schemes

Microsoft® .NET Passport is an online service that provides common Internet authentication across participating Web sites. Using a .NET Passport account, a user can move among participating sites without the need to authenticate with each of them. These sites become participating .NET Passport sites by implementing support for the .NET Passport authentication service through the .NET Passport single sign-in (SSI). This implementation includes a link to login through Passport using a common interface that supports co-branding. Users that are already logged into Passport will automatically be authenticated with the site, by means of a browser redirect to acquire Passport identity, and creation of site cookies containing the Passport identity. However, Passport does not authorize or deny a specific user's access to participating sites or resources.

The Passport identity does not contain any data for authorization. The only field that can be used to derive identity and subsequently permissions is the Passport Unique ID (PUID). SiteMinder uses this PUID to map to a local identity that is used to personalize the site and to authorize access to resources protected by SiteMinder policies.

The PUID is converted to a string and is mapped to a SiteMinder identity through a user directory attribute. In the following diagram, the user DN for "Joe User" is mapped to a Passport PUID through the directory attribute "altSecurityIdentities".



Since Passport authentication in SiteMinder is based on mapping the Passport identity to a SiteMinder user, registration is a required component for using Passport for authentication or personalization.

Registration is controlled through a registration URL that is configured in the authentication scheme. The configuration parameter is ?registrationurl=? followed by a registration page or the keyword "FORM=" and SiteMinder form URL. This model provides two methods for registration of Passport users.

The first method uses a site-provided web application to link the Passport identity with a SiteMinder user account through the attribute configured in the authentication scheme. This requires the site to develop the web pages to accept registration data and to provide a service for setting the user directory attribute. SiteMinder can be used to provide the interfaces to the user directory and the secure tunnel behind the firewall using a tunnel service.

The second method of registration leverages the SiteMinder FCC and the forms authentication model. When the registrationurl includes the keyword FORM=, the subsequent URL is treated as a forms redirect. SiteMinder provides the passport.fcc file to use as a template for developing a Passport registration page using forms.

SiteMinder r12.5 provides a Passport Authentication Scheme template. The library name is smauthmspp. This authentication scheme can be used on both Windows and UNIX Policy Servers.

Passport Authentication Support in the Policy Server

The Policy Server and SiteMinder Web Agents support the following implementations of Passport authentication:

- Passport authentication established through a common registration URL
- Passport authentication using SiteMinder HTML forms for registration
- Anonymous logins when no Passport identity exists

For information about deploying Passport authentication in your enterprise, see: http://www.microsoft.com

Set Protection Levels for Passport Authentication

Since the process of establishing a Passport identity does not include any authorization for access to participating sites or resources, the Passport authentication scheme should be assigned a relatively low protection level. We recommend using Passport authentication for personalization, and enforcing an authentication scheme with a higher protection level for sensitive resources. For example, Passport users could be authenticated, and their identities established using a SiteMinder protection level of 1. When the users request sensitive financial information, they might be forced to reauthenticate using an HTML forms authentication scheme with a protection level of 10.

Passport Authentication Prerequisites

Ensure the following prerequisites are met before configuring a Passport authentication scheme:

- Passport SDK version 1.4 for WinNT, or version 2.1 for Win2000 is configured
- Passport Partner site is configured
- SiteMinder 5.x QMR1 or QMR v2 Web Agents, when available, are installed and running on Internet Information Server (IIS) on Windows systems
- Policy Server version 5.5 must be installed

Configure a Microsoft Passport Authentication Scheme

You can use a Microsoft Passport authentication scheme to authenticate users across participating .NET Passport Web sites.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.

3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.

4. Click OK

   The Create Authentication Scheme page appears.

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

5. Enter a name and protection level.
6. Select Microsoft Passport Template from the Authentication Scheme Type list.

   Scheme-specific settings appear.

7. Enter configuration information.
8. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

Map Search Specifications for Passport Authentication

You configure a search specification so SiteMinder can recognize the identity established through Microsoft® .NET Passport. The search specification lets SiteMinder locate the Passport Unique ID (PUID) in a user directory.

You can configure a search specification for the following namespaces:

- LDAP
- ODBC
- Custom

Note: SiteMinder does not support Passport authentication using a WinNT user directory.

To map a search specification for a namespace

1. Open the MS Passport Authentication scheme that you want.

2. Enter a mapping in the respective directory field in the Scheme Setup group box.

   Mappings should be configured as follows:

   - LDAP:

     ```
     altSecurityIdentities=Kerberos:%s@company.local
     ```

   - ODBC:

     ```
     PassportUID=PUID:%s@company.local
     ```

   - Custom:

     ```
     PassportUID=PUID:%s@company.local
     ```

   Note: The authentication scheme can be configured to provide a search specification for one or more directory types.

3. Click Submit.

   The search specification is saved.

RADIUS CHAP/PAP Authentication Schemes

The RADIUS protocol can be used to implement CHAP or PAP based authentication.

PAP Overview

The Password Authentication Protocol (PAP) provides a simple method for a user to authenticate using a 2-way handshake. PAP only executes this process during the initial link to the authenticating server. With this scheme, an Id/Password pair is repeatedly sent by the user's machine to the authenticating server until authentication is acknowledged or the connection is terminated.

This authentication method is most appropriately used where a plain text password must be available to simulate a login at a remote host. In such use, this method provides a similar level of security to the usual user login at the remote host.

CHAP Overview

CHAP (Challenge-Handshake Authentication Protocol) is a more secure authentication scheme than PAP. In a CHAP scheme, the following takes place in order to establish a user's identity:

1. After the link between the user's machine and the authenticating server is made, the server sends a challenge message to the connection requester. The requester responds with a value obtained by using a one-way hash function.
2. The server checks the response by comparing it against its own calculation of the expected hash value.
3. If the values match, the authentication is acknowledged; otherwise the connection is usually terminated.

At any time, the server can request the connected party to send a new challenge message. Because CHAP identifiers are changed frequently and because authentication can be requested by the server at any time, CHAP provides more security than PAP.

RADIUS CHAP/PAP Scheme Overview

The RADIUS CHAP/PAP scheme authenticates users by computing the digest of a user's password, and then comparing it to the CHAP password in the RADIUS packet. The digest consists of the user's hashed password, which is calculated using a directory attribute specified during the configuration of the RADIUS CHAP/PAP authentication scheme.

RADIUS CHAP/PAP Scheme Prerequisites

Be sure that the following prerequisites are met before configuring a RADIUS CHAP/PAP authentication scheme:

- The field in the user directory specified for the clear text password contains a value.
- The Policy Server is not operating in FIPS–only mode. If the Policy Server is operating in FIPS–only mode, a RADIUS CHAP/PAP authentication scheme is not supported.

Configure a RADIUS CHAP/PAP Authentication Scheme

You can use a RADIUS CHAP/PAP authentication scheme when you are using the RADIUS protocol.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and protection level.
6. Select RADIUS CHAP/PAP Template from the Authentication Scheme Type list.

   Scheme-specific settings appears.
7. Specify the clear text password in Scheme Setup section.
8. Click Submit.

   The authentication scheme is saved and may be assigned to a realm.

RADIUS Server Authentication Schemes

SiteMinder supports the RADIUS protocol by using the Policy Server as the RADIUS server and an NAS client as the RADIUS client. RADIUS Agents allow the Policy Server to communicate with the NAS client devices. In the RADIUS Server authentication scheme the Policy Server acts as a RADIUS server attached to the SiteMinder protected network.

This scheme accepts user name and password as credentials. Multiple instances of this scheme can be defined. This scheme does not interpret RADIUS attributes that may be returned by the RADIUS server in the authentication response.

For more information on RADIUS Server authentication with SiteMinder, see the material on using the Policy Server as a RADIUS server in the *Policy Server Administration* guide.

RADIUS Server Scheme Prerequisites

Be sure that the following prerequisites are met before configuring a RADIUS Server authentication scheme:

- The RADIUS server is on a network accessible by the Policy Server.
- The Policy Server is not operating in FIPS–only mode. If the Policy Server is operating in FIPS–only mode, a RADIUS Server authentication scheme is not supported.

Configure a RADIUS Server Authentication Scheme

You can use a RADIUS Server authentication scheme when you are using the Policy Server as the RADIUS Server and a NAS client as a RADIUS client.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and a protection level.
6. Select RADIUS Server Template from the Authentication Scheme Type list.

   Scheme-specific settings appear.
7. Enter the RADIUS server IP address, port number, and shared secret in Scheme Setup.
8. Click Submit.

   The authentication scheme is saved and may be assigned to a realm.

SafeWord Server Authentication Schemes

This Authentication Scheme authenticates users against a SafeWord Server, including users who are logging in via the SafeWord hardware tokens. You can define multiple instances of this scheme. The exact configuration parameters of the SafeWord Server are specified by the SafeWord configuration file.

Note: SafeWord authentication schemes, smauthenigma and smauthenigmahtml, are only supported on Windows and Solaris platforms after the 6.0 SP3 CR03 release.

SafeWord Server Scheme Prerequisites

Ensure the following prerequisites are met before configuring a SafeWord authentication scheme:

- The SafeWord Server is installed on a network accessible by the Policy Server.
- The exact location of the SafeWord Server is specified in the SafeWord configuration file.

Configure a SafeWord Server Authentication Scheme

You can use a SafeWord Server authentication scheme when you need to authenticate users against a SafeWord Server, including users who are logging in via SafeWord hardware tokens.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

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

5. Enter a name and protection level.
6. Select SafeWord Template from the Authentication Scheme Type list.

   Scheme-specific fields and controls appear.
7. Enter the location of the SafeWord server configuration file.
8. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

SafeWord Server and HTML Forms Authentication Schemes

This Authentication Scheme authenticates users against a SafeWord Server in conjunction with a custom HTML form, including users who are logging in via the SafeWord hardware tokens. You can define multiple instances of this scheme. The exact configuration parameters of the SafeWord Server are specified by the SafeWord configuration file.

SafeWord Server and HTML Forms Scheme Prerequisites

Ensure the following prerequisites are met before configuring a SafeWord Server and HTML Forms authentication scheme:

- The SafeWord Server is installed on a network accessible by the Policy Server.
- The exact location of the SafeWord Server is specified in the SafeWord configuration file.
- A customized .fcc file resides on a Web Agent server in the cookie domain in which you want to implement HTML Forms authentication. CA provides sample .fcc files under the Samples/Forms subdirectory, where you installed your Web Agent.
- A customized .unauth file resides on the Web Agent server

  Note: The .unauth file is not required if the .fcc file uses smerrorpage directive.
- A directory connection exists between the Policy Server and the user directory.
- The default HTML forms library is installed. The HTML forms library handles HTML Forms authentication processing:
  - SmAuthHTML.dll on Windows
  - smauthhtml.so on Solaris

  These files are installed automatically when you configure a Web Agent.
- (Sun Java Systems) If you are using a Sun Java Systems web server, increase the value of the StackSize parameter in the magnus.conf file to a value greater than 131072. Failing to change the value causes the web server to dump its core and restart each time SiteMinder makes an authentication request using forms.

More information:

SiteMinder FCC Files (see page 320)
User Directories (see page 157)

Configure a SafeWord Server and HTML Forms Authentication Scheme

You can use a SafeWord Server and HTML Forms authentication scheme when you need to authenticate users against a SafeWord Server and a custom HTML form, including users who are logging in via SafeWord hardware tokens.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and protection level.
6. Select SafeWord HTML Form Template from the Authentication Scheme Type list.

   Scheme-specific fields and controls appear.
7. Enter the server name, the location of the SafeWord configuration file, and server and target information.
8. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

SecurID Authentication Schemes

The RSA Ace/SecureID authentication schemes authenticate users logging in with ACE credentials, which include the user names, PINs, and TOKENCODEs. ACE usernames and passwords reside in the ACE/Server user store and can be changed by ACE/Server administrator. Single-use TOKENCODEs are generated by SecureID tokens.

SiteMinder supports the following SecurID authentication schemes:

SecurID Authentication

This Authentication Scheme authenticates users who are logging in via RSA SecureID hardware tokens. The scheme accepts a user name and password. The password is the user's token PIN followed by the dynamic code.

Note: The SecurId Template authentication scheme will fail to authenticate a user if the user does not use the user directory attribute named 'uid' to map to the userid of the user in the Ace Server. Use the SecurID HTML Forms Template when mapping LDAP directory attributes other than uid to the userid of the user in the Ace Server.

SecurID Authentication with HTML Forms Support

SiteMinder supports a scheme for SecurID authentication that includes HTML forms support. The HTML forms provide a method for users to identify themselves and reactivate their accounts after they have been disabled because they entered their PIN or token information incorrectly.

The RSA ACE/SecurID scheme authenticates users who are not allowed to change their ACE PIN. However, users allowed or required to change their PIN are authenticated through the RSA ACE/SecurID scheme with HTML form. Both schemes are based on ACE/Server - Ace/Agent model and require RSA/SecurID (tokens) and RSA/ACE (server software) products.

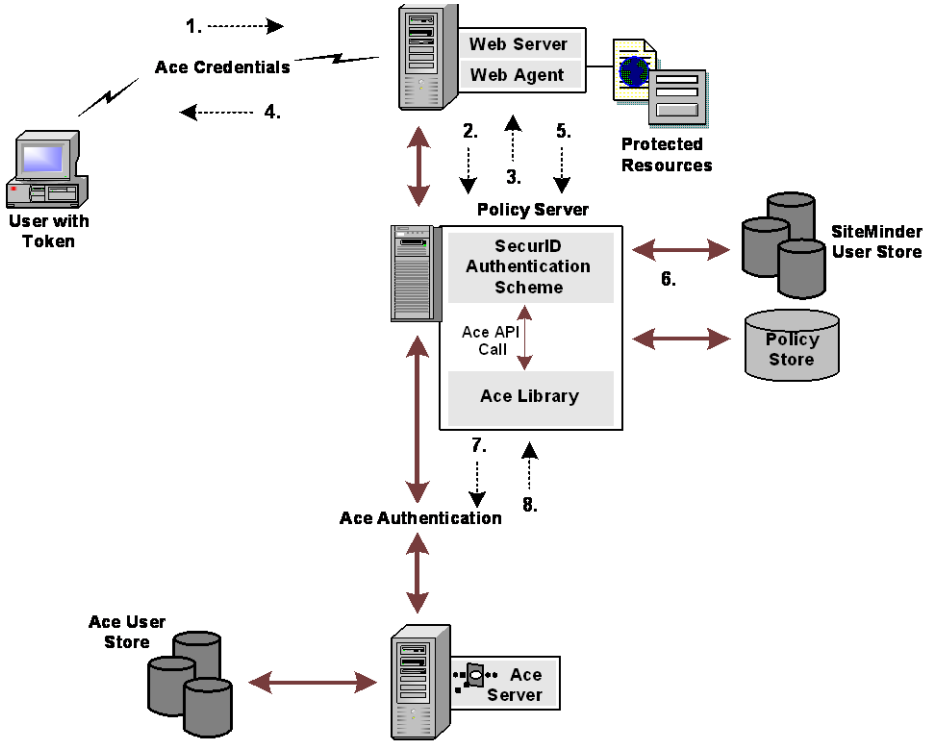
The RSA ACE/Server works with RSA SecurID tokens to authenticate users' identities through valid RSA Ace/Agents. Further, RSA supports Web Agents and custom Agents. SiteMinder SecurID authentication schemes act as custom ACE/Agents and use ACE/SDK and libraries.

The ACE/Server, however, has its own policy in handling users' credentials and stores this information in the ACE user store. The policy includes several requirements and limitations for each user, which are defined by the ACE administrator that can be changed at any time. The administrator configures users to authenticate by either PIN or TOKENCODE. If users are configured to authenticate by PIN, the system does not require a TOKENCODE. If users are configured to authenticate with TOKENCODE, both the TOKENCODE and PIN are required. In addition, users could be required to set a new PIN while trying to authenticate. Under this "new PIN mode", the old and new PINs are required.

To successfully authenticate users, SiteMinder SecureID authentication schemes require mapping between the ACE and SiteMinder users. This mapping is represented as an authentication scheme object attribute in the SiteMinder policy store.

In 5.5 SP1, the enhancement to the RSA ACE/SecurID Scheme with HTML form affects the "new PIN mode".

The following figure shows how the RSA ACE/Server interacts with SiteMinder:



1. A user requests a resource such as a Web page.
2. The SiteMinder Web Agent determines if the resource is protected.
3. The Policy Server indicates the requested resource is protected by the RSA ACE/SecurID Authentication Scheme with HTML form.
4. The Web Agent prompts the user for credentials.
5. The user enters credentials and the Agent sends them to the Policy Server.
6. The Policy Server performs user disambiguation and maps the SiteMinder user name to RSA/ACE user name. At this point, the Policy Server does not apply SiteMinder password policies.
7. The Policy Server forwards the authentication request to the ACE/Server by making a sequence of ACE API calls. It also send the user's credentials to the ACE/Server.

8. The ACE/Server indicates that the user is required to change the PIN and then returns control back to the Policy Server.
9. The Policy Server returns control to the Web Agent and then the Agent redirects the user to a CGI or JSP.
10. The CGI or JSP generates the applicable HTML form, presents it to the user, and then prompts for a user name, old PIN, new PIN, and new PIN confirmation.
11. The Web Agent sends a new set of credentials to the Policy Server.
12. The Policy Server makes a new sequence of API calls to the ACE/Server.
13. If new the PIN is accepted, then the ACE API call returns success. From here, the user is asked to login with the new PIN and the authentication process is complete.
14. If new PIN is rejected, Step 10 to Step 12 are repeated.

The PIN can be rejected if it:

- Is too long
- Is too short
- Contains alphabetic characters

In the process illustrated in the previous figure authentication includes presenting back-end PIN policy-related messages on the HTML form. The PIN policy validation is done by the SecurID Authentication scheme before sending a new PIN to the ACE/Server. The ACE SDK has a set of functions that can retrieve the following PIN attributes:

- Maximum length
- Minimum length
- Alphabetic and numeric or only numeric characters

Based on this information, the PIN is validated and the appropriate error messages are constructed and presented to the user, and the validation is done in the following order:

- The PIN is not too long
- The PIN is not too short
- The PIN does not have invalid characters

In the 5.5 SP1, only the relevant portion of the policy is made available to users in these new error messages. The following shows an example of how these new messages might appear:

If a sample user, Joe, has a PIN that is 5 to 8 digits long but enters a new PIN as "poem", then he would see the following error message:

> Your new PIN is too short. PIN must contain at least 5 character(s).

If the next PIN he entered was "novel", he would get:

Your new PIN may not have alphabetic characters.

If the next PIN he entered was "123412341234," he would see.

Your new PIN is too long. PIN must contain 8 or fewer character(s).

It is possible that, despite the successful PIN validation by the SecureID Authentication scheme, the ACE/Server will reject a new PIN. In this case, the user is asked for a new set of credentials, but no reason is given. Further, in the enhanced SecurID Authentication scheme, the user is automatically granted access to the target Web page after a successful PIN change.

SecurID Scheme Prerequisites

Be sure that the following prerequisites are met before configuring a SecureID authentication scheme:

- On Windows Policy Servers, the RSA ACE/Client software is installed on the same system as the Policy Server. For information about supported RSA ACE/Client versions, see the Platform Support Matrix on the Support site.
- If the following are true, be sure to configure the ACE paths to point to the location of the securid file:
  - The ACE environment is using ACE Client 7.0 or later.
  - The ACE environment is not using a Node Secret.
  - One of the following:
    - ACE is protecting another application, which SiteMinder does not protect.
    - ACE is protecting another non–SiteMinder product.

  Configuring the ACE paths prevents the authentication request that the Policy Server sends to the ACE Server from failing.

Note: The SM_ACE_FAILOVER_ATTEMPTS environment variable, which is used to set the failover attempts to the ACE server, has been removed.

Configure a SecurID Authentication Scheme

You can use a SecurID authentication scheme to authenticate users logging in with ACE credentials.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and a protection level.
6. Select SecurID Template from the Authentication Scheme Type list.

   Scheme-specific fields and controls appear.
7. Enter the ACE User ID attribute.
8. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

SecurID with HTML Forms Support Scheme Prerequisites

Ensure the following prerequisites are met before configuring a SecureID with HTML authentication scheme:

- The RSA ACE/Client software is installed on the same machine as the Policy Server.
- For Policy Servers on Windows, the ACE configuration information file (sdconf.rec) resides in the winnt\system32 directory. The VAR_ACE and USR_ACE variables are pointing to the appropriate ACE agent data and prog directories respectively.
- For Policy Servers on UNIX, the sdconf.rec file resides in the *<policy server installation dir>*/lib directory. In addition, the VAR_ACE and USR_ACE variables are pointing to the *<policy server installation dir>*/lib, allowing the Policy Server to use SiteMinder API libraries and not the ACE agent.
- A customized .fcc file resides on a Web Agent server in the cookie domain in which you want to implement HTML Forms authentication. CA provides sample .fcc files under the Samples/Forms subdirectory, where you installed your Web Agent.

- A customized .unauth file resides on the Web Agent server.

  Note: The .unauth file is not required if the .fcc file uses the smerrorpage directive.

- A directory connection exists between the Policy Server and the user directory.
- The default HTML forms library is installed. This library handles HTML Forms authentication processing:
  - SmAuthHTML.dll on Windows
  - smauthhtml.so on Solaris

  These files are installed automatically when you configure a Web Agent.

More information:

SiteMinder FCC Files (see page 320)
User Directories (see page 157)

Configure a SecurID and HTML Forms Authentication Scheme

You can use a SecurID and HTML forms authentication scheme to use a custom HTML form to authenticate users logging in with ACE credentials.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and a protection level.
6. Select SecurID HTML Form Template from the Authentication Scheme Type list.

   Scheme-specific settings open.

7. Enter server, target, and ACE attribute information.
8. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

Forms Support for Re-activating and Verifying SecurID Users

If you protect a realm with the SecurID and HTML Forms scheme, users who are suspended due to improper logins can attempt to activate their accounts using a number of customizable HTML forms provided with SiteMinder. You can modify the layout and wording of these forms, but you must not modify the tags that gather user information.

The forms provided with SiteMinder include the following:

PWLogin.template

This form is where users enters a username and passcode to login.

PWNextToken.template

This template requests multiple tokencodes to confirm that the user is in possession of a working SecurID token.

Forms Support for Activating New User Accounts

The following forms are used by SiteMinder when an administer creates a new user account, and that user logs in. Through the forms, a user creates a PIN, or has SiteMinder generate a random PIN.

PWSystemPIN.template

For new users, or users whose accounts have been suspended (due to too many invalid login attempts), this template prompts the user to acquire a new PIN. This template accepts the user's original username and passcode, but instead of granting access to a protected resource, it redirects the user to another form where the user can receive or create a new PIN.

PWNewPINSelect.template

This template allows a user to indicate if the system should generate a new PIN, or if the user wants to enter a new PIN.

PWUserPIN.template

This template allows a user to enter a new PIN. It requires that the user provide a valid username and passcode along with the new PIN. In this template, $USRMSG$ is replaced with instructions for creating a new PIN number. For example:

PINs must be between 4 and 8 characters in length.

PWPINAccept.template

This template indicates that a new system-generated PIN has been created. In this template, $USRMSG$ is replaced by the system generated PIN.

When a user clicks Continue, the user is immediately prompted to log in using the new PIN.

X.509 Client Certificate Authentication Schemes

X.509 client certificates provide cryptographic evidence of a user's identity. A user certificate, supplied by a certificate vendor, is unique, and can be used to identify the user who attempts to access a protected resource.

A client certificate contains the following information:

- Subject name
- Issuer name
- Serial number
- Version
- Validity period
- Signature (algorithm ID and parameters)
- Subject public key (and associated algorithm ID)
- X.509 v3 extensions (optional)

SiteMinder uses the X.509 Client Certificate authentication schemes to implement certificate authentication. To use X.509 client certificate authentication, your environment must be able to handle SSL communication. This means that the client browser, the web server and any user certificates must be configured to accept and perform certificate authentication. These tasks are outside the scope of SiteMinder configuration.

After the necessary SSL components are set up properly, you can configure a SiteMinder X.509 authentication scheme. SiteMinder configuration tasks require that you do the following:

- Select an advanced SSL authentication scheme when running the SiteMinder Web Agent Configuration Wizard.

  For information about the Web Agent Configuration Wizard, see the *SiteMinder Web Agent Installation Guide*.

- Configure one of the X.509 authentication schemes using the Administrative UI, which is detailed in this guide.

The SiteMinder X.509 Client Certificate authentication schemes perform the following tasks:

- Collects the client certificate information.
- Identifies a user in a directory based on the information from the user certificate. For SiteMinder, this process is named *certificate mapping*.
- Optionally, checks whether the certificate is valid, using Certificate Revocation Lists (CRLs) or the Online Certificate Status Protocol (OCSP).

More information:

Certificate Mapping for X.509 Client Authentication Schemes (see page 403)

Extracting a Certificate for Certificate Authentication

When a user requests a SiteMinder-protected resource, the Web Agent first contacts the Policy Server to determine which authentication scheme is protecting the resource. If an X.509 authentication scheme is protecting a resource, the Web Agent redirects the user's browser to the SiteMinder credential collector that corresponds to the configured authentication scheme. The path to the credential collector is defined in the authentication scheme configuration.

The connection to the credential collector is an SSL-secured connection and the web server is configured to require a client certificate. Therefore, the browser must submit a client certificate for authentication. The resource name and extension at the end of the credential collector URL instructs the Web Agent to extract a user certificate from the web server. The Web Agent then passes the certificate to the Policy Server for use by the authentication scheme.

More information:

Authentication over SSL (see page 307)

How SiteMinder Uses Certificate Data to Identify Users

After the Web Agent collects certificate information, it passes the data to the Policy Server for verification. The Policy Server then performs certificate mapping. The goal of certificate mapping is to locate a SiteMinder user by the Subject Name in the user certificate.

First, the Policy Server looks up the appropriate certificate mapping in the policy store. The Policy Server uses the certificate Issuer DN to locate the mapping. The Issuer DN is part of the certificate mapping configuration. After the Policy Server finds the mapping, it takes the Subject Name from the certificate and applies the mapping to find the user entry in the user directory.

The Policy Server can access user certificates that are stored only in the following repositories:

- LDAP/AD user directory
- ODBC store

Important! You are required to configure certificate mapping for any X.509 client certificate authentication scheme.

More information:

Certificate Mapping for X.509 Client Authentication Schemes (see page 403)

X.509 Client Certificate Scheme Prerequisites

Satisfy the following prerequisites before configuring an X.509 Client Certificate authentication scheme:

- Install an X.509 server certificate on the SSL web server. Be sure that the certificate is not expired.

  Note: If the Policy Server is operating in FIPs mode, ensure the certificate was generated using only FIPS-approved algorithms.
- Verify that the network supports an SSL connection to the client browser (HTTPS protocol).
- Verify that the X.509 client certificates are installed for client browsers. Be sure that the certificates are not expired.

Configure an X.509 Certificate Authentication Scheme

In addition to setting up the SSL environment, complete the following process to configure certificate authentication:

1. Set up your environment to handle SSL communication. Configure the client browser, the web server and any user certificates to accept and perform certificate authentication.
2. Verify that when you installed a SiteMinder Web Agent you configured it to handle SSL authentication.
3. Configure a SiteMinder X.509 authentication scheme in the Administrative UI.
4. Define certificate mappings to identify a user that is based on the information in the client certificate.
5. (Optionally) Configure certificate validation using CRLs or OCSP.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and a protection level.
6. Select the X.509 Client Cert Template from the Authentication Scheme Type list.

   Scheme-specific settings open.

7. Enter the server name and target information for the SSL Credentials Collector.
8. (Optional) Select the Persist Authentication Session Variables in Scheme Setup. This option specifies that the authentication context data is saved in the session store for later use in authentication decisions.
9. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

The X.509 certificate authentication scheme is now configured in the Administrative UI. Now set up certificate mapping (see page 403).

X.509 Client Certificate and Basic Authentication Schemes

The X.509 Client Certificate and Basic authentication scheme combines Basic authentication and X.509 Client Certificate authentication. This authentication scheme provides an extra layer of security for critical resources.

In order for a user to authenticate successfully, the following two events must occur:

- The user's X.509 client certificate must be verified.

  AND

- The user must provide a valid user name and password.

For X.509 Client Certificate authentication, SiteMinder processes authentication using the following steps:

1. The Policy Server instructs the SiteMinder Web Agent to redirect the user to an SSL server and map the user's certificate to the server.
2. SiteMinder verifies the user exists.
3. SiteMinder verifies the user's basic credentials.
4. SiteMinder verifies that the certificate credentials and the basic credentials represent the same user.

More information:

Basic Over SSL Authentication Schemes (see page 314)
X.509 Client Certificate Authentication Schemes (see page 306)

X.509 Client Certificate and Basic Scheme Prerequisites

Ensure the following prerequisites are met before configuring a X.509 Client Certificate and Basic authentication scheme:

- An X.509 Server Certificate is installed on the SSL Web server.

  Note: If the Policy Server is operating in FIPs mode, ensure the certificate was generated using only FIPS-approved algorithms.
- The network supports an SSL connection to the client browser (HTTPS protocol).
- X.509 client certificates are installed on client browsers.
- Trust must is established between client certificates and server certificates.
- The certificate is issued by a valid and trusted Certification Authority (CA).
- The issuing CA's public key validates the issuer's digital signature.
- Client and server certificates have not expired.
- The user's public key validates the user's digital signature.
- Client user name and password information exists in a user directory.
- A directory connection exists between the Policy Server and the user directory.

Note: For Apache Web servers where Certificates are required or optional, the SSL Verify Depth 10 line in the httpd.conf file must be uncommented.

More information:

User Directories (see page 157)

Configure an X.509 Certificate and Basic Authentication Scheme

You can use an X.509 Certificate and Basic authentication scheme to combine certificate authentication and basic authentication.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.

3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.

4. Click OK

   The Create Authentication Scheme page appears.

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

5. Enter a name and a protection level.

6. Select X509 Client Cert and Basic Template from the Authentication Scheme Type list.

   Scheme-specific settings open.

7. Enter the server name and target information for the SSL Credentials Collector.

8. (Optional) Select Persist Authentication Session Variables in Scheme Setup. This option specifies that the authentication context data is saved in the session store.

9. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

X.509 Certificate or Basic Authentication Schemes

The X.509 Client Certificate or Basic authentication scheme allows either Basic authentication or X.509 Client Certificate authentication to establish a user identity. For a user to authenticate successfully, one of the following two events must occur:

- The user X.509 client certificate must be verified.

  OR

- The user must provide a valid user name and password.

With this scheme, when a user requests a protected resource, the Web Agent challenges the browser to present a certificate. If the user does not have a certificate or chooses not to provide one (by clicking Cancel), the Web Agent challenges the user with the HTTP Basic protocol. HTTP Basic authentication allows the agent to obtain a user name and password.

This scheme is useful if you must gradually deploy X.509 certificates. For example, in a company with 50,000 users, it is a challenge to issue and deploy 50,000 certificates simultaneously. This scheme allows you to issue certificates as you see fit (500 or 5,000 at a time). During this transition period, your resources can be protected with certificates for those users who already have them, allowing other authorized users to access resources based on directory user names and passwords.

This scheme gives you the option of configuring the Basic authentication exchange to require an SSL connection.

Note: If you implement multiple certificate-based authentication schemes that include a mixture of X509 Certificate OR Basic schemes, a browser caching limitation may cause unexpected behavior. When a user does not choose certificate-based authentication for accessing a resource in a realm protected by a Certificate or Basic authentication scheme, the browser automatically caches this decision. If the same user (using the same browser session) then attempts to access a resource that is protected by an authentication scheme with a mandatory certificate portion (such as X509 Certificate, X509 Certificate and Basic, or X509 Certificate and Form) the user receives a " Forbidden " error message.

Because the user chose not to send a certificate for the certificate-based authentication when accessing the first resource, and the browser cached that decision, the user is automatically rejected when accessing the realm that requires the certificate.

Encourage users who have valid certificates to use them when accessing resources in a deployment that includes a mixture of realms protected by certificate-based authentication schemes that include X509 Certificate or Basic schemes and other certificate-based schemes that do not allow a user to choose whether to send a certificate for authentication.

More information:

Basic Authentication Schemes (see page 305)
X.509 Client Certificate Authentication Schemes (see page 306)

X.509 Client Certificate or Basic Scheme Prerequisites

Ensure the following prerequisites are met before configuring a X.509 Client Certificate or Basic authentication scheme:

- An X.509 Server Certificate is installed on the SSL Web server.

  Note: If the Policy Server is operating in FIPs mode, ensure the certificate was generated using only FIPS-approved algorithms.
- The network must support an SSL connection to the client browser (HTTPS protocol).
- X.509 client certificates are installed on client browsers.
- Trust is established between client certificates and server certificates.
- Certificates are issued by a valid and trusted Certification Authority (CA).
- The issuing CA’s public key validates the issuer’s digital signature.
- Client and server certificates have not expired.
- The user’s public key validates the user’s digital signature.

- Client user name and password information exists in a user directory.
- A directory connection exists between the Policy Server and the user directory.

Note: For Apache Web servers where Certificates are required or optional, the SSL Verify Depth 10 line in the httpd.conf file must be uncommented.

More information:

User Directories (see page 157)

Configure an X.509 Certificate or Basic Authentication Scheme

You use an X.509 Certificate or Basic authentication scheme to implement certificate authentication or basic authentication or both.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and a protection level.
6. Select X509 Client Cert or Basic Template from the Authentication Scheme Type list.

   Scheme-specific settings open.
7. Enter server and target information for the SSL Credentials Collector.
8. (Optional) Select Persist Authentication Session Variables in Scheme Setup. This option specifies that the authentication context data is saved in the session store.
9. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

X.509 Client Certificate and HTML Forms Authentication Schemes

The X.509 Client Certificate and HTML Forms authentication scheme combines HTML Forms authentication and X.509 Client Certificate authentication. This authentication scheme provides an extra layer of security for critical resources. In order for a user to authenticate successfully, the following two events must occur:

- The user's X.509 client certificate must be verified.

  AND

- The user must provide the credentials requested by an HTML form.

For X.509 Client Certificate authentication, SiteMinder processes authentication using the following steps:

1. The Policy Server instructs the SiteMinder Web Agent to redirect the user to an FCC on an SSL-enabled web server.
2. The Web Agent presents the form.
3. The FCC passes the certificate and form back to the Policy Server.
4. The Policy Server verifies that the user in the certificate mapping exists.
5. The Policy Server verifies the user's HTML form credentials.
6. SiteMinder verifies that the certificate credentials and the HTML Forms credentials represent the same user.

More information:

HTML Forms Authentication Schemes (see page 315)
X.509 Client Certificate Authentication Schemes (see page 306)

X.509 Client Certificate and HTML Forms Scheme Prerequisites

Ensure the following prerequisites are met before configuring a X.509 Client Certificate and HTML Forms authentication scheme:

- An X.509 Server Certificate is installed on the SSL Web server.

  Note: If the Policy Server is operating in FIPs mode, ensure the certificate was generated using only FIPS-approved algorithms.

- The network supports an SSL connection to the client browser (HTTPS protocol).
- X.509 client certificates are installed on client browsers.
- Trust is established between client certificates and server certificates.

- The certificate is issued by a valid and trusted Certification Authority (CA).
- The issuing CA's public key validates the issuer's digital signature.
- Client and server certificates have not expired.
- The user's public key validates the user's digital signature.
- Form credentials information exists in a user directory.
- A directory connection exists between the Policy Server and the user directory.
- (Sun Java Systems) If you are using a Sun Java Systems web server, increase the value of the StackSize parameter in the magnus.conf file to a value greater than 131072. Failing to change the value causes the web server to dump its core and restart each time SiteMinder makes an authentication request using forms.

Note: For Apache Web servers where Certificates are required or optional, the SSL Verify Depth 10 line in the httpd.conf file must be uncommented.

The certificate and forms data are collected and passed to the Policy Server together.

If...	then...
There is no certificate	SiteMinder issues error 500
The certificate and forms credentials are not accepted	SiteMinder issues error 500

More information:

User Directories (see page 157)

Agent API Support

The X.509 Client Certificate and HTML Forms uses the Sm_AuthApi_Cred_SSLRequired and the Sm_AuthApi_Cred_FormRequired bits.

Configure an X.509 Certificate and HTML Forms Authentication Scheme

You can use an X.509 Certificate and HTML authentication scheme to combine certificate authentication and HTML forms-based authentication.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and a protection level.
6. Select X509 Client Cert and Form Template from the Authentication Scheme Type list.

   Scheme-specific settings open.
7. Enter the server name and target information for the SSL Credentials Collector.
8. (Optional) Select Persist Authentication Session Variables in Scheme Setup. This option specifies that the authentication context data is saved in the session store.
9. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

X.509 Client Certificate or HTML Forms Authentication Schemes

he X.509 Client Certificate or HTML Forms authentication scheme allows either HTML Forms authentication or X.509 Client Certificate authentication to establish a user identity. For a user to authenticate successfully, one of the following two events must occur:

- The X.509 client certificate must be verified.

  OR
- The user must provide the credentials requested by an HTML form.

With this scheme, when a user requests a protected resource, the Web Agent challenges the user browser to present a certificate, after which the scheme has the following effect:

If...	then...
A certificate is presented	SiteMinder processes the certificate
The certificate is not accepted	SiteMinder issues error 500
No certificate is presented	SiteMinder presents a form
The form is rejected	SiteMinder prompts again for a form

This scheme is useful if you must deploy X.509 certificates gradually. For example, in a company with 50,000 users, it is a challenge to issue and deploy 50,000 certificates simultaneously. This scheme allows you to issue certificates as you see fit (500 or 5,000 at a time). During this transition period, your resources can be protected with certificates for those users who already have them, allowing other authorized users to access resources based on HTML forms credentials.

Note: If you implement multiple certificate-based authentication schemes that include a mixture of X509 Certificate OR Forms schemes, a browser caching limitation may cause unexpected behavior. When a user does not use the certificate-based authentication for accessing a resource in a realm protected by a Certificate or Forms authentication scheme, the browser automatically caches this decision. If the same user (using the same browser session) then attempts to access a resource that is protected by an authentication scheme with a mandatory certificate portion, such as X509 Certificate, X509 Certificate and Basic, or X509 Certificate and Form, the user receives a " Forbidden " error message.

Because the user chose not to send a certificate for the certificate-based authentication when accessing the first resource, and the browser cached that decision, the user is automatically rejected when accessing the realm that requires the certificate.

Encourage users who have valid certificates to use them when accessing resources in a deployment that includes a mixture of realms protected by certificate-based authentication schemes that include X509 Certificate or Forms schemes and other certificate-based schemes that do not allow a user to choose whether to send a certificate for authentication.

More information:

HTML Forms Authentication Schemes (see page 315)
X.509 Client Certificate Authentication Schemes (see page 306)

X.509 Client Certificate or HTML Forms Scheme Prerequisites

Ensure the following prerequisites are met before configuring a X.509 Client Certificate or HTML Forms authentication scheme:

- An X.509 Server Certificate is installed on the SSL Web server.

  Note: If the Policy Server is operating in FIPs mode, ensure the certificate was generated using only FIPS-approved algorithms.
- The network must supports SSL connection to the client browser (HTTPS protocol).
- X.509 client certificates are installed on client browsers.
- Trust must is established between client certificates and server certificates.
- Certificates are issued by a valid and trusted Certification Authority (CA).
- The issuing CA's public key validates the issuer's digital signature.
- Client and server certificates have not expired.
- The user's public key validates the user's digital signature.
- User attributes requested by the HTML form exist in a user directory.
- A directory connection exists between the Policy Server and the user directory.
- (Sun Java Systems) If you are using a Sun Java Systems web server, increase the value of the StackSize parameter in the magnus.conf file to a value greater than 131072. Failing to change the value causes the web server to dump its core and restart each time SiteMinder makes an authentication request using forms.

More information:

User Directories (see page 157)

Agent API Support

In the Agent API, the value Sm_AuthApi_Cred_CertOrForm has been added to the enumerated type Sm_Api_Credentials_t. Sm_Api_Credentials_t specifies the credentials, if any, that are required for a user to access the realm referenced by the structure Sm_AgentApi_Realm_t. The enumerated type applies to the nRealmCredentials field of the structure.

The new value specifies that user authentication requires either an X.509 certificate or a forms-based authentication scheme.

Configure an X.509 Certificate or HTML Forms Authentication Scheme

You can use an X.509 Certificate or HTML Forms authentication scheme to implement certificate authentication or HTML forms-based authentication.or both.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and a protection level.
6. Select X509 Client Cert or Form Template from the Authentication Scheme Type list.

   Scheme-specific settings open.
7. Enter server and target information.
8. (Optional) Select Persist Authentication Scheme Data in Scheme Setup. This option specifies that authentication context data is saved in the session store.
9. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

Note: For Apache Web servers where Certificates are required or optional, uncomment the SSL Verify Depth 10 line in the httpd.conf file.

Anonymous Authentication Schemes

The Anonymous authentication scheme allows SiteMinder to provide access privileges to users who are not yet identified in your network. Assigning an Anonymous authentication scheme to a realm does not provide access control, but it does allow SiteMinder to personalize content for the user.

When a user accesses a resource in a realm that uses the Anonymous scheme, the Policy Server assigns a Global Unique Identifier (GUID). This GUID is stored on the user's browser and provides a method for identifying the anonymous user.

When you create an Anonymous authentication scheme, you must specify a guest distinguished name (DN). You can bind policies to this guest DN that provide personalized content.

Note: Personalized content in a realm protected by an Anonymous scheme is based on the guest DN, not the GUID of the user. Anonymous users view content according to policies that include the guest DN. Identified users have a distinct DN, so an identified user who accesses the same resource (protected by an anonymous scheme) views the content of the resource based on their unique DN rather than the guest DN.

More information:

Realms (see page 501)

Anonymous Scheme Prerequisites

Ensure the following prerequisites are met before configuring an Anonymous authentication scheme:

- A guest DN for anonymous user exists in a user directory.
- A directory connection exists between the Policy Server and the user directory.
- If you want to track users according to GUIDs assigned by Anonymous authentication, you enable user tracking on the SiteMinder Global Settings pane.

  Note: More information on enabling user tracking exists in the *Policy Server Administration* guide.

More information:

User Directories (see page 157)

Configure an Anonymous Authentication Scheme

You can use an Anonymous authentication scheme to give non-registered users access to specific Web content.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and a protection level.
6. Select Anonymous Template from the Authentication Scheme Type list.

   Scheme-specific settings appear.
7. Enter the DN of a user.
8. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

Custom Authentication Schemes

If you want to use an authentication method that is not provided by SiteMinder, you can create a custom authentication scheme. Once you create a Custom scheme, you must configure the scheme on the SiteMinder Authentication pane.

Note: For information on configuring an smauthetsso custom authentication scheme, which is needed for enabling single sign-on from CA Single Sign-On to SiteMinder, see CA SSO/WAC Integration (see page 717).

Note: If you have installed the Software Development Kit, see the *API Reference Guide for C* for information about creating a custom authentication scheme.

Custom Scheme Prerequisites

The prerequisites of a Custom authentication scheme are determined when you create the scheme using CA's APIs. Prerequisites will differ between authentication schemes.

Configure a Custom Authentication Scheme

You can use a custom authentication scheme to specify a scheme that the product does not provide.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and a protection level.
6. Select Custom Template from the Authentication Scheme Type list.

   Scheme-specific settings open.
7. (Optional) Select the Allow Protection Override check box in Scheme Common Setup. This option specifies that the protection level in the library takes precedence over the protection level specified in the Administrative UI.
8. Enter the library that is to process the credentials for the authentication scheme and the parameters that are passed to the library in Scheme Setup.
9. (Optional) Select Persist Authentication Session Variables in Scheme Setup. If you do not select this option, session variables are not saved in the session store.
10. Click Submit.

    The authentication scheme is saved and can be assigned to a realm.

OpenID Authentication Scheme

This scenario describes how a SiteMinder administrator and a SiteMinder agent owner configure an OpenID authentication scheme.

The SiteMinder OpenID authentication scheme lets SiteMinder users submit credentials through an OpenID provider. The OpenID provider authenticates the user and sends SiteMinder an authentication response. The Policy Server verifies the authentication response, completes the authentication process, and authorizes access to the resource.

The goal of the scenario is to let the intended audience:

- Identify how the OpenID authentication process works.
- Configure the required agent–side components to enable the authentication scheme.
- Configure the required Policy Server–side components to enable the authentication scheme.

SiteMinder r12.5 supports OpenID versions 1.1 and 2.0 and OpenID Attribute Exchange version 1.0.

How OpenID Authentication Works in SiteMinder

The following process explains how OpenID authentication works in SiteMinder:

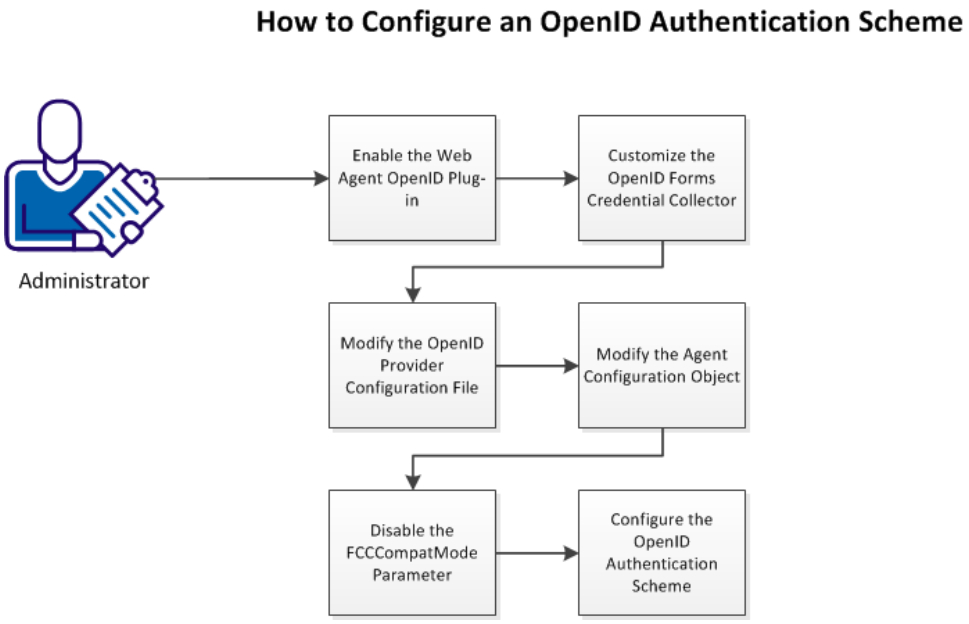
1. A user requests the resource.
2. The agent intercepts the request and contacts the Policy Server to see if the resource is protected.
3. The Policy Server determines that the resource is protected with an OpenID authentication scheme and requests that the agent redirect the user to the OpenID forms credential collector (FCC).
4. The agent redirects the user to the FCC.
5. The user completes one of the following tasks. How the FCC is customized determines the work flow in this step. The user can:
   - Select an OpenID provider and submit OpenID credentials at the provider site.
   - Select an OpenID provider and submit an OpenID user name.
   - Select OpenID and submit the complete OpenID user identifier.
6. The FCC constructs the OpenID user identifier and passes it to the Policy Server.
7. The Policy Server redirects the user to the OpenID provider for authentication by constructing an OpenID authentication request.

8. The user authenticates with the OpenID provider using provider–specific credentials.
9. The OpenID provider redirects the successful authentication response to the FCC.
10. The agent passes the authentication response to the Policy Server.
11. The Policy Server verifies the provider authentication response and uses it to determine the value of the first required claim. The Policy Server searches all user directories in the policy domain to locate a user with an attribute that matches the claim value. Upon a match, the Policy Server authenticates the user.

    Note: The authentication scheme supports an anonymous mode of authentication. In anonymous mode, the Policy Server does *not* use the claim value to map to a SiteMinder user. The Policy Server searches all user directories in the policy domain for a user matching the anonymous user that you defined in the authentication scheme.
12. The Policy Server returns the session details of the authenticated user to the FCC.
13. The FCC creates the SiteMinder session cookie and passes it to the Web browser. The user is redirected to the requested resource and the Policy Server maintains all authorization decisions.

How to Configure an OpenID Authentication Scheme

This section describes how a SiteMinder administrator can configure the OpenID authentication scheme. The following diagram illustrates the required tasks:



Follow these steps:

1. Enable the web agent OpenID plug–in (see page 390).
2. Customize the OpenID forms credential collector (see page 391).
3. Modify the OpenID providers configuration file (see page 393).
4. Modify the agent configuration object.
5. Disable the FCCCompatMode parameter.
6. Configure the OpenID authentication scheme (see page 396).

Enable the Web Agent OpenID Plug–in

The OpenID plug–in is referenced in the web agent configuration file (WebAgent.conf). The plug–in is required to let agents communicate with an OpenID provider and communicate the OpenID provider authentication response to the Policy Server.

Contact agent owners and instruct them to enable the required plug–in.

Follow these steps:

1. Log in to the web agent host system.
2. Open the web agent configuration file.

   Note: The default location of the file depends on the web server type:

   - IIS

     agent_home\bin\IIS

     agent_home

     Specifies the agent installation path.

   - Oracle iPlanet (iPlanet/SunOne)

     web_server_home/https-hostname/config

     web_server_home

     Specifies the web server installation path.

   - Apache, IBM HTTP Server, and Oracle HTTP Server

     web_server_home/conf

     web_server_home

     Specifies the web server installation path.

   - (Windows) Domino

     C:\lotus\domino

   - (UNIX) Domino

     $HOME/notesdata

3. Uncomment line that loads the OpenID plug–in.

   Example:

   ```
   #LoadPlugin="C:\Program Files\CA\webagent\bin\OpenIDPlugin.dll"
   ```

4. Save the file.
5. Restart the web server.

Customize the OpenID Forms Credential Collector

A sample OpenID FCC is included with the web agent installation. The FCC is required to let users authenticate by:

- Entering an OpenID provider user name.
- Entering a complete OpenID identifier.

By default, the FCC presents numerous OpenID providers. Contact the agent owner and instruct them to display only those providers that the protected application supports by modifying the FCC.

Follow these steps:

1. Log in to the web agent host system.
2. Go to the following location:

   agent_home\samples\forms

   agent_home

   Specifies the web agent installation path.

3. Open the following file with a text editor:

   ```
   openid.fcc
   ```

4. Review the FCC. Determine if the OpenID providers you require are available or if you have to add a profile. The default providers are located in the following sections:

   ```
   var providers_large
   var providers_small
   ```

5. Remove the unnecessary providers from the FCC by commenting them. Begin the comment at the provider name. End the comment at the end of the profile.

   Example:

   ```
   /*google : {
       name : 'Google',
       url : 'https://www.google.com/accounts/o8/id'
   },*/
   ```

6. If you have to add a provider, locate the custom provider ID in either the large or small provider sections:

   Example:

   ```
   }/*,
   myprovider : {
       name : 'MyProvider',
       label : 'Enter your provider username',
       url : 'http://ca.com/{username}/',
       image : 'images/image.png'
   }*/
   ```

   Note: The separate provider sections correspond to the sizes of provider icons that the FCC displays:

   - The supported size of an icon in the large section is 100 pixels by 60 pixels. The FCC can display up to five large icons.
   - The supported size of an icon in the small section is 24 pixels by 24 pixels. The FCC can display up to 11 small icons.

   a. Add the new provider by removing the following characters:

      ```
      /*
      */
      ```

   b. Update the label and name values. The label value determines the text that the user sees after clicking the provider icon.

      Example:

      ```
      myprovider : {
      name : 'Foward Inc',
      label : 'Enter your Forward Inc user name',
      url : 'http://ca.com/{username}/',
      image : 'images/image.png'
      }
      ```

      Note: Forward, Inc. is a fictitious company name that is used strictly for instructional purposes only and is not meant to reference an existing company.

   c. Update the URL value. The URL value represents the OpenID user identifier. The Policy Server forwards the user identifier to the OpenID provider.

      Example:

      ```
      myprovider : {
      name : 'Foward Inc',
      label : 'Enter your Forward Inc user name',
      url : 'http://{username}.forwardinc.com/'
      image : 'images/image.png'
      }
      ```

d. Update the image value. The image value represents the location of the provider icon that the FCC is to display.

Example:

```
myprovider : {
name : 'Foward Inc',
label : 'Enter your Forward Inc user name',
url : 'http://{username}.forwardinc.com/'
image : 'images/forwardinc.png'
}
```

7. By default, the FCC displays the provider icons in the order in which the provider ID is configured and enabled in the FCC. If you want to change the icon order, adjust the order of the provider IDs accordingly.

   Important! The default provider IDs include the following image index property:

   `imageidx`

   Do *not* remove or change the property. The property verifies that the FCC displays the correct provider icon.

8. Save the script.
9. Restart the web server.

Modify the OpenID Provider Configuration File

The product provides an OpenID provider configuration file. The file must reference the configuration details of each provider that the protected application supports. If the file does not include the correct settings, authentication fails.

- By default, the file includes sample settings for all of the providers that the OpenID FCC makes available. Review the sample settings and modify them as required.

  Important! The values are samples only. We recommend that you verify all configuration settings with your OpenID provider before deploying the authentication scheme.

- If you added a provider to the FCC, add configuration settings for the provider.

Follow these steps:

1. Log in to the Policy Server host system.
2. Go to the following location:

   siteminder_home`\config\properties`

   siteminder_home

   Specifies the Policy Server installation path.

3. Do one of the following steps:
   - Open the default provider configuration file:

     `Openidproviders.xml`
   - Create another instance by copying the default configuration file. Each OpenID authentication scheme that you configure can use its own provider configuration file.

     Example: You can enable Federal Identity, Credential, and Access Management (ICAM) compliance for one instance of the authentication scheme and can disable ICAM compliance for another.
4. Review the file and determine if the OpenID provider settings you require are available or if you have to add settings.
5. If you have to add settings, complete the following steps:
   a. Copy an existing OpenID provider node and all of its child nodes. All required and optional nodes are included within the following nodes:

      ```
      <OpenIDProvider>
      </OpenIDProvider>
      ```
   b. Add the new OpenID provider node and all of its child nodes to the following root node:

      ```
      <TrustedOpenIDProviders>
      </TrustedOpenIDProviders>
      ```
6. Configure the settings for each provider that the authentication scheme is to support using the following node descriptions:

   OpenIDProvider *RequestType="value"*

   Indicates the beginning of the configuration settings for a provider.

   RequestType

   (Optional) Specifies the schema type that the provider supports.

   Valid values: ax or sreg.

   Default: ax.

   ProviderName

   Specifies the URL of the OpenID provider hosting the service. The value can include a comma–separated list of provider URLs.

   Required Claims

   Specifies the claims that the OpenID provider returns as part of the authentication request. If the provider cannot provide all of the required claims, authentication fails. This node requires at least one claim node.

   Claim

   Defines an individual required claim.

URI

Specifies the URI form of the OpenID provider claim. The Policy Server constructs the authentication request using this value.

Important! Verify that the value of the first required claim maps to a user attribute in your user directories. The Policy Server determines the value of the first required claim that is based on the provider authentication response. The Policy Server then searches all user directories in the policy domain for a user that matches the claim value. If the Policy Server cannot map the claim value to a user attribute, authentication fails.

Value: The value must adhere to the type of schema that the provider supports.

Alias

(Optional) Defines the user-friendly name of the URI node value and prevents the URI from being stored or referenced. The system uses the alias to identify the claim.

Value: Any string.

Example: Instead of storing a URI that returns the first name of users in the session store, the system can reference the claim name as fullname.

Note: The system appends the following prefix to an alias that is stored in the session store:

smopenidclaim

Optional Claims

(Optional) Specifies the optional claims that the OpenID provider is to return as part of the authentication request. If the provider cannot provide an optional claim, authentication does not fail. This node requires at least one claim node.

Pape

(Optional) Defines the properties that ICAM compliance requires. If you are configuring the authentication scheme for ICAM compliance, this node and all child nodes are required.

max_auth_age

(Optional) Specifies the time for which the OpenID provider user session is valid. If the user session is valid, the OpenID provider authenticates the user for a protected resource using a provider–specific cookie. If the session expires, the user is prompted to reauthenticate.

Unit of measurement: seconds.

Default: 0.

If you leave the default value, the user must authenticate against the OpenID provider, regardless of a valid session.

Value: The value must be a positive integer.

Policies

(Optional) Specifies a comma–separated list of the ICAM policies to which the OpenID provider must adhere. If the provider does not adhere to the compliance level, authentication fails.

7. Save and close the file.

Configure the OpenID Authentication Scheme

You use the Administrative UI to configure the OpenID authentication scheme object.

If your network includes multiple cookie domains and each cookie domain requires the authentication scheme, configure a separate authentication scheme object in each cookie domain.

Note: For operation on Solaris platforms, modify the java.security file in the directory *jre_root*/lib/security so that the **sun.security.provider.Sun** provider is registered as the first provider.

Follow these steps:

1. Click Infrastructure, Authentication, Authentication Schemes.
2. Click Create Authentication Scheme.
3. Select the new object option and click OK.

   The Create Authentication Scheme page appears.
4. Enter a name and a protection level.
5. Select the OpenID template from the Authentication Scheme Type list.

   The scheme-specific settings appear.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
6. Configure the remaining parameters and click Submit.

   Important! Restart the Policy Server if you change any of the proxy settings.

You have completed the required tasks to configure an OpenID authentication scheme. You can now bind the authentication scheme to a policy realm and an Enterprise Policy Management (EPM) application component.

Legacy Federation Authentication Schemes

The following legacy federation authentication schemes are available to you:

- SAML Artifact Template
- SAML POST Template
- SAML 2.0 Template
- WS-Federation Template

Note: More information about these authentication schemes, see the *Federation Manager Guide: Legacy Federation*.

Impersonation Authentication Schemes

By configuring a series of Policy Server objects, you can allow privileged users to impersonate other users. This feature is useful in situations where a helpdesk or customer service representative must troubleshoot problems for a customer, or when an employee is out of the office. Part of the impersonation process requires an impersonation authentication scheme which allows a privileged user to begin the impersonation process, identify the user to be impersonated (impersonatee), and establish an impersonation session. This authentication scheme is similar to the HTML Forms authentication scheme.

More information:

HTML Forms Authentication Schemes (see page 315)
Impersonation (see page 663)

Impersonation Scheme Prerequisites

Verify that the following prerequisites are met before configuring an Impersonation authentication scheme:

- A customized .fcc file resides on a Web Agent server in the cookie domain in which you want to implement impersonation. CA provides sample .fcc files under the /forms subdirectory, where you installed your Web Agent.

  For general details about composing .fcc files, see SiteMinder FCC Files (see page 320). For information about specific .fcc file requirements for impersonation, see Enable Impersonation through an .fcc File.

- Directory connections exist between the Policy Server and the user directories containing impersonators and impersonatees.

- The following default HTML forms library, which handles authentication processing is installed on the Policy Server:
  - smauthimpersonate.dll on Windows
  - smauthimpersonate.so on Solaris

  These libraries handle authentication processing These files are installed automatically when you install the Policy Server.

Note: Directory mapping does not support impersonation. The impersonatee, the user being impersonated, must be uniquely present in the authentication directories that are associated with the domain or the impersonation fails.

More information:

User Directories (see page 157)

Configure an Impersonation Authentication Scheme

You use an Impersonation authentication scheme to let privileged users impersonate other users.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and a protection level.

6. Select Impersonation Template from the Authentication Scheme Type list.

   Scheme-specific fields and controls appear.

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

7. Enter the server name and target information.
8. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

Chapter 11: Certificate Mapping and Validity Checking for X.509 Certificates

This section contains the following topics:

Chapter 12: Certificate Mapping for X.509 Client Authentication Schemes

For the Policy Server to use a certificate to identify a user, it compares the certificate information to a user record in the user directory. A certificate mapping defines how the Policy Server uses the Subject Name from the user certificate to locate a SiteMinder user in a user directory and then authenticate that user.

You can configure certificate mapping for users whose authentication information is stored in a Microsoft SQL Server, Oracle, or LDAP user directory.

More information:

User Directories (see page 157)

This section contains the following topics:

Configure a Certificate Mapping

Configure a certificate mapping that lets SiteMinder determine how to compare user certificate information with the information stored in the user directory.

To configure a certificate mapping

1. Click Infrastructure, Directory.
2. Click Certificate Mappings.

   The Certificate Mappings page appears.
3. Click Create Certificate Mapping.

   The Create Certificate Mapping page appears.

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

4. Type the Issuer DN exactly as it appears in the certificate. Do not add any additional spaces or characters.

   When entering the DN, escape reserved special characters with a backslash (\). Special characters include:

   - semicolons (;)
   - quotes (")
   - backslashes (\)
   - plus character (+)
   - greater than character (>)
   - less than character (<)

   More information about reserved special characters for DNs exists at http://www.faqs.org/rfcs/rfc2253.html.

   Note: If you use a relational database as a policy store, Issuer DNs cannot exceed 255 characters. If you use an LDAP directory as a policy store, verify the character limit for your specific directory.

5. Select the directory type against which the certificate is mapped.

   For LDAP directories only, you can configure the Policy Server to verify that the certificate the user presents matches the certificate stored in the user record in the user directory. The Certificate Required in Directory option lets you require this verification.

   Note: The attribute in the user record where the certificate is stored is named **usercertificate**.

6. Specify how to map X.509 user certificate information to a user entry in the user directory. The Policy Server can apply a mapping using a single attribute, a custom mapping expression, or the entire Subject Name from the user certificate to locate the correct user entry.

7. Select an attribute name from the list.

8. Click Test to test the certificate mapping.

9. (Optional) Select Perform CRL Checks and specify the CRL settings.

   If you do not select CRLs, you can use OCSP.

10. Click Submit.

    The certificate is mapped with the selected user directory.

More information:

Certificate Validity Checking (optional) (see page 411)

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.

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 the SiteMinder 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}

More information:

Certificate-based Authentication Tests (see page 698)

EnableCustomExprOnly Registry Key

When you create a custom certificate mapping for an LDAP user directory, the resulting search query string includes the LDAP User DN Lookup Start and End strings in addition to the Mapping Expression that you specify on the Create Certificate Mapping pane. The resulting query is invalid, as seen in the following example:

LDAP User DN Lookup Start

(samAccountName=

LDAP User DN Lookup End

)

Certificate Mapping Expression

(mail=%{E})

Resulting Search Query

(samAccountName=(mail=%{E}))

To omit the User DN Lookup Start and End strings from the search query, navigate to \Netegrity\SiteMinder\CurrentVersion\PolicyServer\ and set the EnableCustomExprOnly registry key to 1. The resulting search query string is valid, as seen in this example:

Certificate Mapping Expression

mail=%{E}

Resulting Search Query

mail=%{E}

Note: If the EnableCustomExprOnly registry key is 0 (the default) or the key does not exist, the User DN Lookup Start and End strings are included in the resulting search query.

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.

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

Custom Mapping: 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.

Custom Mapping: 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. SiteMinder 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}.

Certificate Validity Checking (optional)

Certificate validity checking is an optional feature for X.509 client certificate authentication.

The Policy Server can confirm whether a user certificate is valid using the following methods:

- Certificate Revocation List (CRL) checking

  The Policy Server can use CRLs to determine whether a certificate is revoked. In the Administrative UI, you can specify a path to a CRL directory or select CRL Distribution Points (CDPs) to locate CRLs.

- Online Certificate Status Protocol (OCSP) checking

  The Policy Server sends a request to an OCSP responder in reference to a single user. The OCSP responder determines the revocation status of the user certificate and sends back the response.

The Policy Server determines which certificate validation method it uses as follows:

- If you configure only CRL checking, the Policy Server uses CRLs.
- If you configure only OCSP, the Policy Server uses OCSP.
- If you configure CRL checking and OCSP with failover enabled, the Policy Server uses the designated primary validation method first (CRL or OCSP). If the primary validation method fails, the secondary method is used. For the next request, the Policy Server reverts to the primary method.

The Policy Server regards the first *good* or *revoked* response it obtains to be definitive. The Policy Server does not request subsequent CRLs or OCSP responses after the first valid response. In addition, the Policy Server does not aggregate the results of CRL and OCSP validation to determine the comprehensive status of the user certificate.

More information:

Failover Between OCSP and CRLs (see page 428)

Prerequisites for Implementing Validity Checking

For the Policy Server to validate a user certificate, an X.509 client certificate authentication scheme must be configured and be able to authenticate a user when he requests a protected resource.

Review the instructions for setting up an X.509 client certificate authentication scheme (see page 370).

The instructions for configuring CRLs and OCSP are described in the sections that follow.

Certificate Revocation List Checking

A certificate revocation list (CRL) is a digitally signed list of revoked certificates published by a Certificate Authority (CA) that issued the corresponding certificates. Comparing certificates against CRLs is one method of determining whether a certificate is valid.

You can use one CRL for each Issuer DN that you configure in a Policy Server certificate mapping.

The Policy Server retrieves a CRL in one of the following ways:

- Retrieves a CRL from an LDAP directory

  The Policy Server can establish a connection to an LDAP directory that you specify in the CRL configuration. From that directory, the Policy Server retrieves a CRL.

  Multiple CRLs can exist in an LDAP directory; however, each certificate mapping can refer to only one CRL identified by its entry point. Therefore, each entry point for each CRL must be different.

  Note: The Policy Server only accepts a CRL that contains all reason codes, and rejects a CRL with only specific reason codes.

- Retrieves a CRL from a location specified by a CRL distribution point extension (CDP)

  A user certificate can contain a distribution point extension. A CDP extension points to a location from where a CRL can be obtained. The Policy Server supports a distribution point extension with a single entry to a CRL or multiple pointers to different CRLs.

  The distribution points can use different sources, such as HTTP, HTTPS, and LDAP. If the CDP extension contains entries that use distribution point names with multiple values, these values must all point to the same CRL.

After the Policy Server retrieves a CRL, it can make the necessary checks. If you enable CRL caching in the Administrative UI, the Policy Server can store the CRL in memory. If you do not enable caching, the Policy Server has to go out and retrieve a CRL for every authentication request.

Reason Code Requirements for CRLs

The Policy Server only supports CRLs that include revocation information for all possible reason codes. If a CRL contains certificates revoked for only some reason codes, the Policy Server generates an error and treats the CRL as invalid. The Policy Server ignores invalid CRLs and continues looking for an available CRL until it finds a valid one.

The Policy Server treats *delta* CRLs as invalid CRLs. A delta CRL lists only those certificates whose revocation status changed after the CA issued the complete CRL. The Policy Server ignores delta CRLs and continues looking for an available CRL until it finds a valid one.

If the Policy Server searches through all available CRLs and cannot find a valid one, it does not authenticate the user.

Size Limits for CRLs

The Policy Server caches CRLs. The Policy Server default cache size is up to 2 MB. If your CRLs exceed the default cache size, increase the cache size up to a maximum of 1 GB. To increase the cache size, add the MaxCRLBufferMB registry key.

Follow these steps:

1. Access the Policy Server and follow the step for your operating platform:

   Windows: Open the Registry Editor and navigate to HKEY_LOCAL_MACHINE\Software\Netegrity\SiteMinder\CurrentVersion\PolicyServer.

   UNIX: Open the sm.registry file. The default location of this file is *siteminder_home*/registry.

   siteminder_home

   Specifies the Policy Server installation path.

2. Add MaxCrlBufferMB with a registry value type of REG_DWORD.

   Unit of measurement: Megabytes

   Base: Decimal

   Default value: 2

   Minimum value: 1

   Maximum value: 1023

3. Complete one of the following steps:

   Windows: Exit the Registry Editor.

   UNIX: Save the sm.registry file.

4. Restart the Policy Server.

CRL Signature Verification

CRL signature verification is an optional feature of CRL checking.

Before the Policy Server compares certificates against a CRL, it verifies the signature of the CRL with a CA certificate stored in an LDAP directory. The Policy Server retrieves the CA certificate from a specific entry in an LDAP user directory, which is identified based on the Issuer DN in the certificate or the DN in the CRL directory that you configure for the certificate mapping in the Administrative UI.

Store the CA certificates in an LDAP directory that the Policy Server can access. In the LDAP directory, configure the specific directory entry with an attribute named **cacertificate**. The cacertificate attribute is a multi-valued attribute where you can store more than one CA certificate. Multiple CA certificates can be necessary if CRLs are partitioned and a different CA key signs each partition. The Policy Server can only verify the signature of a partition if it can access the associated CA signing certificate for a given partition.

For signature verification, the Policy Server can use the following hash algorithms:

- MD5
- MD2
- SHA1
- SHA2 algorithms (SHA224, SHA256, SHA384, and SHA512)

Note: The signature algorithm in use is specified in the CRL.

If a CA certificate is not available or your CRL is signed with an unsupported algorithm, you can disable signature checking during the CRL verification process.

Important! If signature checking is turned off, confirm that the repository where CRLs are stored is protected appropriately.

CRL Distribution Points to Locate CRLs

A CRL Distribution Point (CDP) is a certificate extension that points to a location of a CRL. From the specified location, the Policy Server can retrieve the CRL and can confirm which certificates are revoked.

A CDP extension can specify several sources to locate a CRL. Each source contains all the information to locate a CRL. The different options for retrieval help ensure that the Policy Server obtains a CRL. The sources in a CDP extension include:

- LDAP
- HTTP

- HTTPS

  For an HTTPS distribution point, the Policy Server makes a secure connection. To make this secure connection, a valid CA certificate file or certificate bundle (a file of concatenated certificates that form a chain) must exist in the directory *policy_server_home*/config. If there is no valid certificate, the connection to the HTTPS server fails.

The CA certificate file for the HTTPS connection must be in PEM format (base64 encoded) and named **cert.pem**. If the certificate is not in the PEM format, convert it using the OpenSSL command–line utility. The cert.pem file must contain the issuer certificate for the SSL web server configured in the CDP extension, and it must contain the trusted CA certificate for each distribution point.

Note: For more information about the OpenSSL utility, see the OpenSSL documentation.

If a CDP extension has multiple entries, the Policy Server uses the first successfully retrieved CRL with all reason codes to validate certificates. The order in which it retrieves the CRLs is the same order that the entries are listed in the certificate itself. If the Policy Server cannot retrieve the first CRL in the CDP list, it tries to retrieve the second CRL, and so on. The Policy Server continues in this manner until it is successful.

If the Policy Server cannot retrieve a valid CRL from any source, authentication fails and the user is denied access. Enabling failover between CRLs and OCSP is the only exception to this behavior. If CRL checking is the primary validation method and it fails, the Policy Server fails over to OCSP as the secondary method.

Note: Enable failover (see page 428) in the configuration file for OCSP.

Configure CRL Distribution Points as part of the CRL Checking settings in the Certificate Mapping dialog.

Verifying Signatures of Partitioned CRLs

Different CA keys can sign different partitions of CRLs. The Policy Server can verify the signature of any CRL partition as long as it can access the associated CA signing certificate for each partition.

The use of partitioned CRLs is relevant when using certificate distribution points to locate CRLs. The extension can have multiple links to different CRLs, all whose signatures need verifying.

The Policy Server verifies the signature of the CRL with the CA certificate stored in an LDAP directory. In this LDAP directory, configure a specific entry with the attribute named **cacertificate**, which is a multivalued attribute. Multiple CA certificates are required to verify partitioned CRLs signed by different CA keys.

Configure Certificate Revocation List Checking

Configure CRL checking to verify whether a user certificate has been revoked. This verification helps ensure that a user with an invalid client certificate cannot access a protected resource.

You can obtain a CRL from an LDAP directory or from a location specified by a CDP. If the Policy Server is going to obtain CRLs from a specific LDAP directory, be sure to configure a connection to that directory in the User Directory section of the Administrative UI. This LDAP directory can act as a user store and a CRL store. Configure the directory before configuring CRL checking or during the CRL configuration process.

To configure CRL checking

1. Click Infrastructure, Directory.
2. Click Certificate Mappings.

   The Certificate Mappings page appears.
3. Click the Issuer DN name to select the certificate mapping.

   The View Certificate Mapping page appears.
4. Click Modify.

   The settings and controls become active.
5. Select Perform CRL Checks.

   CRL-specific fields and controls display.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
6. Select the name of the LDAP directory from where the Policy Server obtains the CRL,in CRL Directory.

   The directory name is the name you assigned when configuring the directory in the User Directory section of the Administrative UI. If there is no user directory in the list, click Create to add a directory connection.

   If you do not specify an LDAP directory, select Use Distribution Points as the method by which the Policy Server retrieves a CRL.

   Note: An optional text string value for the CRL Directory field exists and it reads "Take from Certificate Extension." Only select this option if you plan to use distribution points for CRL retrieval.
7. If you specified a user directory in CRL Directory, enter a value for the entry point in DN in CRL Directory.

   The value specified in DN in CRL Directory is the DN where the Policy Server looks in the CRL directory to locate the CRL. This value is valid only when an LDAP directory is selected as the CRL Directory . If you enable distribution points to locate CRLs, leave this field blank.

8. (Optional) Select Verify signature to verify the signature of the CRL.

   The Policy Server requires an accessible LDAP host to retrieve the certificate necessary to verify the signature of the CRL. Be sure that you have configured an LDAP host as a user directory connection in the Administrative UI.

   The Policy Server can use a CRL distribution point to locate a CRL. If that distribution point is an LDAP URI, the Policy Server can verify the CRL signature. If the distribution point is an HTTP URI, do not select the Verify Signature option because no LDAP host is available from which to retrieve the certificate.

9. (Optional) Select Use Distribution Points to use the CDP extension to locate CRLs. You can use this option as an alternative to specifying a CRL directory.

   If you select Use Distribution Points and enter a directory in CRL Directory, the Policy Server uses only the distribution points to locate the CRL. Distribution points take precedence over the CRL directory.

10. Complete the remaining settings, as necessary, and click Submit.

    Certificate revocation list checking is enabled.

Accessing a CRL through an HTTP Proxy

CRLs requests can be made over an HTTP connection, requiring an HTTP GET request to retrieve the CRL for certificate validation.

In many enterprise environments, HTTP traffic goes through an HTTP proxy. For the Policy Server to retrieve a CRL through an HTTP proxy, set the http_proxy environment variable on the system where the Policy Server resides. For example:

```
set http_proxy=http://username:password@proxy.example.org:8080
export http_proxy
```

If you do not specify a port number, SiteMinder defaults to port 1080.

username

The login user name for the proxy server. This name has to be a valid user in the proxy server configuration.

password

The login password for the proxy server. This password has to be a valid entry in the proxy user configuration.

Note: You cannot use this environment variable for accessing an OCSP Responder through a proxy.

Online Certificate Status Protocol Checking (OCSP)

The Online Certificate Status Protocol (OCSP) helps maintain security across your environment by verifying whether user certificates are valid. OCSP uses OCSP responders to determine the revocation status of an X.509 client certificate. The OCSP responder does its verification in real time by aggregating certificate validation data and responding to an OCSP request for a particular certificate.

One benefit of OCSP is that you do not have to keep downloading a CRL at the client side to maintain the most up-to-date certificate status information. Additionally, CRLs can be large.

To implement OCSP checking, the Policy Server uses a text-based configuration file named SMocsp.conf file. This file is located in the directory *policy_server_home*/config, and it must exist to use the SiteMinder OCSP feature.

The SMocsp.conf file contains settings that define the operation of one or more OCSP responders. When verifying if a user certificate is valid, the Policy Server looks for an Issuer DN in the SMocsp.conf file. If it finds the Issuer DN, a certificate status check is made using the specified OCSP responder associated with the Issuer DN. If the Policy Server does not find the Issuer DN, the Policy Server does not perform OCSP checking and the certificate is considered valid. A certificate is considered valid in the absence of an Issuer DN to satisfy cases where OCSP validation is not required.

You can sign an OCSP request; however, signing requests is an optional feature.

When the OCSP responder returns a response to the Policy Server, the Policy Server default behavior is to validate the signed response. Several settings in the SMocsp.conf file require configuration to enable response verification.

Note: If CRL checking is enabled in the Administrative UI, the Policy Server uses CRL checking by default, regardless of whether an SMocsp.conf file is present. OCSP take precedence over CRL checking only if you enable failover and set OCSP as the primary validation method. Failover is configured in the OCSP configuration file.

OCSP Prerequisites

Set up the following components to use OCSP for certificate validation:

- Establish a Certificate Authority (CA) environment.
- Set up an OCSP responder.
- Configure an LDAP directory to store an OCSP trusted responder certificate that validates the signature of an OCSP response returned to SiteMinder. The OCSP trusted responder certificate is a single trusted verification certificate or a collection of certificates.

  You obtain these certificates in a communication that is separate from an OCSP transaction.

To store the trusted certificate or collection of certificates, configure the LDAP directory in the User Directory section of the Administrative UI. The presence of the user directory enables the Policy Server to connect to it and locate the certificate or collection of certificates to verify the response signature. If you are storing a collection of certificates, be sure to use a multi–valued binary attribute for the directory entry to store all the certificates.

The OCSP responder can include the signature verification certificate with the response. In this case, the Policy Server validates the certificate and the response signature with the trusted certificate in the LDAP directory.

If the signature verification certificate is not in the response, the Policy Server verifies the signature of the response with the certificate or collection of certificates in the LDAP directory.

When you configure OCSP, specify the location of the certificate or the collection of certificates in the ResponderCertEP setting of the SMocsp.conf file.

The Policy Server can work with any OCSP response that is signed using SHA-1 and the SHA-2 family of algorithms (SHA224, SHA256, SHA384, SHA512).

- Store the CA certificate that issued the user certificate in an LDAP directory. This CA certificate validates the user certificate. You can store this certificate in the same LDAP directory where you store the OCSP trusted responder certificate or in a different LDAP directory.
- Configure a SiteMinder OCSP configuration file (SMocsp.conf). A sample configuration file, SMocsp.Sample.conf, is installed with the Policy Server. Copy this file, rename it to SMocsp.conf, then modify it as needed.

  For UNIX operating platforms, the file name must maintain the case-sensitivity.
- Optionally, be sure that the private key/certificate pair that the Policy Server uses to sign the OCSP request is available to the Policy Server. Store this key/certificate pair in the certificate data store.

Create an OCSP Configuration File

The Policy Server uses a file named SMocsp.conf to implement OCSP checking. This file is an ASCII file with one or more OCSPResponder records.

The SMocsp.conf file must reside in the directory *siteminder_home*/config. For ease of configuration, a sample file (SMocsp.Sample.conf) is installed with the Policy Server in the config folder. To configure OCSP for your environment, copy the sample file and rename it SMocsp.conf.

Note: For UNIX platforms, maintain the case–sensitivity of the file name.

The following excerpt is an example of an SMocsp.conf file with a single OCSPResponder entry.

Note: The sample file shows all available settings. Not all settings are required.

```
[
OCSPResponder
IssuerDN C=US,ST=Massachusetts,L=Boston,O=,OU=QA,CN=Issuer,E=user@domain.com
AlternateIssuerDN C=US,ST=New York,L=Islandia,O=,OU=QA,CN=Issuer,E=user@domain.com
CACertDir 10.1.22.2:389
CACertEP uid=caroot,dc=systest,dc=com
ResponderCertDir 10.2.11.1:389
ResponderCertEP cn=OCSP,ou=PKI,ou=Engineering,o=ExampleInc,c=US
ResponderCertAttr cacertificate
ResponderLocation http://10.12.2.4:389
AIAExtension NO
HttpProxyEnabled YES
HttpProxyLocation http://10.11.2.5:80
HttpProxyUserName proxyuser1
HttpProxyPassword letmein
SignRequestEnabled YES
SignDigest SHA256
Alias defaultenterpriseprivatekey
IgnoreNonceExtension NO
PrimaryValidationMethod OCSP
EnableFailover YES
ResponderCertAlias cert1
ResponderGracePeriod 0
]
```

Guidelines for modifying the SMocsp.conf file are as follows:

- Names of settings are not all case-sensitive. Case sensitivity for entries is dependent on the particular setting.
- If a setting in the file is left blank, the Policy Server sends an error message. The message indicates that the entry is invalid. The Policy Server ignores the setting. If you intended to leave the setting blank, disregard the message.
- Do not put leading white spaces in front of the name of a setting.

The settings in the file are as follows:

OCSPResponder

Required. Indicates that the entry is an OCSP responder record. Each OCSP Responder record must start with the name OCSPResponder.

IssuerDN

Required. Specifies the DN of the certificate issuer. This value labels each OCSP Responder record in the file.

Entry: The Issuer DN value in the certificate.

AlternateIssuerDN

Optional. Specifies a secondary IssuerDN or reversed DN.

CACertDir

Required. Specifies the name of the CA certificate directory that holds the CA certificate that issues the user certificate.

Be sure to configure this directory as a SiteMinder user directory in the Administrative UI so the Policy Server can connect to it.

Enter a valid IP address and port number of the user directory.

CACertEP

Required. Specifies the entry point in the CA certificate directory where the CA certificate resides.

Enter a string representing an entry point in the certificate directory.

ResponderCertDir

Required. Specifies the LDAP directory where the responder certificates is stored.

Be sure to configure this directory as a SiteMinder user directory in the Administrative UI so the Policy Server can connect to it.

Enter a valid IP address and port number of the directory.

ResponderCertEP

Required. Specifies the entry point in the LDAP directory where the responder certificate resides. The responder certificate directory is specified in the ResponderCertDir setting.

The signature verification certificate is the certificate that directly verifies the response signature or the collection of intermediate certificates.

The OCSP responder can include the signature verification certificate with the response. In this case, the Policy Server verifies the response signature and the certificate using the trusted certificate in the LDAP directory. If the signature verification certificate is not in the response, the Policy Server verifies only the response signature with the certificate or collection of certificates in the LDAP directory.

Enter a string representing an entry point in the directory where the certificate or the collection of certificates resides.

ResponderCertAttr

Required. Indicates the LDAP directory attribute that the Policy Server uses to look up the responder certificate in the responder certificate directory, which is specified in the ResponderCertDir setting.

ResponderLocation

Optional. Indicates the location of the OCSP responder server.

You can use the ResponderLocation setting or the AIAExtension setting, but note the following conditions:

- If the ResponderLocation setting is left blank or it is not in the SMocsp.conf file, set the AIAExtension setting to YES. Additionally, an AIA extension must be in the certificate.
- If the ResponderLocation setting has a value and the AIAExtension is set to YES, the Policy Server uses the ResponderLocation for validation. The ResponderLocation setting takes precedence over the AIAExtension.
- If the OCSP responder specified for this setting is down and the AIAExtension is set to YES, authentication fails. The Policy Server does not try the responder specified in the AIA extension of the certificate.

If you enter a location, enter the value in the form *responder_server_url*:*port_number*.

Enter a URL and port number of the responder server.

AIAExtension

Optional. Specifies whether the Policy Server uses the Authority Information Access extension (AIA) in the certificate to locate validation information.

You can use the AIAExtension or ResponderLocation settings, but note the following items:

- If AIAExtension is set to YES and the ResponderLocation is not configured, the Policy Server uses the AIA Extension in the certificate for validation. The extension has to be in the certificate.
- If the AIAExtension is set to YES and ResponderLocation setting also has a value, the Policy Server uses the ResponderLocation for validation. The ResponderLocation setting takes precedence over the AIAExtension.
- If AIAExtension is set to NO, the Policy Server uses the ResponderLocation setting. The Policy Server disregards the AIA exteniонsion if it exists.

Enter YES or NO.

Default: NO

HttpProxyEnabled

Optional. Tells the Policy Server to send the OCSP request to the proxy server, not to the web server.

Enter YES or NO.

Default: NO

HttpProxyLocation

Optional. Specifies the URL of the proxy server. This value is only required if HttpProxyEnabled is set to YES.

Enter a URL beginning with http://.

Note: Do not enter a URL beginning with https://.

HttpProxyUserName

Optional. Specifies the user name for the login credentials to the proxy server. This user name must be the name of a valid user of the proxy server. This value is only required if HttpProxyEnabled is set to YES.

Enter an alphanumeric string.

HttpProxyPassword

Optional. Specifies the password for the proxy server user name. This value is displayed in clear text. This value is only required if HttpProxyEnabled is set to YES.

Enter an alphanumeric string.

SignRequestEnabled

Optional. Instructs the Policy Server to sign the generated OCSP request. Set this value to Yes to use the signing feature.

This value is independent of any user certificate signatures and is only relevant for the OCSP request.

Note: This setting is required only if the OCSP responder requires signed requests.

Enter YES or NO.

Default: NO

SignDigest

Optional. Designates the algorithm the Policy Server uses when signing the OCSP request. This setting is not case-sensitive. This setting is required only if the SignRequestEnabled setting is set to YES.

Enter one of the following options: SHA1, SHA224, SHA256, SHA384, SHA512

Default: SHA1

Alias

Optional. Specifies the alias for the key/certificate pair that signs the OCSP request that is sent to an OCSP responder. This key/certificate pair must be in the SiteMinder certificate data store.

Note: The alias is required only if the SignRequestEnabled setting is set to YES.

Enter an alias using lower-case ASCII alphanumeric characters.

IgnoreNonceExtension

Optional. Tells the Policy Server not to include the nonce in the OCSP request. The nonce (number that is used once) is a unique number sometimes included in authentication requests to prevent the reuse of a response. Setting this parameter to Yes instructs the Policy Server *not* to include the nonce in the OCSP request.

Enter YES or NO.

Default: NO

PrimaryValidationMethod

Optional. Indicates whether OCSP or CRL is the primary method the Policy Server uses to validate certificates. This setting is only required if the EnableFailover setting is set to Yes.

Enter OCSP or CRL.

Default: OCSP

EnableFailover

Tells the Policy Server to failover between OCSP and CRL certificate validation methods.

Enter YES or NO.

Default: NO

ResponderCertAlias

(Required for federation only). Names the alias of the certificate that verifies the signature of the OCSP response. For the Policy Server to perform response signature verification, specify an alias for this setting. Otherwise, the CA issuer has no available OCSP configuration.

Note: The Policy Server does not use this setting for X.509 certificate authentication.

Enter a string that names the alias.

You can see whether each issuer has an OCSP configuration after the SMocsp.conf file is loaded. The following message is a sample status message:

```
The SMocsp.conf file was loaded.
OCSP configuration was added for the following issuer aliases:
ocspcacert
ocspcacert1
ocspcacert2
```

The issuer alias in the status message refers to the alias you specified in the Administrative UI when adding a CA certificate to the data store. If an issuer alias is not in the list, check the SMocsp.conf and the cds.log file. The log file is located in *siteminder_home*\log.

RevocationGracePeriod

(Optional) Specifies the period (in days) to delay the invalidation of a certificate after it is revoked. The OCSP grace period gives you time to update certificates so that the configuration does not suddenly stop working. A value of 0 indicates that when a certificate is revoked it becomes invalid immediately.

If you do not specify a value for this field, the Policy Server uses the default revocation grace period setting in the Administrative UI. You can find the default setting by navigating to Infrastructure> X509 Certificate Management > Certificate Management.

Default: 0

Configure OCSP Checking

Configure OCSP checking so that a user with an invalid client certificate cannot access a protected resource.

Note: Before you can enable OCSP checking, set up your environment for certificate authentication (see page 370).

The Policy Server can work with any OCSP response that is signed using SHA-1 and the SHA-2 family of algorithms (SHA224, SHA256, SHA384, SHA512).

To configure OCSP checking (without failover enabled)

1. Navigate to *policy_server_home*/config.

   A sample file, named SMocsp.Sample.conf is installed with the Policy Server in the config directory.

2. Copy the sample configuration file and rename it SMocsp.conf.

   For UNIX platforms the file name is case-sensitive; for Windows platforms it is not.

3. Open the file in a text editor.

4. Add a unique OCSPResponder entry in the file for each IssuerDN that matches an IssuerDN specified in your certificate mapping.

   Important! If you do not configure a responder record for each Issuer DN, the Policy Server authenticates users without confirming the validity of the certificate.

5. Save the file.

6. Restart the Policy Server.

7. Log on to the Administrative UI.

8. Select Infrastructure, Directory.

9. Expand the Certificate Mapping option.

10. Select Create or Modify a Certificate Mapping.

    The Certificate Mappings dialog opens.

11. Clear the Perform CRL Checks check box if OSCP is the only validity checking method that you plan to use. Do not disable CRL checking if you plan to use failover.

    If the issuer of the user certificate matches a certificate mapping and CRL Checking is enabled, the Policy Server uses CRL checking and not OCSP. Enabling failover is an exception to CRL checking taking precedence over OCSP. If you enable failover (see page 428), the Policy Server uses the configured primary validation method.

12. Save the changes then exit the Administrative UI.

13. (Optional) Configure the Policy Server to sign the OCSP requests.

OCSP is now enabled. To disable OCSP, change the name of the SMocsp.conf file.

Accessing an OCSP Responder through a HTTP Proxy

OCSP requests are made over an HTTP connection, requiring an HTTP GET for the request to the OCSP responder for certificate validation.

In many enterprise environments, HTTP traffic goes through an HTTP proxy. For the Policy Server to send an OCSP request through an HTTP proxy, configure the proxy settings in the SMocsp.conf file.

To configure access to an OCSP Responder through a proxy

1. Edit the existing SMocsp.conf file or create a file in the Policy Server config directory, *policy_server_home*/config.
2. Edit the following settings as follows:
   - HttpProxyEnabled YES
   - HttpProxyLocation *proxy_server_URL*
   - HttpProxyUserName *user_login_name*
   - HttpProxyPassword *password_for_login*

   Learn how to set each value by reviewing how to create an OCSP configuration file.
3. Restart the Policy Server.

Signing OCSP Requests (Optional)

The Policy Server can sign OCSP requests when using a SiteMinder certificate authentication scheme. Signing requests is only necessary if an OCSP responder requires signed requests.

Prerequisites for Signing OCSP Requests

Before you configure OCSP signing, complete the following prerequisite tasks:

1. Configure OCSP checking (see page 418).
2. Add the key/certificate pair that signs requests to the certificate data store. Perform this task using the Administrative UI.

   When you add a key/certificate pair, specify an alias. The Policy Server uses the alias to identify the certificate entry in the certificate data store. When you specify an alias, note the following restrictions:

   - Store a certificate only once under a single alias. Attempts to store the same certificate under a different alias fail. Use the same alias for multiple responders if they use the same signing certificate.
   - The alias value that you specify must match the value for the alias setting in the SMocsp.conf file.
   - A certificate alias can be any name, but the first alias must be *defaultenterpriseprivatekey*.
   - An alias must:
     - Be all lower–case USA ASCII
     - Use alphanumeric characters.

   Note: Private keys must be RSA keys.

Configure OCSP Request Signing

The Policy Server can sign requests and verify responses when using a SiteMinder certificate authentication scheme.

To configure OCSP request signing

1. Open the SMocsp.conf file in an editor. The file is in the directory *policy_server_home*/config.
2. Add the following entries to the SMocsp.conf file for each responder:

   SignRequestEnabled

   Instructs the Policy Server to sign an OCSP request.

   Limits: Yes or No

   Set this value to Yes to use the signing feature.

   Accept the default, No, to disable signing.

   Default: No. If the SignRequestEnabled entry is not present in a responder record, the Policy Server cannot sign OCSP requests.

SignDigest

Indicates the algorithm to use to sign an OCSP request.

Options:

- SHA1
- SHA224
- SHA256
- SHA384
- SHA512

SignDigest is not case-sensitive.

Default: SHA1

Alias

Indicates the alias under which the Policy Server looks to retrieve the signing key/certificate pair. Aliases are restricted to lower-case USA ASCII alphanumeric characters.

3. Save the SMocsp.conf file.
4. Restart the Policy Server.

Signing of OCSP requests is now enabled.

Failover Between OCSP and CRLs

The Policy Server can use OCSP and CRLs as certificate validation mechanisms. If you configure both mechanisms, you can configure the Policy Server to failover between the two. Enabling failover is preferable to failing the authentication when one or the other certificate validation mechanisms is not available.

You implement failover for certificate checking by designating a primary verification method in the OCSP configuration file (SMocsp.conf). If the primary validation method is unavailable, the Policy Server uses the secondary validation to complete the request. For the next request, the Policy Server reverts to the primary method and uses that method unless a failure occurs.

OCSP as the Primary Validation Method

If the primary method is OCSP and an OCSP responder is not available, the Policy Server uses a CRL to perform the certificate validation instead.

Failover does not override OCSP functionality as long as the OCSP responder returns a response indicator of *good* or *revoked*. If the response indicator is "unknown," failover to CRL checking occurs.

If you configure OCSP as the primary validation method, the Policy Server behaves as follows:

OCSP Certificate Validation	Failover Disabled	Failover Enabled
Valid	User authenticated	User authentication based on OCSP results only. No CRL checking required.
Revoked	User not authenticated	User is not authenticated. No CRL checking required.
Unknown or No response	User not authenticated	Perform CRL checking

CRL Checking as the Primary Validation Method

If the primary method is CRL checking and the Policy Server cannot retrieve a CRL, the Policy Server fails over to the OCSP responder. In this case, the Policy Server only relies on OCSP when a connection to the CRL directory server is not available. If the CRL returns a valid response, the Policy Server does not use OCSP.

Note: If failover is not enabled, and CRL checking and OCSP are both configured, the Policy Server uses only CRL checking for certificate validation.

If you configure CRL as the primary validation method, the Policy Server behaves as follows:

CRL Certificate Validation	Failover Disabled	Failover Enabled
Valid	User authenticated	Checking is based on the first valid CRL results. No further CRL checking required.
Revoked	User not authenticated	No further CRL or OCSP checking required.
No response/retrieval failed	If a CDP extension is available, the Policy Server tries each distribution point in consecutive order until it successfully retrieves a CRL. If the status for the certificate is valid or revoked, refer to the descriptions for those states. If a CRL with all reason codes is not retrieved, the Policy Server defaults to Not Authenticated.	If a CDP extension is available, the Policy Server tries each distribution point in consecutive order until it successfully retrieves a CRL. If the status for a certificate is valid or revoked refer to the descriptions for those states. If a CRL with all reason codes is not retrieved, use OCSP.

Configure Failover Between OCSP and CRLs

The Policy Server can use OCSP and CRLs as certificate validation mechanisms, enabling failover between the two. Before enabling failover, configure CRL checking in the Administrative UI and configure OCSP by creating the SMocsp.conf file. CRL checking and OCSP checking must be enabled for failover to work.

To enable OCSP Failover to a CRL

1. Open the SMocsp.conf file in an editor. This file is in the directory *policy_server_home*/config.
2. Add or modify the following entries for each responder record:

   EnableFailover

   Enables SiteMinder to fail over from the primary validation method to the secondary method.

   Set this value to Yes to enable failover.

   Accept the default, No, if you do not want to configure failover. If you do not configure failover and the OCSP responder cannot perform validation, the transaction fails.

   Limits: YES or NO

   Default: No

   PrimaryValidationMethod

   Indicates which certificate validation method the Policy Server uses first before trying the other method.

   If EnableFailover is set to YES and the value for this setting is OCSP, the Policy Server uses OCSP validation first. If there is no response from the OCSP responder or the response indicator is "unknown," then the Policy Server fails over to a CRL.

   If the value for this setting is CRL, the Policy Server ignores OCSP validation, even if it is configured, and uses a CRL. If the Policy Server cannot obtain the CRL or the CRL expires, the Policy Server fails over to OCSP.

   Limits: OCSP, CRL

   Default: OCSP
3. Save the changes to the SMocsp.conf file.
4. Restart the Policy Server.

Troubleshooting Certificate Validation

Detailed trace logging is available to help you solve your X.509 certificate authentication and validation problems.

In addition to the typical OCSP and CRL messages, if a failover event occurs the Policy Server logs diagnostic messages specific to the certificate validation failure, followed by messages describing the failover. The message can indicate that OCSP could not be contacted and that it is using a CRL or that the CRL fetch failed and that the Policy Server is failing over to OCSP checking.

To view OCSP and CRL log message, enable authentication trace logging using the Profiler in the Policy Server Management Console.

You can determine which components and data fields the Policy Server includes for trace logging by modifying the default template file smtracedefault.txt.

The following smtracedefault.txt file shows some recommended components to include in the file for certificate validation diagnostics in the trace log.

```
components: Login_Logout/Authentication, Login_Logout/Certificates,
Login_Logout/Receive_Request, IsAuthorized/Policy_Evaluation,
IsAuthorized/Receive_Request, Directory_Access, LDAP/Ldap_Error_Messages

data: Date, PreciseTime, SrcFile, Function, ReturnValue,
Message, User, Directory, SearchKey, ErrorString, ErrorValue,
AuthStatus, AuthReason, CertSerial, SubjectDN, IssuerDN,
CertDistPt, UserDN, Data, HexadecimalData, CallDetail, Returns, Result
```

For instructions on enabling authentication trace logging, see the *Policy Server Administration Guide*.

For OCSP signing only, you can enable trace messages when trying to validate signatures.

To enable tracing for OCSP signing

1. Navigate to *policy_server_home*/config.

   policy_server_home is the directory where you installed the Policy Server

2. Open the JVMOptions file in a text editor.

3. Add the setting -DOCSP_PS_TRACE and set it to true, as follows:

   -DOCSP_PS_TRACE=true

4. Save the file
5. Restart the Policy Server.

The trace file, named OcspCertKeyRetriever.log is written to the current working directory of the Policy Server, as follows:

Windows: system32

Unix: siteminder or siteminder/bin

Chapter 13: Strong Authentication

This section contains the following topics:

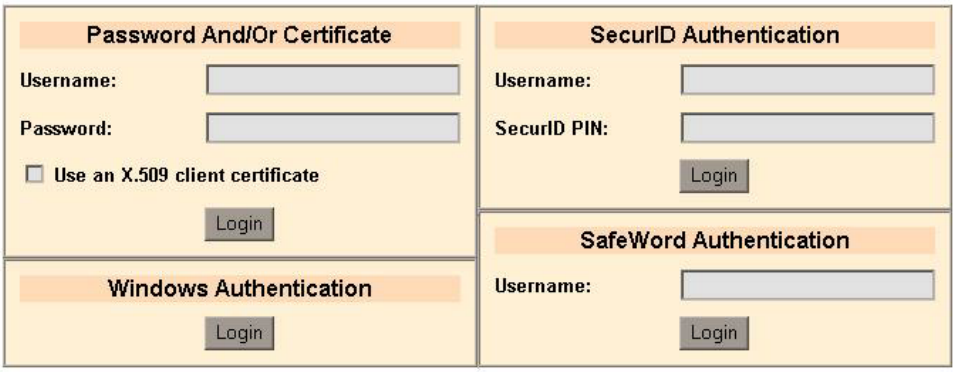
Credentials Selector Introduction

The SiteMinder Credentials Selector is one of SiteMinder's strong authentication solutions. The Credentials Selector enables users to choose the type of authentication credentials necessary to access protected resources. Based on the user's authentication context, the Policy Server can make authorization decisions and then generate user responses in the same single sign-on environment.

The Credentials Selector functionality is implemented as a standalone component, which can be used by any SiteMinder-protected application.

Credentials Selector Use Case

In this use case, the user is given a choice of different credentials to obtain different levels of access when he requests access to a protected resource. When the user requests a protected sample application, he is presented with the following login dialog:



Each login button on the dialog submits different credentials. The user's experience depends on the type of credentials he provides. The user can choose from the following types of authentication:

- Password And/Or Certificate authentication
- Windows authentication
- SecureID authentication
- SafeWord authentication

After the user is successfully authenticated and authorized, the user is permitted access to the sample application, which displays a greeting that informs him of his authentication level and the type of authentication scheme he used to log in.

More Information

Request Access with Password And/Or Certificate Authentication (see page 434)
Request Access with Windows Authentication (see page 435)
Request Access with SecurID Authentication (see page 435)
Request Access with SafeWord Authentication (see page 436)

Request Access with Password And/Or Certificate Authentication

In the Password And/Or Certificate section of the login dialog, the user can choose one of the following combinations of credentials to provide:

- Username and Password only entered in an HTML form
- X.509 client certificate only
- Username/Password and X.509 client certificate

If the user provides only his valid username and password, the following message is displayed:

```
Greetings, SampleUser!
Your authentication level is 5
You have used username/password authentication
```

If the user selects only the X.509 client certificate check box, he is prompted to select one of the client certificates configured with the browser. If it is recognized by the Policy Server, the following message is displayed:

```
Greetings, SampleUser!
Your authentication level is 10.
You have used X.509 client certificate authentication
```

The Password And/Or Certificate option offers the flexibility of providing a different authentication level depending on the credentials the user provides. SiteMinder's X.509 Cert Or Form authentication scheme, which may seem similar to the Password And/Or Certificate option, does not distinguish between the types of provided credentials and therefore, the protection level is the same regardless of what the user provides.

If both Username and Password are provided and the X.509 client certificate check box is marked, the user is prompted for a client certificate. If the certificate is recognized by the Policy Server, and if it matches the username provided, the following message is displayed:

```
Greetings, SampleUser!
Your authentication level is 15
You have used X.509 client certificate and username/password authentication
```

Request Access with Windows Authentication

If the user is currently logged into a Windows domain, the message he sees when he requests the protected resource is:

```
Greetings, SampleUser!
Your authentication level is 5
You have used the Windows domain authentication
```

If he is not logged into a Windows domain, the user is prompted for his Windows domain credentials.

Request Access with SecurID Authentication

If the user provides a valid Username and SecurID PIN for SecurID authentication, the following message is displayed when he requests the protected resource:

```
Greetings, SampleUser!
Your authentication level is 20
You have used the SecurID authentication
```

Request Access with SafeWord Authentication

If the user provides only his username for SafeWord authentication, a two-step process occurs. SiteMinder passes the username to the SafeWord server, and the server determines the credentials for which it will challenge the user. SafeWord supports up to four authenticators per login. The authenticators can be fixed (using a password) or dynamic (using a token card pin).

Upon successful access, the following message is displayed:

```
Greetings, SampleUser!
Your authentication level is 20
You have used the SafeWord authentication
```

Credentials Selector Solution for the Use Case

To set up the Credentials Selector for this use case, you configure the following components:

- The Forms Credential Collector (FCC) used by the front-end authentication scheme to render the login dialog that is presented to the user when he requests the protected sample application. A sample FCC called selectlogin.fcc is supplied with the SiteMinder Web Agent installation.
- In the Administrative UI:
  - A front-end authentication scheme that is exposed to applications that will use the Credentials Selector.
  - Several back-end authentication schemes, one for each type of credentials that the user might choose. Back-end processing refers to functions only the Credentials Selector interacts with. The end-user is not aware of these functions.
  - A specially configured policy domain, which includes a realm, rule, responses, and a policy, that represents the Credential Selector's back-end processing.
- A Web Agent that serves as the entry point for the policy domain representing the back-end processing.
- A sample application that uses the front-end authentication scheme. For this use case, the sample application presents a greeting message to the user. This message changes depending on the credentials the user chooses when logging in.

Establish a Front-End Authentication Scheme

The Forms Credential Collector (FCC), selectlogin.fcc, is used by the front-end authentication scheme to generate the login selection screen used to request access to the protected resource. The FCC dynamically constructs the FCC directives for the Web Agent so the Agent can redirect the user as appropriate for any of the authentication scheme choices.

Be aware that the selectlogin.fcc is a sample for use by the Credentials Selector. The set of authentication choices and HTML formatting depends upon your particular situation.

Note: For detailed information about FCCs, see the Authentication Schemes chapter in this guide or the *Web Agent Configuration Guide*.

Use a Forms Credential Collector (FCC)

The FCC format is a proprietary format used by SiteMinder Web Agents to collect credentials and pass them to the Policy Server. An FCC consists of a header and a body.

FCC Header Description

An FCC header is a list of FCC directives, one per line. The FCC directives have the following syntax:

```
@<directive>[=<value>]
```

The values may contain % substitutions, formatted as %*parameter*%. The parameters are passed with the FCC file on a POST action when the credentials are submitted.

There is a limited set of FCC directives. For the purposes of this example, the most important directives are:

@target

Resource URL that the Web Agent must pass to the Policy Server

@username

Username that the Web Agent must pass to the Policy Server

@password

Password that the Web Agent must pass to the Policy Server

@smagentname

Agent name that the Web Agent must pass to the Policy Server. If the EncryptAgentName parameter in the Agent's configuration is set to yes, the name is encrypted.

Not all FCC directives need to be listed in the header; many have implicit defaults, such as:

- @target=%target%
- @username=%username%
- @password=%password%
- @smagentname=%smagentname%

FCC Body Description

The FCC body contains HTML or other web-browser readable format. It is rendered in the web browser when the user is challenged.

The body may contain substitutions, formatted as $$parameter$$. The parameter name must belong to a certain set of known parameters that are passed with the FCC file on a GET action.

For the purposes of this example, the important directives are:

$$target$$

Resource URL that the user has requested

$$smagentname$$

Web Agent's name. If the EncryptAgentName parameter in the Agent's configuration is set to yes, the name will be encrypted

Configure the selectlogin.fcc File for Front-End Authentication

The selectlogin.fcc file is included with the Web Agent installation as a sample FCC. This file is used by the front-end authentication scheme to render the login dialog that is presented to the user when he requests the protected resource.

When a user requests a resource, the Web Agent passes the requested URL to the Policy Server. In most cases, the resource URL that the Web Agent passes is the same one that the user requests. The FCC files defined for the SiteMinder authentication schemes ensure that the requested URL is sent by passing $$target$$ (the GET parameter) as %target% (the POST parameter), and using the @target=%target% directive , as follows:

```
<!-- some HTML code -->
<form name="Login" method="POST">
<!-- some HTML code -->
<input type="hidden" name="target" value="$$target$$">
<!-- more HTML code -->
</form>
<!-- more HTML code -->
```

Note: The @target=%target% directive is used by default, if it is omitted.

In this case, the selectlogin.fcc file works by replacing the value of the %target% parameter with the following value:

```
/path/redirect.ext?authtype=type&target=$$target$$
```

redirect.*ext*

A simple script that redirects to the URL provided in the target parameter. Example values are redirect.asp or redirect.jsp. Alternatively, you can use different redirect script files or different virtual directories that expose the same physical file as long as the redirect script's URLs depend on the credentials provided.

type

A string determined by the user's choice of credentials.

If different redirect URLs are protected by different authentication schemes, the credentials collected by the FCC are processed by the chosen authentication scheme. This is the authentication scheme that establishes the user's session, including the user's authentication level. After the user is authenticated and authorized for the redirect script resource, the user is redirected to the originally requested resource.

Note: If single sign-on is in effect and the user's protection level is equal or higher than the front-end authentication scheme's protection level, then the user's session is validated against the original resource. Whether or not the user is authorized depends on the policy configuration, which may check for the user's authentication context. For example, a minimal protection level or certain conditions may be required to access particular resource.

More Information:

Selectlogin.fcc Configuration Details (see page 440)

Selectlogin.fcc Configuration Details

You can configure various authentication schemes in the selectlogin.fcc file. The following are configuration details for some schemes:

- The cert-and-form scheme requires posting over an SSL connection. If the front-end authentication scheme does not use an SSL connection, the Web Agent cannot POST to the same selectlogin.fcc URL that was obtained on the GET request.

  The following JavaScript code may be used to convert the URL to an SSL URL:

  ```
  arr = document.URL.split("://");
  document.Login.action = "https://" + arr[1];
  ```

- To support the SafeWord's two-step authentication, the username collected from the first challenge form must be remembered by the client side in time for the second challenge. The safeword.fcc file, also included in the Web Agent installation, uses the @smtransient FCC directive to keep the username in a transient cookie. The same directive may be used in the selectlogin.fcc file too, but then the cookie is going to be created even if the user makes a different choice of credentials. A better alternative is to modify the action argument to POST to the safeword.fcc file, as follows:

  ```
  document.Login.action = "safeword.fcc";
  ```

- The Windows authentication scheme does not use an FCC file. It uses a specific pseudo-resource URL, which does not exist on the web server but which is recognized by the SiteMinder Web Agent. To make Windows authentication work, set the *action* argument to this same pseudo-resource URL, as follows:

  ```
  document.Login.action = "/siteminderagent/ntlm/creds.ntc";
  ```

- The SecurID authentication scheme does not use an FCC; however, the Agent can POST the SecurID credentials directly to the selectlogin.fcc file. No modification of the action argument is required.

- The action URL may be hosted by a different web server; however, the other web server must also be protected by a SiteMinder Web Agent so that the SiteMinder-specific resource URLs (FCC, SCC, and NTC) are recognized and properly processed.

  Note: SCC pseudo-resource URLs are used for certificate-only authentication.

Sample selectlogin.fcc File

A simplified version of the selectlogin.fcc file (without the HTML formatting) follows. There are hidden input fields for smquerydata and postpreservationdata; those are necessary for passing the GET and POST parameters, respectively.

The smauthreason parameter holds the reason code provided by the Policy Server together with the authentication challenge.

A sample selectlogin.fcc file follows:

```
@username=%USER%
@smretries=0

<html>
<head>
  <script language="JavaScript">
    function submitForm(form)
    {
      authtype = "none";

      if (form == 1)
      {
        document.Login.USER.value     = document.Login.USER1.value;
        document.Login.PASSWORD.value = document.Login.PASSWORD1.value;

        if (!document.Login.UseCert.checked)
        {
          // username/password only
          authtype = "form";
        }
        else if (document.Login.USER.value == "" &&
                 document.Login.PASSWORD.value == "")
        {
          // certificate only
          authtype = "cert";
        }
        else
        {
          // username/password and certificate
          authtype = "certform";

          // This option requires posting over SSL.
          arr = document.URL.split("://");
          document.Login.action = "https://" + arr[1];
        }
      }
```

```
      else if (form == 2)
      {
        // SecurID authentication
        authtype = "securid";
        document.Login.USER.value     = document.Login.USER2.value;
        document.Login.PASSWORD.value = document.Login.PASSWORD2.value;
      }
      else if (form == 3)
      {
        // SafeWord authentication
        authtype = "safeword";
        document.Login.USER.value     = document.Login.USER3.value;
        document.Login.PASSWORD.value = "";

        // POST to safeword.fcc, for additional processing.
        // NOTE: This forces the web agent to POST to safeword.fcc
        // even if the authentication scheme's URL parameter
        // is set to selectlogin.fcc for redirection purposes.
        document.Login.action = "safeword.fcc";
      }
      else if (form == 4)
      {
        // Authenticate with the current Windows login credentials
        authtype = "windows";
        document.Login.USER.value     = "";
        document.Login.PASSWORD.value = "";

        // POST to creds.ntc (required by the Windows authentication scheme).
        document.Login.action = "/siteminderagent/ntlm/creds.ntc";
      }
      // Generate the target, depending on the user's choice of credentials.
      // This sample uses redirect.asp, but it could also be redirect.jsp,
redirect.pl, etc.
      // This sample uses the following format:
/auth/redirect.asp?authtype=<choice>&target=<original target>
      // Other formats are also possible, e.g.:
/auth-<choice>/redirect.asp?target=<original
         target>
      // The helper realms' resource filters must be defined accordingly (see the tech
note).
```

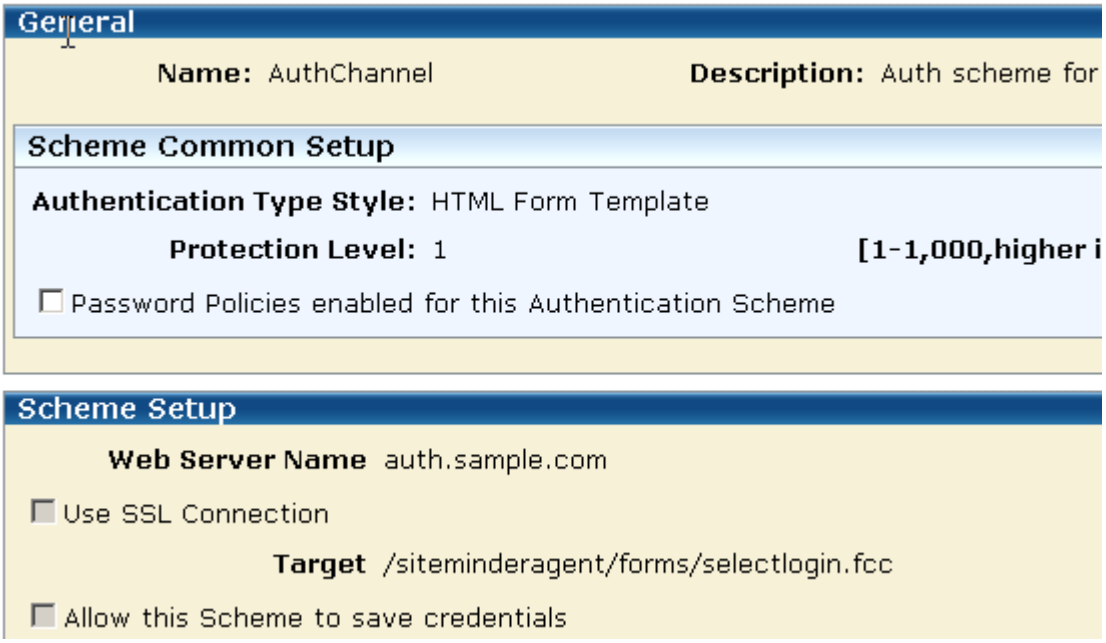
```
      // Check if the target is not already in the same format. The user may
      // have been redirected back to selectlogin.fcc upon authentication failure,
      // if the authentication scheme's URL parameter is set to selectlogin.fcc.
      if ("$$target$$".indexOf("/auth/redirect.asp?authtype=") == 0 &&
          "$$target$$".indexOf("&target=") > 0)
      {
        // This must be a redirect. Extract the original target, but not
        // the authtype parameter, because the user may have made a different
        // choice of credentials this time.
        trgarr = "$$target$$".split("&target=");
        document.Login.target.value = "/auth/redirect.asp?authtype=" + authtype +
"&target=" + trgarr[1];
      }
      else
      {
        // This is not a redirect. Pass $$target$$ as a URL query parameter.
        document.Login.target.value = "/auth/redirect.asp?authtype=" + authtype +
"&target=$$target$$";
      }

      document.Login.submit();
    }
    function resetCredFields()
    {
      document.Login.PASSWORD.value  = "";
      document.Login.PASSWORD1.value = "";
      document.Login.PASSWORD2.value = "";
    }
  </script>
</head>

<body onLoad="resetCredFields();">
  <center>
    <form name="Login" method="POST">
      <input type="hidden" name="USER">
      <input type="hidden" name="PASSWORD">
      <input type="hidden" name="smagentname"  value="$$smagentname$$">
      <input type="hidden" name="smauthreason" value="$$smauthreason$$">
      <input type="hidden" name="smquerydata"  value="$$smquerydata$$">
      <input type="hidden" name="postpreservationdata"
value="$$postpreservationdata$$">
      <input type="hidden" name="target">
```

```
      <!-- Some table formatting throughout -->
        <!-- Authentication Choice: Password And/Or Certificate -->
          <input type="text"     name="USER1">
          <input type="password" name="PASSWORD1">
          <input type="button"   value="Login" onClick="submitForm(1);">

        <!-- Authentication Choice: Windows Authentication -->
          <input type="button"   value="Login" onClick="submitForm(4);">

        <!-- Authentication Choice: SecurID Authentication -->
          <input type="text"     name="USER2">
          <input type="password" name="PASSWORD2">
          <input type="button"   value="Login" onClick="submitForm(2);">

        <!-- Authentication Choice: SafeWord Authentication -->
          <input type="text"     name="USER3">
          <input type="button"   value="Login" onClick="submitForm(3);">
      <!-- More table formatting -->
    </form>
  </center>
</body>
</html>
```

Configure the Front-end Authentication Scheme

You need to configure a front-end authentication scheme that protects the sample application which generates the greeting. For this solution, you can configure the AuthChannel authentication scheme.

See the following illustration for an example of an AuthChannel authentication scheme.



The AuthChannel authentication scheme is set as follows:

Authentication Type Style

HTML Form Template

Protection Level

1

The AuthChannel scheme's Protection Level is set to 1 because the front-end authentication scheme must have a lower protection level than any other scheme in the configuration. The user's actual protection level is determined by the authentication scheme he chooses when logging in to access the protected resource. When the user is redirected back to the originally requested resource, the front-end scheme's low protection level ensures that the user is not re-challenged.

Web Server Name

auth.sample.com

This is the web server where the sample application resides

Target

/siteminderagent/forms/selectlogin.fcc

This target points to the selectlogin.fcc file. The selectlogin.fcc file is a sample file included with the Web Agent installation.

Manage Unsuccessful Authentication Attempts

If the user is rejected by an authentication scheme, the user is redirected to the URL specified in that authentication scheme's Target parameter, if that parameter is available for that scheme.

Configure the following behaviors that best suits your situation:

- Have the user prompted to resubmit his credentials for the same authentication scheme, instead of being presented with a choice of authentication schemes again.
- Allow the user to return to the original login screen to repeat the selection. To do this, the selectlogin.fcc must be configured as each authentication scheme's Target parameter, if applicable.

If a particular choice of credentials requires posting the credentials to an FCC file other than the selectlogin.fcc file, then set the HTML form's action parameter in the selectlogin.fcc to the desired FCC file's URL. For example, this command sets the action parameter to the safeword.fcc file:

```
document.Login.action = "safeword.fcc";
```

If you are using a scheme that does not have a Target parameter, such as the basic authentication scheme, the user experience is the same for a successful authentication. However, if the user has to be re-challenged, the re-challenge is based on basic authentication scheme, that is, with a prompt dialog instead of an HTML form.

Note: The SafeWord basic scheme does not support two-step authentication and multiple authenticators, unlike the SafeWord HTML forms scheme.

Set Up Back-end Processing

To use the Credentials Selector, the following components are required for back-end processing:

- Authentication schemes for each choice of login credentials
- Policy domain that contains the back-end components
- Realms that use each authentication scheme
- Rules for each realm
- Policies that set an authentication context and that authorize resource access

Set Up Authentication Schemes for Back-End Processing

You need to set up several authentication schemes for back-end processing, one for each choice of credentials made available to the user. These schemes enable a user to choose the type of credentials he provides to access a protected resource.

There is one authentication scheme for each type of credentials:

- An HTML Form authentication scheme
- An X.509 Client Cert authentication scheme
- An X.509 Client Cert And Form authentication scheme
- A SecurID HTML Form authentication scheme
- A SafeWord HTML Form authentication scheme
- A Windows authentication scheme

Note: The Basic authentication scheme is a default scheme set up as part of the Policy Server's installation.

Set Up a Policy Domain for Back-End Processing

The Credentials Selector back-end processing is represented by a policy domain. In this solution, the policy domain is named BackendAuth.

Configure Realms for the Back-end Policy Domain

There is one realm for each configured back-end authentication scheme. Each realm needs a resource filter defined as follows:

`/auth/redirect.asp?authtype=`*type*`&target=`

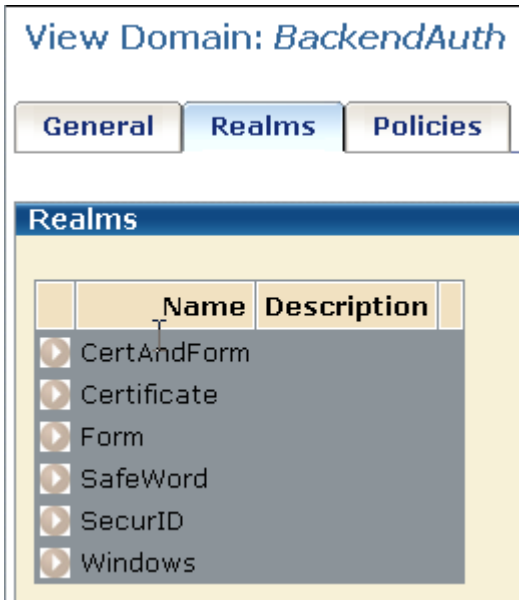
type

Can be one of the following:

form	username/password authentication
cert	certificate authentication
certform	cert-and-form authentication
securid	SecurID authentication
safeword	SafeWord authentication
windows	Windows authentication

Note: These are the types chosen for the purposes of this use case. You are not restricted to these specific values, but the types must correspond to the authtype values in the selectlogin.fcc file, or any other FCC file based on the selectlogin.fcc template. Also, the realm's resource filter must match the redirect target in the FCC file.

The following pane lists the realms.



The Web Agents protecting the realms may or may not be the same. This solution uses a single Web Agent; however, if multiple Web Agents are used, they must satisfy specific requirements.

The following requirements are necessary for Web Agents protecting realms as part of the Credentials Selector functionality:

- Single sign-on must be configured between all Web Agents protecting realms. This means that they are in the same cookie domain or they use the same cookie provider.

  Note: To configure single sign-on, see the *Web Agent Configuration Guide*.

- The value of the @smagentname directive in the FCC file must match the expected value for at least one of the Web Agents protecting the originally requested resource.

  The expected value is the value of the Web Agent's AgentName parameter or the DefaultAgentName parameter, if the AgentName parameter is not assigned a value. If the EncryptAgentName parameter in the Agent's configuration is set to yes, the value must be encrypted.

  One way of setting the @smagentname directive is by configuring each Web Agent with the same naming properties. They can even share the same Agent Configuration Object. Another method is to configure the @smagentname directive programmatically in the FCC file, provided that the name is not encrypted.

  Important! If the @smagentname directive is misconfigured, you may see a "No realm received in request" error message in the Policy Server log.

- The FCCForceIsProtected parameter must be set to yes to ensure that a second IsProtected call is made for the new target generated by the selectlogin.fcc file. The IgnoreQueryData parameter must be set to no so the Web Agent does not ignore the URL query parameters.

Create Rules for the Back-end Policy Domain

In each realm that you configure for the BackendAuth policy domain, there are two rules you need to configure:

- An OnAuthAccept rule
- A Web Agent action rule (for the purposes of this example, use a GET action rule)

Both rules should have an asterisk (*) as the value for the Resource field.

For example, for the Form realm in the BackendAuth policy domain, the rules would be:

OnAuthAccept

Resource: /auth/redirect.asp?authtype=form&target=*

Action: OnAuthAccept

GetRule

Resource: /auth/redirect.asp?authtype=form&target=*

Action: Get

Configure AuthContext Responses for the Back-end Policy Domain

An AuthContext response is configured for each authentication scheme in the BackendAuth domain. Each of these responses contains an AuthContext response attribute, which is evaluated only on an OnAuthAccept event. Its value is added to the SiteMinder session ticket as the value of the SM_AUTHENTICATIONCONTEXT user attribute. It is not, however, returned to the client as a user response.

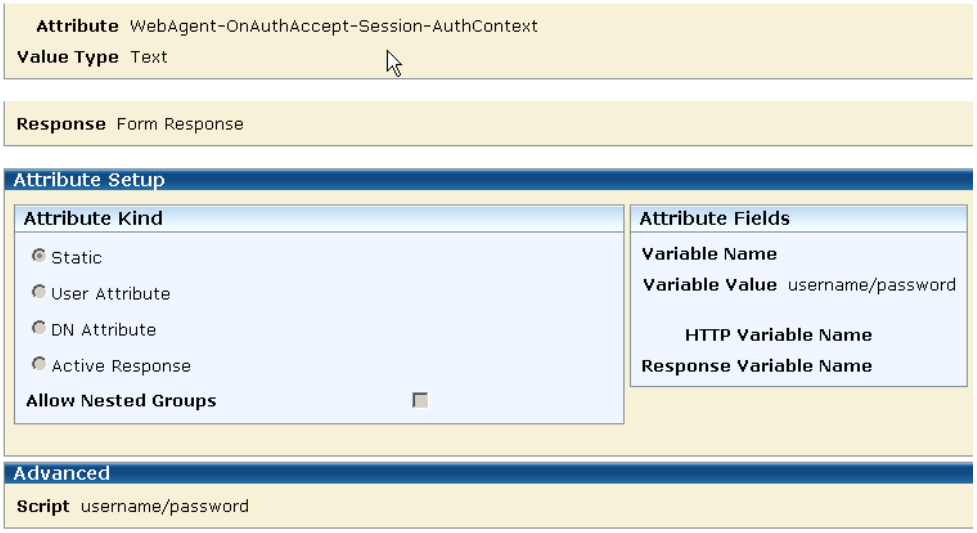
For this example, the list of responses should be:

Name	Agent Type	Description
Form	Web Agent	AuthContext for username/password auth
Certificate	Web Agent	AuthContext for certificate auth
CertandForm	Web Agent	AuthContext for cert and form auth
SecurID	Web Agent	AuthContext for SecurID auth
SafeWord	Web Agent	AuthContext for SafeWord auth
Windows	Web Agent	AuthContext for Windows auth

Note: The response attribute value is truncated to 80 bytes in length.

To configure an AuthContext response attribute, select the WebAgent-OnAuthAccept-Session-AuthContext response attribute type.

The following illustration shows the creation of an AuthContext response attribute using the WebAgent-OnAuthAccept-Session-AuthContext attribute type.



As the illustration shows, the AuthContext response attribute type is static. When legacy federation is in use, you can specify a static attribute to define a constant or literal value for better encapsulation. Constant values include strings.

SiteMinder variables and active expressions add more flexibility to configuring AuthContext response attributes. They may also contain the authentication timestamp and/or a hash value of a SAML assertion.

The following group box shows one of the resulting responses configured for this solution. This is the attribute for the Form response.

Domain BackendAuth

Attribute List

Agent Type Attribute Name	Value
WebAgent-OnAuthAccept-Session-AuthContext	username/password

Configure Policies for Back-end Credential Selection

There are two policies for the BackendAuth domain in this example:

- One for setting the user's authentication context
- One for Web Agent actions

The following illustration shows the two policies.

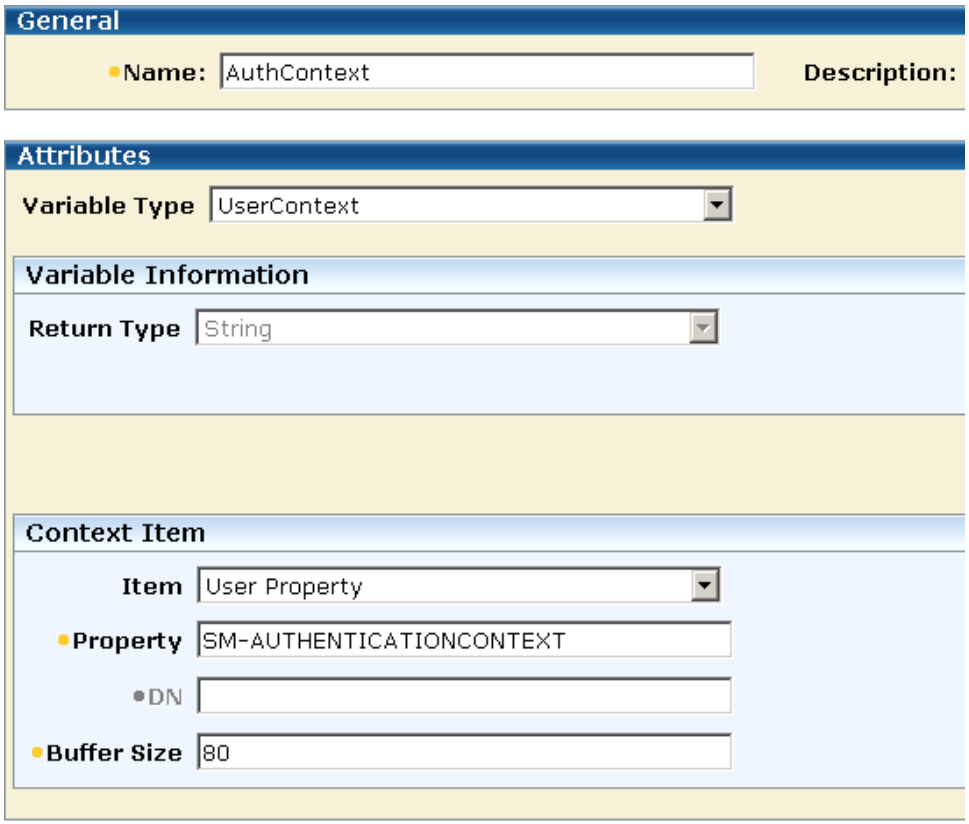
Create an Authentication Context Policy

The AuthContext policy sets the authentication context in the SiteMinder session ticket, which is used for authentication and validation. In this policy, the OnAuthAccept rules from each realm are paired with the corresponding responses to permit access to protected resources. For example, the OnAuthAccept Rule for the Form realm is paired with the Form response and the OnAuthAccept rule for the SafeWord realm paired with the SafeWord response.

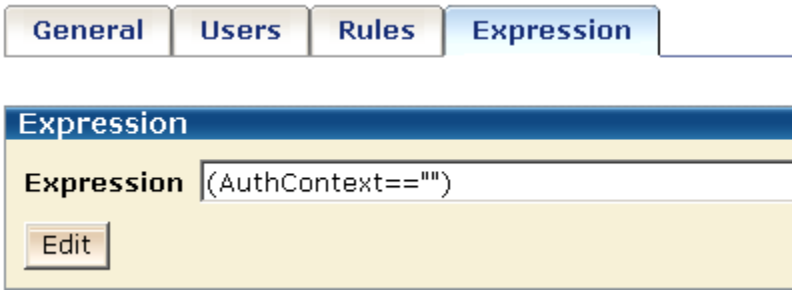
User authentication and user validation are OnAuthAccept events so the authentication context in the session ticket may be overwritten after each validation. The ability to update the authentication context attribute can be useful in some circumstances, for example, if that attribute's value will include a counter. However, in this solution using the Credentials Selector, the AuthContext policy is configured to fire only if the authentication context is empty to ensure that the session ticket is not overwritten, thereby remembering the user's choice of credentials.

You need to protect the authentication context from being overwritten. To do this, write an active expression in the AuthContext policy to retrieve the SM_AUTHENTICATIONCONTEXT attribute from the session ticket.

When legacy federation is in use, you can create a user context variable called AuthContext and use it in the AuthContext policy to define an active expression that retrieves the SM_AUTHENTICATIONCONTEXT attribute from the session ticket.



Define an active expression using the AuthContext variable in the AuthContext policy:



Create a Protection Policy

Authorizing an authenticated user for a requested resource requires a second policy in the BackendAuth domain. This policy protects the resources and is in addition to the authentication context policy in this domain.

The protection policy should authorize the authenticated user for the redirection target, which is a GET action on the redirect.asp file. Name the policy GetPolicy.

The GetPolicy would contain the associated rules:

Rule	Realm
GetRule	Form
GetRule	Certificate
GetRule	CertandForm
GetRule	SecureID
GetRule	SafeWord
GetRule	Windows

Protect the Sample Application

To use the Credentials Selector, a SiteMinder application only has to have its protected realms configured with the component's front-end authentication scheme. The policies protecting the application may have restrictions based on the user's authentication level and authentication context.

In this solution, the sample application generates the greeting message for the authenticated and authorized user.

The file that generates this greeting has the following code:

```
<html>
<head></head>
<body>

<h3>
<p>Greetings, <%=Request.ServerVariables("HTTP_USERNAME") %>!
<p>Your authentication level is <%=Request.ServerVariables("HTTP_AUTHLEVEL") %>
<p>You have used <%=Request.ServerVariables("HTTP_AUTHCONTEXT") %>
authentication
</h3>

</body>
</html>
```

The different authentication options in the login dialog result in different access levels and a different greeting, such as:

```
Greetings, SampleUser!
Your authentication level is 5
You have used username/password authentication


Greetings, SampleUser!
Your authentication level is 10.
You have used X.509 client certificate authentication


Greetings, SampleUser!
Your authentication level is 5
You have used Windows domain authentication
```

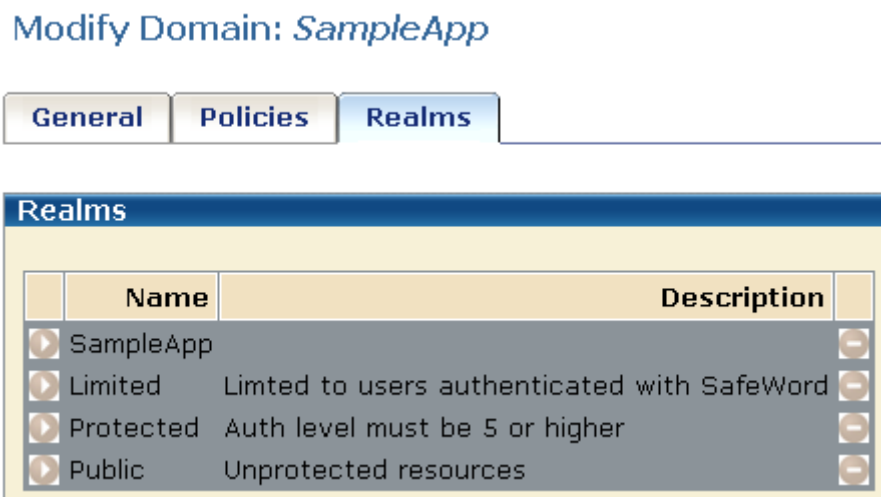
The sample application has four kinds of resources contained in different realms. The realms must each be configured with the front-end authentication scheme.

Realms and Rules for the Sample Application

For this example, the following four realms protect the various resources that make up the sample application:

- A realm that contains public resources
- A realm that contains resources requiring at least a level 5 authentication
- A realm that contains resources requiring at least a level 15 authentication
- A realm that contains resources available only to users authenticated with SafeWord

The realms are shown in the following illustration.



The protected realms are configured with the AuthChannel front-end authentication scheme, as shown in the following illustration.

The SampleAgentGroup may contain any Web Agents that provide points of entry to the sample application.

The following requirements are necessary for Web Agents protecting realms as part of the Credentials Selector functionality:

- Single sign-on must be configured between all Web Agents protecting realms. This means that they are in the same cookie domain or they use the same cookie provider.

  Note: To configure single sign-on, see the *Web Agent Configuration Guide*.

- The value of the @smagentname directive in the FCC file must match the expected value for at least one of the Web Agents protecting the originally requested resource.

  The expected value is the value of the Web Agent's AgentName parameter or the DefaultAgentName parameter, if the AgentName parameter is not assigned a value. If the EncryptAgentName parameter in the Agent's configuration is set to yes, the value must be encrypted.

  One way of setting the @smagentname directive is by configuring each Web Agent with the same naming properties. They can even share the same Agent Configuration Object. Another method is to configure the @smagentname directive programmatically in the FCC file, provided that the name is not encrypted.

  Important! If the @smagentname directive is misconfigured, you may see a "No realm received in request" error message in the Policy Server log.

- The FCCForceIsProtected parameter must be set to yes to ensure that a second IsProtected call is made for the new target generated by the selectlogin.fcc file. The IgnoreQueryData parameter must be set to no so the Web Agent does not ignore the URL query parameters.

Rules for the Policy Protecting Sample Application

You can configure any type of rule for the realms that contain the sample application resources.

Configure a Policy to Protect the Sample Application

The GetProtected policy requires a protection level of 5 or greater for access to protected resources. To enforce this protection level restriction, you can write an active expression in the GetProtected policy to retrieve the SM_AUTHENTICATIONLEVEL attribute from the SiteMinder session ticket.

Note: This authentication level restriction is designed to protect applications from custom Web Agents that only support password authentication levels of one.

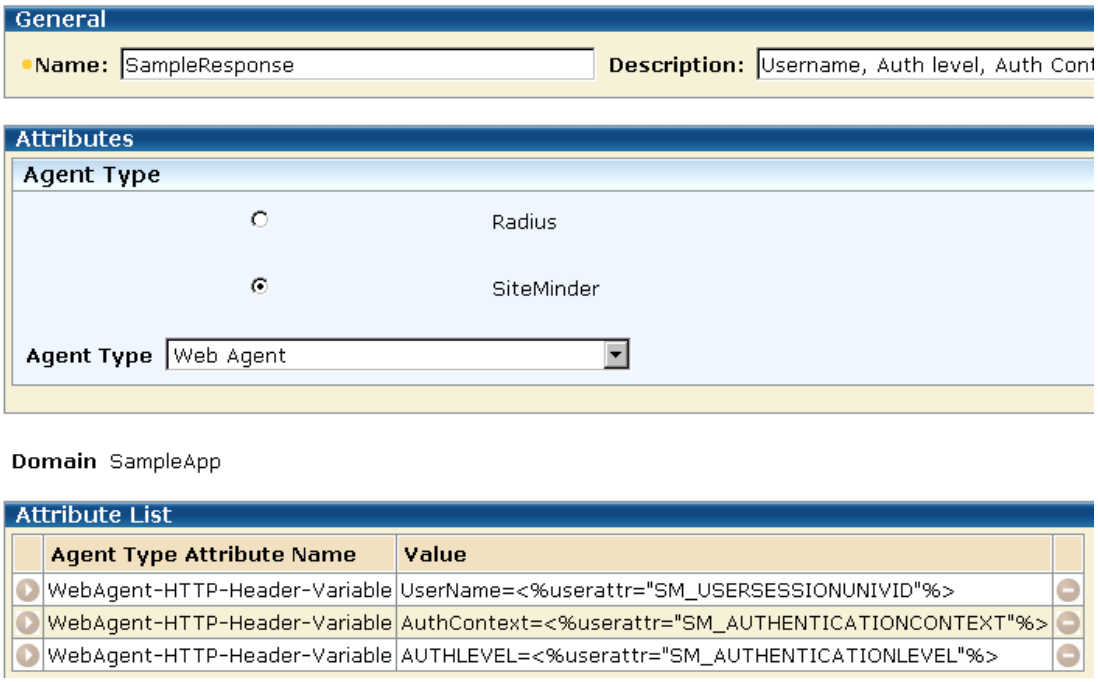
When legacy federation is in use, you can create a user context variable called AuthLevel and use it in the GetProtected policy to define an active expression that retrieves the SM_AUTHENTICATIONLEVEL attribute from the session ticket.

Configure Responses for the Sample Application

In this example, the sample application that displays the greeting message uses three HTTP header variables:

- HTTP_USERNAME
- HTTP_AUTHLEVEL
- HTTP_AUTHCONTEXT

These headers are returned to the client with the following response:



Attribute values are specified on the Response Attribute pane.

Test the Credentials Selector Solution

If you have set up the various components described in this use case, you can test the Credentials Selector functionality. This test should show that you see different greetings depending on the credentials you specify.

To test the Credentials Selector

1. Try to access the sample application, for example, by clicking on a link to it.

   You should be presented with the login screen defined by the selectlogin.fcc file.

2. Enter one type of credential to login.

   You should see the greeting appropriate for the credentials you entered. For example, if you entered a username and SecurID PIN, you should see a greeting

   ```
   Greetings, SampleUser!
   Your authentication level is 20
   You have used the SecurID authentication
   ```

3. Exit the application and try accessing the sample application again.
4. Enter a different set of credentials than you did in the previous step.

   You should see a greeting message appropriate for the specified credentials.

You have successfully tested the Credentials Selector.

Chapter 14: Securing Resources Using EPM Application Objects

This section contains the following topics:

Secure Applications Using Enterprise Policy Management

Enterprise Policy Management (EPM) is an access management model that lets you protect business applications without an in-depth knowledge of SiteMinder-specific concepts and components.

EPM presents policy configuration in the context of securing an application. To protect an application, you create an *Application* object and are only required to provide data for configuration settings that do not have defaults. Modifying other settings is optional. EPM therefore makes policy configuration more straightforward. You can manipulate additional SiteMinder settings that allow you to define more fine-grained protection of an application; however, such manipulation is not required.

For the administrator already familiar with SiteMinder domain-based policies, there is a relationship between the application-oriented concepts and the underlying SiteMinder policy objects. This relationship is reflected in the Administrative UI and is shown in the following table:

Application Dialogs and Group Boxes	Underlying SiteMinder Component
General settings	Defines the policy domain
Components	Defines the realm
Resource	Specifies the rule
Application Roles	Define the policy users

Application roles define the set of users who have access to a resource or group of resources defined in an Application object. Roles can include all users in configured user directories, be limited to selected groups, organizations, and users with matching user attributes, or specified using a named or unnamed expression.

EPM offers the following benefits:

- Application-centric approach

  The focus on applications relates closely to the view of access management by most businesses.

- Consistent security enforcement model

  The security enforcement model for EPM is no different than implemented by the more SiteMinder-centric model. However, the SiteMinder-specific components are hidden from configuration.

- Simplified security

  Securing resources is simplified—you name the application, the application resources that need protecting, and the application roles that are permitted access. You are not required to examine or modify every aspect of a component to establish a security policy.

- Enhanced delegation

  A SiteMinder administrator can grant access to an application without expert knowledge of SiteMinder. This ability enables a senior security administrator to delegate access management responsibilities to other administrators.

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. Define an application that you want to protect.
2. Create a new user directory or associate an existing user directory with the application.
3. Specify the resources (such as the parts of an application or web pages) that you want to protect.
4. Create application roles that identify the users that should have access to the protected resources.
5. (Optional) Configure responses to customize an application.
6. (Optional) Specify custom attributes to provide metadata about the application.

7. Repeat Steps 4 and 5 until all your resources and roles are defined.
8. Create policies by associating the application roles with the resources.

A series of use cases (see page 464) show the detailed configuration steps for protecting applications.

Note: You may want to have the Administrative UI open to follow along with the use cases.

Administrative Rights to Create Application Security Policies

SiteMinder 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.

Follow these steps:

1. Click Administration, Administrator.
2. Objects related to administrators appear on the left.
3. Click Administrators.

   The Administrators screen appears.
4. Click Create Administrator.

   The Create Administrator screen appears.
5. Enter a name for the administrator in the Name field.
6. Click Create in the Rights area.

   The Rights dialog appears.

7. In the table, choose one or more of the following:
   - application administration
   - policy administration
8. Click OK.

The administrator has been given application and policy administration privileges.

Use Cases for Defining Application Security Policies Using Application Objects

Learn how to define application security policies using application objects by reviewing the following use cases:

- Application Security Policy to Protect a Web Portal (see page 464)
- Application Security Policies Based on Roles (see page 469)
- Application Security Policies with User Mapping and Named Expressions (see page 476)
- Application Security Policy Based on CA DLP Content Classifications (see page 484)

Application Security Policy to Protect a Web Portal

In this use case, a software company, sample-software-company.com, has a web portal that provides information about the company and its products to the public.

Anyone can access the main home page and product information pages, such as promotional materials and white papers without restrictions. This area of the web portal does not require any security policy. Access to the software downloads area; however, is restricted to registered customers. Each customer is assigned a user name and password which is stored in an LDAP directory server.

The following use case shows how an application security policy protects the restricted software downloads area so that only registered customers have access.

Given:

- The SiteMinder environment contains a single LDAP directory server that stores the user names and passwords for all of the registered customers.
- Customers must authenticate with a user name and password before they can access the software downloads area.

Solution:

To solve this use case, use the following process:

1. Identify the web portal that needs protecting and select the directory containing the customer information.
2. Create separate resources for the software download area of the portal.
3. Create a registered customers role.
4. Associate the resources with the registered customers role to create an application security policy.

Identify the Web Portal and Select the User Directory

An application security policy for a web portal 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.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

To identify the web portal and select the directory server

1. Click Policies, Application.
2. Click Applications.

   The Applications page appears.
3. Click Create Application.

   The Create Application page appears.
4. Enter the name and description of the application. Provide 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

   Sample Software Company Portal

   Description

   Allows access to all parts of the portal except the downloads area.

5. In Components, provide a name for the component and specify the directory that contains the resources you want to protect. For this web portal use case, use the following example:

   Component Name

   Downloads

   Agent Type

   Web Agent

   Agent

   PortalAgent

   Resource Filter

   /downloads

   Note: A subcomponent can be created only after you save the main component.

6. Accept the defaults for the remaining settings.
7. In User Directories, click Add/Remove.

   The Choose User Directories page appears.

8. Select the directory that contains the relevant users then click the right arrow to move the directory from the Available members column to the Selected Members column.
9. Click OK.

   You return to the General tab.

10. Click Submit.

    The web portal application is identified and the directory selected.

Create the Web Portal Resources

After the location of the resources and the user directory have been specified, the individual resources in the subdirectories of the web portal that you want to protect must be specified.

To create the web portal resources

1. Click the Resources tab.

   A list of resources appears.

2. Click Create.

   The Create Application Resource pane opens.

3. Enter values for the fields in the General group box. Select distinctive values that help you remember its purpose or function, as shown in the following examples:

   Name

   Downloads Area

   Description

   Software downloads restricted to registered customers

   Resource

   *

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

4. Verify that the Effective Resource matches what you want to protect. For this use case, the effective resource is:

   Effective Resource

   /downloads/*

   This string specifies that all resources in the downloads directory are protected.

5. Verify that the Web Agent actions radio button in the action group box is selected, and then click the following items in the Action list:

   - Get
   - Post

6. Click OK.

   The web portal resource is created and appears in the resources list.

Create the Web Portal Roles

After the web portal resources have been specified, create a role for the registered customers of the web portal. A role associates resources with groups of users.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

To create web portal roles

1. Click the Roles tab.
2. Click Create Role.
3. Verify that the Create a new object of type Role button is selected, and then click OK.

   The Create Role pane opens.

4. 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

   Registered Customers

   Description

   Registered customers permitted to access software downloads.

   Role applies to

   All Users

   Note: The Users Setup and Advanced group boxes do not apply when this option is set and are no longer displayed.

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

5. Click OK.

   The registered customers role is created.

Create the Web Portal Policy

After the resources and roles have been created, you must associate the resources of the web portal you want to protect with the roles of the users who will access the resources in the web portal. This creates the policies that protect your applications.

To create the policy to protect the web portal

1. Click the Policies tab.
2. Select the Registered Customers role in the Software Downloads row.

   By selecting this role, you indicate that only registered customers have access to the software downloads area of the web portal.

3. Click Submit.

   A confirmation screen appears. The application security policy for the web portal is created.

Displaying the List of Resources

You can sort how list of resources is displayed by clicking one of the following radio buttons:

Name

Sorts resources according to the name you provided when you specified the resource.

Example: Software Downloads

Filter

Sorts resources according to the actual resource that is being protected.

Example: * (asterisk indicates all resources)

Application Security Policies Based on Roles

In this use case, a financial services company, acme-financial.com, has an internal human resources application that handles benefits and performance management. All employees should have access to the benefits portion of the application while only managers should be permitted access to the performance management portion.

The following procedures detail how you can use the EPM model together with application roles to create a security policy for the human resources application.

Given:

- The SiteMinder environment contains one user LDAP directory, called AcmeLDAP.
- The user directory identifies all employees and employees who are managers. They are defined in the directory as follows:
  - group:cn=employees,ou=Groups,o=acme-financial.com
  - group:cn=managers,ou=Groups,o=acme-financial.com
- Employees, including managers, must authenticate with the Basic authentication scheme

Solution for application security based on roles:

To solve this use case, you complete the following steps:

1. Create an application.
2. Select the user directory where you locate the users that meet the role criteria.
3. Specify the resources that are the sub-components of the main application.
4. Define the two roles that should have access to the application.
5. Combine the resources and roles into an application policy.

Identify the Application that Needs Protecting

In this use case, you need to establish different access privileges for different parts of the human resources application. To do this, you must identify the directories underneath the main application and configure the appropriate access.

To protect the example human resources application

1. Click Policies, Application.
2. Click Applications.

   The Applications page appears.
3. Click Create Application.

   The Create Application page appears.
4. Click the General tab.
5. Enter values for the following fields in General. For this use case, the following data is specified:

   Name

   HR Application

   Description

   Identifies the internal human resources application
6. Enter values for the following fields in Components. For this use case, the following data is specified:

   Component Name

   Benefits

   Agent Type

   Web Agent

   Agent

   hrportal agent

   Resource Filter

   /benefits

Default Resource Protection

Protected

Authentication Scheme

Basic

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

Note: A subcomponent can be created only after you save the main component.

7. Specify the user directory associated with the resources being protected. This directory is where SiteMinder will locate users who meet the role criteria.
   a. Click Add/Remove.
   b. Select Employees from the Available Members box and click the right arrow to place this group in the Selected Members box.
   c. Click OK.

The human resources application is now identified.

Designate the Application Resources

After specifying the sub-areas of the main application that you want to protect, you can then designate the specific resources within that subdirectory that you want to protect with an application policy.

For this use case, there are two resources to protect:

- benefits management
- performance appraisals

To specify the specific resources or functions of the main application

1. Click the Resources tab.
2. Click Create.

   The Resource pane opens.
3. Enter values for the fields in the General group box. For this use case, enter the following:

   Name

   Benefits Management

   Description

   Lets employees manage their benefits

4. Enter values for the fields in the Attributes group box. For this use case, enter the following:

   Resource

   managebenefits.jsp

5. Repeat steps 2–4, but enter the following information:

   Name

   Performance Appraisals

   Description

   Lets a manager write an appraisal report and salary review for an employee

   Resource

   salaryincrease.jsp

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

The resources associated with the performance management application are now defined.

Create an Employee Role

After defining the specific components of an application that require protection, you specify roles that define the set of users who have access to a particular resource. Create a role for all employees.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

To create a role

1. Click the Roles tab.
2. Click Create Role.

   The Create Role pane appears.

3. Verify that the Create option is selected, and click OK.

4. Enter values for the fields in the General group box. For this use case, enter the following:

   Name

   Employees

   Description

   All employees of Acme Financial Services

   Role applies to

   All Users

   Note: The Users Setup and Advanced group boxes do not apply when this option is set and are no longer displayed.

5. Click OK.

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

Create a Manager Role

After defining the specific components of an application that require protection, specify roles that define the set of users who have access to a particular resource. Create a role for managers.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

To create a role

1. Click the Roles tab.
2. Click Create.

   The Create Role pane appears.

3. Verify that the Create option is selected, and click OK.
4. Enter values for the fields in the General group box. For this use case, enter the following:

   Name

   Managers

   Description

   Managers at Acme Financial Services

Role applies to

Selected Users

5. Verify that the Role Applies to Selected Users option is selected and that the Users Setup and Advanced group boxes are visible.
6. Define the set of users in the Managers role by making selections in the Users Setup group box. For this use case, select the following entry in the Member Groups table:

   `cn=managers,ou=Groups,o=acme-financial.com`

   This entry specifies the managers group in the corporate user directory.
7. Click OK.

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

Use a Response to Supply Data to the Application

To make the human resources application more user friendly for employees of Acme Financial Services, you can configure a response that provides the employees ID on their benefit records.

To create a response that provides the employee ID:

1. Click on the Response tab from the Application dialog.
2. Click Create Response.

   The Create Response dialog opens.
3. Complete the field as follows:

   Name

   Employee ID

   Description

   Lists the employee ID.
4. Click Create Response Attribute.

   The Create Response Attribute dialog opens.
5. Complete the fields as follows:

   Attribute

   WebAgent-HTTP-Header-Variable

   Attribute Kind

   User Attribute

Attribute Fields—Variable Name

Personnel_Key

Attribute Fields—Variable Value

EmployeeID

Note: Complete descriptions of response attributes exist in the Web Agent Configuration Guide.

6. Keep the defaults for all the other fields.
7. Click OK until you return to the main Response tab.

The response named Employee ID has been created. When an employee views her benefits information, the data from this response is returned to the human resources application and her customer ID will be displayed in the benefits record.

Establish a Policy Based on Roles

After you have defined the resources and roles, you can group these objects into application security policies.

To create the application security policies

1. Click the Policies tab.

   The Policies pane opens and displays a table listing the configured resources and roles. This table lets you quickly see which roles can be granted access to which resources.

2. Do the following:
   a. Check the Employees role in the Benefits Management row to create a policy that allows all employees to manage their benefits.
   b. Check the Managers role in the Performance Appraisals row to create a policy that allows only managers to access the performance appraisals.
3. Click Submit.

You have created two security policies for the human resources application based on roles.

Note: If you need to edit resources or roles, you must make the changes on the respective tabs and not on the Policies pane.

Include Metadata that Describes the Application

Acme-financial.com wants to ensure that there is some descriptive information about the internal human resources application. Custom attributes can be used to define metadata that describes the application.

The information that Acme-financial wants for the purpose of the application and the date the application was completed.

Follow these steps:

1. Click the Custom Attributes tab.

   The Custom Attributes dialog opens.

2. Click Create.

   A table appears with Name and Value fields.

3. Enter values for the fields in the custom attributes table. For this use case, enter the following:

   Name

   App_Completed

   Value

   November_22_2007

4. Click Create to add another row to the table then enter the following:

   Name

   Purpose

   Value

   Human_Resource_Mgmt

5. Click Submit.

Application Security Policies with User Mapping and Named Expressions

In this use case, a retail clothing company wants to define a role preventing customers from making web-based credit purchases if they have 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.

You can create an application security policy using attribute mapping, named expressions (virtual user attributes and user classes) and roles to satisfy the corporate credit policy.

Given:

- The SiteMinder environment contains two user directories.
- Directory A stores employees. Employees can also be customers. Therefore, Directory A identifies employees that are also customers that belong to:

  `group:cn=Customers,ou=Groups,o=acme.com`
- 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 the user is a customer.

Solution:

1. Define an attribute mapping.
2. Establish a named expression.
3. Use the attribute mapping in an advanced expression to establish roles.
4. Create a response to customize the application further.
5. Create an application security policy.

More information:

Named Expressions (see page 244)

Establish Mappings for the Two User Directories

The retail company maintains two directories. To create a universal schema that identifies customers in both user directories use attribute mappings, which you create in the Administrative UI.

To create attribute mappings for this use case

1. Create a group membership attribute for Directory A:
   - Name the attribute **IsCustomer**.
   - Define IsCustomer as: cn=Customers,ou=Groups,o=acme.com
2. Create a constant attribute for Directory B:
   - Name the attribute **IsCustomer**.
   - Define IsCustomer as **"TRUE"**.

IsCustomer results in a common view of the same user information. You can reference IsCustomer in an expression to determine whether a user is a customer.

Review the section Define Attribute Mappings (see page 261) for detailed procedures on how to configure attribute mappings.

Define Named Expressions to Check the Credit Limit

Named expressions enable SiteMinder to calculate each users credit limit and account balances. An expression can also determine if customers are over their credit limit.

To define named expressions for this use case

1. Define a virtual user attribute that calculates a $1,000 dollar credit limit for customers and a $2,000 credit limit for employees:
   - Name the attribute **#CreditLimit**.
   - Define #CreditLimit as:

     ```
     IsCustomer?1000:2000
     ```

     This calculation contains SiteMinder supported expression syntax.
2. Define a virtual user attribute that retrieves account balances from the accounting database:
   - Name the attribute **#Balance**
   - Define the attribute as:

     ```
     (MyLibrary.GetBalance(""))
     ```

     This attribute definition is an active expression defined by the clothing retailer.
3. Create a user class expression that determines if customers are over their credit limit:
   - Name the attribute **@IsUnderCreditLimit**
   - Define the attribute as:

     ```
     (#Balance > #CreditLimit)
     ```

Read Define Named Expressions (see page 245) for details on creating virtual user attributes and user class expressions.

Protect the Online Shopping Application

In this use case, you want to establish access privileges with specific conditions for the store's Web-based shopping application.

To protect the web-based shopping application

1. Click Policies, Application.
2. Click Applications.

   The Applications page appears.
3. Click Create Application.

   The Create Application page appears.

4. Click the General tab.
5. Enter values for the following fields. For this use case, the following data is specified:

   Name

   Online Catalog

   Description

   Identifies the clothing stores Web-based shopping application
6. Enter values for the following fields in Components. For this use case, the following data is specified:

   Component Name

   Catalog

   Agent Type

   Web Agent

   Agent

   Web Retail Agent

   Resource Filter

   /webcatalog

   Default Resource Protection

   Protected

   Authentication Scheme

   Basic

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

   Note: A subcomponent can be created only after you save the main component.
7. Specify the user directory associated with the resources being protected. This directory is where SiteMinder will locate users who meet the role criteria.
   a. Click Add/Remove.
   b. Select IsCustomer from the Available Members box and click the right arrow to place this group in the Selected Members box.

      IsCustomers maps to the users in both directories associated with the clothing store.
   c. Click Submit.

You have now created an application called Online Catalog.

Designate the Resource Requiring Protection

For this use case, you want to protect the checkout process so that users who exceed their credit limit cannot complete the transaction. Therefore, you need to add a resource to the Online Catalog application you just created.

To protect the specific resource of the web-based shopping application

1. Click Policies, Application.
2. Click Applications.

   The Applications page appears.
3. Specify search criteria and click Search.

   The Applications matching the criteria appear.
4. Click the name of the application you want to modify. For this use case, click Online Catalog.

   The View Application page appears.
5. Scroll down the page and click Modify.

   The settings and controls become active.
6. Select the Resources tab.
7. Click Create.

   The Create Resource page appears.
8. Enter values for the following fields. For this use case, enter the following:

   Name

   Checkout

   Description

   Lets you total your purchases and pay for them.
9. Enter values for the following fields in Attributes. For this use case, enter the following:

   Resource

   total_charges.jsp
10. Select Web Agent actions in Action and select the actions Get and Post.
11. Click OK.

You have created a resource called Checkout.

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

Configure the Customer Role

After the web portal resource is defined, create an application role that lets customers make web-based purchases as long as they have not exceeded their credit limit.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

To create this credit-based role

1. Click the Roles tab.
2. Click Create Role.

   The Create Role dialog appears.
3. Verify that the Create option is selected, and click OK.

   The Create Role dialog opens.
4. Enter values for the fields in the General group box. For this use case, enter the following:

   Name

   PurchasewithCredit

   Description

   Indicates that the customer uses credit to pay for their purchases.

   Role applies to

   Selected users
5. Enter an expression in the Advanced Expression group box. For this use case, enter the following:

   User Expression

   @IsUnderCreditLimit

   The role expression is the product of the two virtual user attribute expressions #Balance and #CreditLimit, which calculate whether the user has exceeded their credit limit.
6. Click OK.

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

You have created a role named PurchasewithCredit, whose value is the combination of two named expressions.

Customize the Application with a Response

To provide a more personalized experience for the customer, the retail clothing company can configure a response that lets customers who are over their credit limit apply for increased credit. If a customer has exceeded their credit limit, this response will redirect them to a credit application where they can apply for a higher credit limit.

To create a response

1. Click on the Response tab.
2. Click Create Response.

   The Create Response dialog opens.
3. Complete the field as follows:

   Name

   CreditNotice

   Description

   Alerts users they have exceeded credit limit.
4. Click Create Response Attribute.

   The Create Response Attribute dialog opens.
5. Complete the fields and settings as follows:

   Attribute

   WebAgent-OnReject-Redirect

   Attribute Kind

   Static

   Attribute Fields—Variable Value

   http://catalog.retailcorp.com/credit_notice.jsp

   Note: Complete descriptions of response attributes exist in the Web Agent Configuration Guide.
6. Keep the defaults for all the other fields.
7. Click OK.

The response named CreditNotice has been created and will be sent to customers who exceed their credit limit.

Configure the Security Policy for the Shopping Application

After you have defined the resource, role, and response, configure the policy that secures the Web-based shopping application.

Follow these steps:

1. Click the Policies tab.

   The Policies dialog opens and displays a table listing the Checkout resource and the PurchaseWithCredit role displayed.

2. Select the PurchaseWithCredit role for the Checkout resource.

   This pairing establishes a policy that lets all customers make a purchase with the store's credit card, if they have not exceeded their credit limit. Additionally, by checking the role the Responses grid becomes populated.

3. Select the CreditNotice response for the Checkout resource.

You now have a security policies for the online catalog application based on roles that define a spending limit. Additionally, a response is associated with the policy and will be sent to those customers who continue to make purchases after exceeding their limit.

Provide Metadata to Describe the Application

The retail clothing company wants to ensure that there is some descriptive information about the online catalog application. Custom attributes can be used to provide metadata that describes the application.

The retail clothing company wants to note that the application is only for the online catalog and the email address of the administrator of this application.

To specify metadata for the online catalog application:

1. Click the Custom Attributes tab.

   The Custom Attributes dialog opens.

2. Click Create.

   A table appears with Name and Value fields.

3. Enter values for the fields in the custom attributes table. For this use case, enter the following:

   Name

   App_Function

   Value

   online_retail

4. Click Create to add another row to the table then enter the following:

   Name

   Admin_email

   Value

   jdoe@retailcorp.com

5. Click Submit.

You have completed all the available tasks related to creating an application security policy.

Application Security Policy Based on CA DLP Content Classifications

In this use case, a financial services company, Forward Inc., has deployed Microsoft SharePoint. The company wants to:

- Give all employees access to SharePoint.
- Restrict access to documents that contain employee compensation information. Only employees in human resources can access documents that contain compensation information.

Note: Forward, Inc. is a fictitious company name that is used strictly for instructional purposes only and is not meant to reference an existing company.

The following details how to configure an application with CA DLP content classifications to create a security policy that protects employee compensation information.

Given:

- The SiteMinder environment contains one LDAP user directory. The name of the directory is ForwardLDAP.
- The user directory identifies all employees, including employees who work in human resources. The user directory identifies the human resources employees as follows:

  `cn=human resources,o=forwardinc.com`

- All employees must authenticate with the Basic authentication scheme.
- A single SharePoint site is deployed:
  - The root URL of the SharePoint site is:

    `usecase.forwardinc.com/SitePages/Home.aspx`

  - The URL of the shared documents directory is:

    `usecase.forwardinc.com/Shared Documents`

Solution:

1. Protect the shared documents directory.
2. Protect all document resources.
3. Create the human resources role.
4. Establish a policy that is based on the human resources role.

Protect the Shared Documents Directory

For this use case, the highest point in the SharePoint environment that you want to protect is the shared documents directory. You leave the SharePoint site (usecase.forwardinc.com/SitePages/Home.aspx) unprotected to be sure that all employees have access.

Follow these steps:

1. Click Policies, Application.
2. Click Applications.
3. Click Create Application.
4. Select the Use DLP Server option.
5. Enter the required values for the General section. For this use case, the following data is specified:

   Name

   SharePoint Site

6. Enter the required values in the Component section. For this use case, the following data is specified:

   Component Name

   Shared Documents

   Agent

   SharePoint

   Resource Filter

   /usecase.forwardinc.com/Shared Documents

7. In the User Directories section, click Add/Remove to list all user directories available to the application policy.
8. Select a user directory and click OK. For this use case, the following user directory is specified:

   ForwardLDAP

9. In the DLP Classifications section, remove any content classifications that you do not want to add to the component. For this use case, do not remove any classifications.
10. Click Submit to save the application.

Protect All Document Resources

For this use case, you want to protect all documents in the Shared Documents directory. You protect all document resources by adding a resource to the application.

Follow these steps:

1. Click Policies, Application.
2. Click Applications.
3. Specify search criteria and click Search.
4. Identify the application you want to modify and click the modify icon. For this use case, modify the following application:

   SharePoint Site.
5. Select the Resources tab and click Create.
6. Enter values for the following fields. For this use case, enter the following:

   Name

   All Documents

   Resource

   /*
7. Select the type of event and action that must take place for the rule to fire. For this use case, select the following:
   - Web Agent actions
   - Connect
   - Delete
   - Get
   - Head
   - Options
   - Post
   - Put
   - Trace

8. Click OK to save the resource.
9. Click Submit to add the resource to the component.

Create the Human Resources Role

For this use case, you want to create a role that that only lets human resources employees access to documents that contain compensation information.

Follow these steps:

1. Click Policies, Application.
2. Click Applications.
3. Specify search criteria and click Search.
4. Identify the application you want to modify and click the modify icon. For this use case, modify the following application:

   SharePoint Site.
5. Select the Roles tab and click Create Role.
6. Select the following option:

   Create a new object of type role
7. Click OK.
8. Enter values in the General section. For this use case, enter the following:

   Name

   Human Resources

   Description

   A role that has access documents that contain employee compensation information.
9. Specify if the role is to apply to all or specific users in the available directories. For this use case, select the following option:

   Selected Users
10. Define the role members in the Users Setup section. For this use case, select the following organization in the Member Groups section:

    cn=human resources,o=forwardinc.com
11. Add content classifications to the role in the DLP Classifications section. For this use case, select the following:

    - US Employee Compensation Information
    - UK Employee Compensation Information

12. Click OK to save the role.
13. Click Submit to add the role to the application.

Establish a Policy Based on the Human Resource Role

For this use case, you want to create a policy that protects documents that contain employee compensation information.

Follow these steps:

1. Click Policies, Application.
2. Click Applications.
3. Specify search criteria and click Search.
4. Identify the application that you want to modify and click the modify icon. For this use case, modify the following application:

   SharePoint Site.
5. Select the Policies tab.
6. Select the Human Resources role for the All Documents resource.
7. Click Submit.

   A policy that only lets human resources employees access SharePoint documents that contain employee compensation information is created.

Configure Confidence Levels in Applications

If SiteMinder is integrated with a supported risk analysis engine, a confidence level is available for use in application objects. Confidence levels extend applications to include the results of the risk evaluation that is completed as part of user authentication. The Policy Server can use these results when making authorization decisions.

You can apply a confidence level to the following objects:

- An application component.

  A confidence level that you configure in an application component applies to all resources that are associated with the component. Confidence levels represent a higher level of granularity than the default application settings provide. Use the advanced settings of the application component to apply a confidence level.

  Note: Applying a confidence level to an application component requires that you enable confidence level support. For more information, see the *SiteMinder Implementation Guide*.

- An application role.

  A confidence level that you configure as part of an application role allows for more granular authorization decisions. A confidence level represents an active component that you can use to define further the user group or groups that can access the resources. Confidence levels represent a higher level of granularity that the default role memberships provide. Use a named expression that references the SM_USER_CONFIDENCE_LEVEL SiteMinder generated attribute to add a confidence level to a role.

  Note: Applying a confidence level to an application role remains supported from previous releases and is enabled by default.

More information:

Named Expressions (see page 244)
SiteMinder Generated User Attributes (see page 550)

Configure CA DLP Content Classifications in Applications

If SiteMinder is integrated with CA DLP, content classifications are available for use with application objects. Content classifications extend applications to include the type of content a user is requesting. The Policy Server can use the results of the CA DLP content analysis to make authorization decisions.

Note: Applying content classifications to an application component requires that you enable the environment for the CA DLP integration. For more information, see the *SiteMinder Implementation Guide*.

You can apply a CA DLP content classification to the following objects:

- An application component:
  - Content classifications that you apply to an application component apply to all resources that are associated with the component.
  - By default, the CA DLP Content Classification Service makes available all content classifications to an application component. If you want the Policy Server to ignore one or more classifications when making an authorization decision, remove them from the application.
  - A SiteMinder administrator cannot administer content classifications. If changes are required, coordinate the changes with the CA DLP administrator.

- An application role:
  - Content classifications that you apply to an application role allow for more granular authorization decisions. A content classification represents an active component that you can use to define further the user group or groups that can access the resources. Content classifications represent a higher level of granularity that the default role memberships provide.
  - The relationship between roles and content classifications is cumulative. If a user is a member of multiple roles on the same application, then the user can access any documents that are associated with the roles.

Configure Advanced Policy Components for Applications

Application objects provide configuration options that let the following types of users modify SiteMinder components beyond the default settings:

- Those users who are familiar with the policy design and interface used in SiteMinder releases before r12 who want to fine–tune aspects of their policies.
- Users who want a higher level of granularity in their policies than the default settings provide.
- Auditors and others who want to determine if the policies implemented using an application object meet the requirements of the organization or of any regulations with which the organization must comply.

Follow these steps:

1. Click Applications.
2. Click Create Application.
3. Enter information in the General and Components sections and then click Advanced Settings.

   The Modify Component page appears. The Modify Component page includes the session and advanced features of policy realms. For example, if confidence level support is enabled, you can add a minimum confidence level to the component.

   Note: For more information about enabling confidence level support, see the *SiteMinder Implementation Guide*.
4. Do one of the following:
   - Configure one or more of the advanced settings for the component.
   - Select the legacy authorization directory mapping.
   - To create a sub–component, click Create Sub–Component to open the Create Component screen. Creating a sub–component is the equivalent of creating a sub, or nested, realm.

5. When you are finished, click OK to save the changes and continue configuring the remaining parts of the application.

More information:

Realms Overview (see page 501)
Rules Overview (see page 513)
Session Timeouts (see page 110)
Authentication Events (see page 517)
Authorization Events (see page 519)

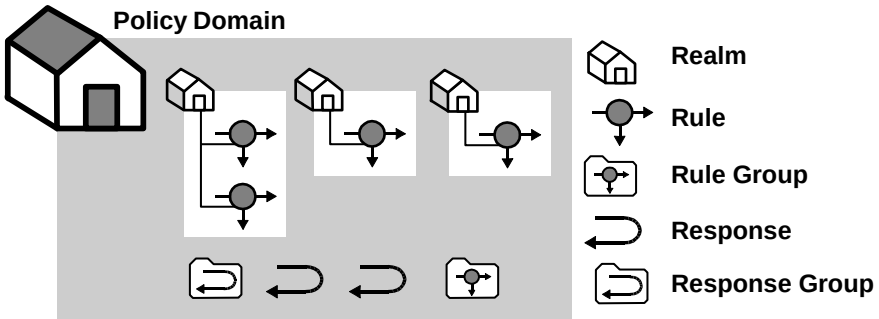
Chapter 15: Domains

This section contains the following topics:

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. For information about administrator privileges, see Policy Server Administrators.

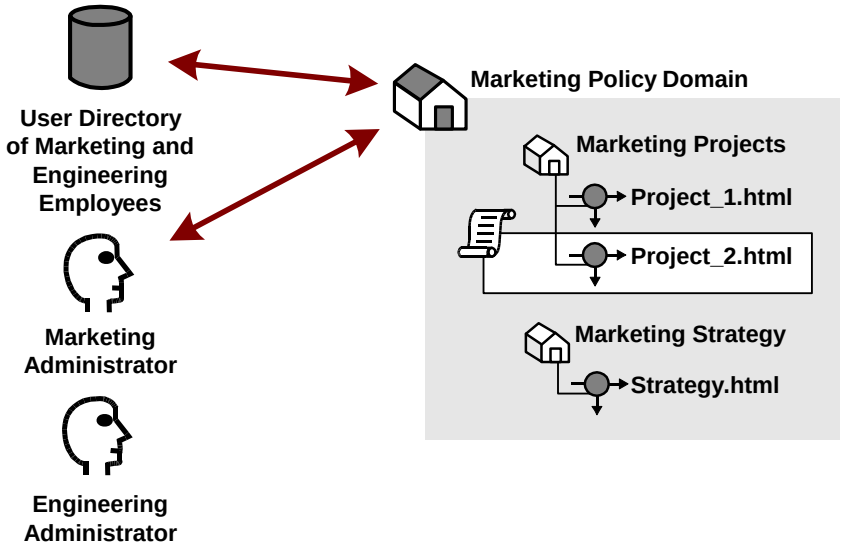
The resources in a policy domain can be grouped in one or more realms. A realm is a set of resources with a common security (authentication) requirement. Access to resources is controlled by rules, which are associated with the realm that contains the resource. The following diagram illustrates a small 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.

In the sample diagram below, a Marketing policy administrator who is specified in the Marketing policy domain can manage the Marketing Strategy and Marketing Projects realms. The policy domain ensures that the 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 is associated with a user directory that contains engineering users.

If the administrator for the Marketing department creates a policy within the Marketing policy domain that allows Engineering staff to access the resource Project 2.html, engineering users may access the resource.



More information:

Policies (see page 561)

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 501)

Configure a Policy Domain

You create a policy domain to protect logical groupings of resources.

Follow these steps:

1. Click Policies, Domain.
2. Click Domains.

   The Domains page appears.
3. Click Create Domain.

   The Create Domain page 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 policy.
5. Add User Directories and Realms.
6. Click Submit.

   You have defined a policy domain.

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 Policies, Domain.
2. Click Domains.

   The Domains page appears.
3. Specify the search criteria and click Search.

   A list of domains that match the search criteria appears.
4. Click the name of the domain that you want to modify.

   The View Domain page appears.
5. Click Modify.

   The settings and controls become active.
6. In the General tab, click Add/Remove.

   The Choose user directories page appears.
7. 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 the additional members. To select a block of members, click the first member and then hold down the Shift key while you click the last member in the block.

8. Click OK.

   The selected user directories are listed under User Directories.

   Note: To create a new user directory and add it to the domain, click Create.

9. Click Submit.

   The selected user directories are added to the policy domain.

Create a Realm

Realms are created in a domain and are associated with a Web Agent. Realms use resource filters to group resources that have similar security requirements and share a common authentication scheme.

More information:

Realms (see page 501)

Configure a Realm with a SiteMinder Web Agent

When you create a domain, you can create one or more realms in the domain and associate them with a Web Agent or Agent group. Realms group resources that have similar security requirements and share a common authentication scheme.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object.

Follow these steps:

1. Click Policies, Domain.
2. Click Realms.

   The Realms page appears.

3. Click Create Realm.

   The Create Realm: Select Domain page appears.

4. Select a domain, and click Next.

   The Create Realm: Define Realm page appears.

5. Type the name and a description of the realm.
6. Click the ellipsis button to select an agent.

   The Select an Agent page appears.

7. Select a Web Agent or Agent group, and click OK.

8. Specify the remaining resource properties.
9. Create new rules or delete existing rules.
10. Create new sub-realms or delete existing sub-realms.
11. Specify the session properties.
12. Specify the authorization directory mappings and types of events the realm must process.
13. Click Finish.

    The Realm is created.

More information:

Duplicate Policy Server Objects (see page 54)

Configure a Realm with a RADIUS Agent

When you create a domain, you can create one or more realms in the domain and associate them with a Radius Agent or Agent group. Realms group resources that have similar security requirements and share a common authentication scheme.

Note: The Administrative UI lets you configure realms protected by a RADIUS Agent. These realms do not require all of the same information that is required for a SiteMinder Web Agent realm.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object.

To create a realm in a domain and associate it with a RADIUS Agent or Agent group

1. Click Policies, Domain.
2. Click Realms.

   The Realms page appears.
3. Click Create Realm.

   The Create Realm: Select Domain page appears.
4. Select a domain, and click Next.

   The Create Realm: Define Realm page appears.
5. Type the name and a description of the realm.
6. Click the ellipsis button to select an agent.

   The Select an Agent page appears.
7. Select a RADIUS Agent or Agent group, and click OK.

8. Specify the remaining resource properties.
9. Create new rules or delete existing rules.
10. Specify the session properties.
11. Click Finish.

    The Realm is created and associated with the selected RADIUS Agent or Agent group.

More information:

Duplicate Policy Server Objects (see page 54)

Add CA Identity Manager Environments to a Domain

Adding a CA Identity Manager environment and the associated user directories to a domain makes available CA Identity Manager roles to policies.

Follow these steps:

1. Click Policies, Domain.
2. Click Domains.
3. Specify search criteria and click Search to locate the domain you want.
4. Click the name of the domain to which you want to add the environment.
5. Click Modify.
6. If the users that are associated with the environment are not bound to the domain, add the respective user directories.
7. Click Add/Remove in the IDM Environments section.
8. Select the IDM Environments you want and click OK.
9. Click Submit.

   The CA Identity Manager environments are added to the domain. The roles associated with the environments are available to all policies created in the domain.

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 641)

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 about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

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 about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

Chapter 16: Realms

This section contains the following topics:

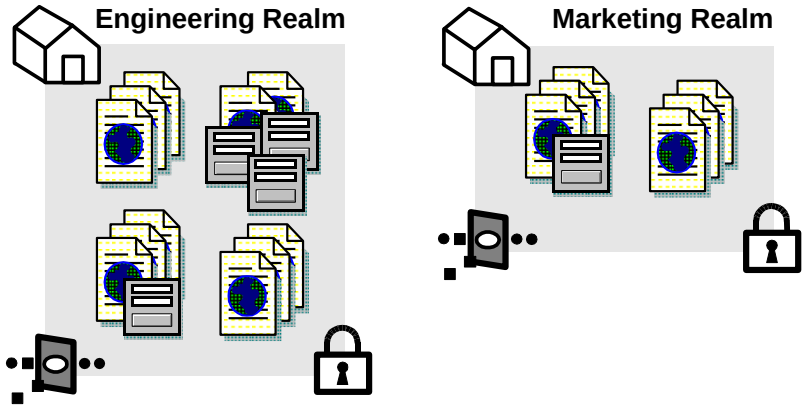
Realms Overview

Complex sets of resources must be logically grouped so that security policies can be created. The basic SiteMinder 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, marketing information that resides in a /marketing directory on your network might be configured as a realm in a policy domain controlled by an administrator in your company's marketing organization.

The contents of a realm are protected by Agents. When users request resources within a realm, the associated Agent handles authentication and authorization of the user. The realm determines the method of authentication.

The following diagram shows the contents of two realms.



Each of the realms contains resources, such as HTML files, forms, or applications. In addition, each realm is associated with an Agent and an authentication scheme.

Confidence Levels Introduced

If SiteMinder is integrated with a supported risk analysis engine, you can configure a realm with a confidence level. A confidence level represents credential assurance, which is the likelihood that the user requesting the protected resource is legitimate.

Consider the following items:

- A confidence level is based on a numeric scale (0–1000). A higher confidence level corresponds to a higher level of credential assurance.
- You can apply a confidence level to authorization decisions using both access management models. For more information, see Confidence Levels in Policies (see page 565) and Confidence Levels in Applications (see page 488).
- When an authentication scheme that generates a confidence level authenticates a user, the confidence level is inserted into the session ticket of the user. The user can access any resource requiring a confidence level equal to or greater than this value, as long as the policy or application applies to the user.
- The factors on which a confidence level is calculated are based on the data the risk analysis engine uses to determine credential assurance.

  Note: SiteMinder supports an integration with an on–premise implementation of CA Arcot WebFort and CA Arcot Risk, and a hosted implementation of CA Arcot A–OK. For more information, see the *SiteMinder Implementation Guide.*

More information:

Configure Advanced Policy Components for Applications (see page 490)

Identify a Resource by Agent, Realm, and Rule

The resources protected by SiteMinder are identified by the following:

[Agent] [Realm Resource Filter] [Rule Resource]

Agent

An Agent monitors a server that contains one or more realms of protected resources.

Realm Resource Filter

A string, such as a relative path to a directory, 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.

For example, the realm might cover the contents of a directory that is immediately below the document root of a Web server, such as:

<document_root>/HR

Here, you could specify the realm resource filter as:

/HR

Rule Resource

A string or regular expression that specifies the resources to which the rule applies. Specify the resources relative to the realm containing the resource. For example, if the realm resource filter ends with a directory name, the rule resource might include a subdirectory of the realm directory and even the name of a file in that subdirectory, such as:

/Managers/PayRanges.html

You can use wildcards to broaden the specification of a rule. For example:

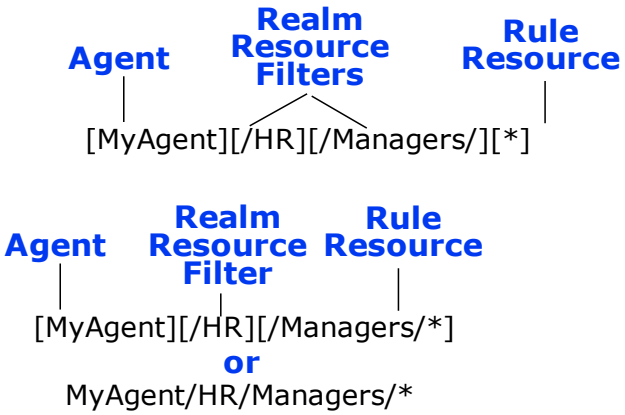
/Managers/*

Combining the three elements, suppose that:

- The Agent called MyAgent protects a Web server on host MyHost, in domain myorg.org.
- You want the realm to cover the contents of the following Web Server directory:

  <document_root>/HR
- The HR directory contains a subdirectory called Managers, and you want to protect all files in the subdirectory.

For the Policy Server, the following figure shows the effective resource.



You could configure the directory called Managers as a nested realm under the /HR realm.

To access the protected page PayRanges.html, under the Managers subdirectory, a user would need to:

1. Specify the resource:

   http://MyHost.myorg.org/HR/Managers/PayRanges.html
2. Provide credentials for a user authorized to access the resource. Administrators use policies to specify which users are authorized to access a resource.

More information:

Agents and Agent Groups (see page 117)
Configure a Realm (see page 507)
Rules (see page 513)

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 SiteMinder 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: file.html.	The file /dir/file.html 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/file.html. All other users cannot access the protected file.

Note: If you want to track users for a realm with an Anonymous authentication scheme, the realm must be a protected realm. For information about the Anonymous scheme, see Anonymous Authentication Schemes (see page 384).

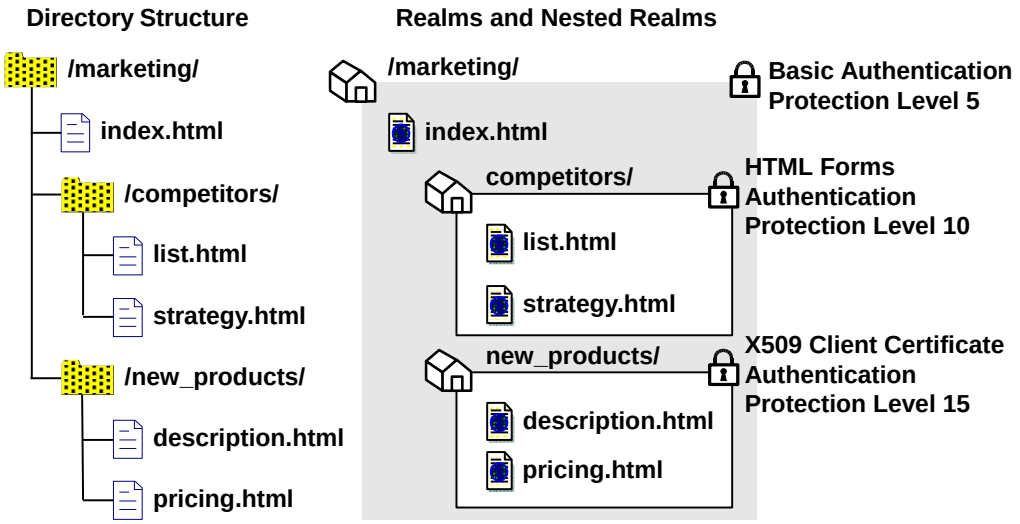
Nested Realms

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.

Note: Do not assign the anonymous authentication scheme to any realm in a nested structure, including the top-level realm. You can't authorize specific users for resources protected by an anonymous authentication scheme, so the and logic will fail.

The following example illustrates how nested realms can be used to provide increasing levels of security.



In the realm structure shown in the previous figure, the realms mimic the file structure of the resources. Each of the nested realms has a different authentication scheme than its parent realm. Since the authentication scheme for each child realm has a higher protection level than that of the parent realm, users will 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 (see page 513)
Policies (see page 561)

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.

Examples

Using the example from the previous section, a file called list2.html in the location /marketing/competitors/list2.html matches the nested realm /marketing/competitors/. When the Policy Server processes authentication for list2.html, the user authenticates via HTML Forms, since that is the authentication scheme associated with the /marketing/competitors/ realm.

In the same example, a file called current_budget.html in the location /marketing/budgets/current_budget.html. Since the /budget directory is not specifically called out in a nested realm, the longest matching realm for this resource is /marketing/. Therefore, the user authenticates via the Basic user name and password authentication scheme.

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 users request resources within a realm, the associated Agent handles authentication and authorization of the user. The realm specifies the method of authentication.

You can configure realms for any type of SiteMinder Agent, including Affiliate Agents.

Configure a Realm Protected by a SiteMinder Web Agent

You can configure a realm to protect a group of resources that users access through a Web Server.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object.

To configure the realm

1. Click Policies, Domain.
2. Click Realms.

   The Realms page appears.
3. Click Create Realm.

   The Create Realm: Select Domain page appears.
4. Select a domain from the Domain list, and click Next.

   The Create Realm: Define Realm page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Type the name and a description of the realm.
6. Click the ellipsis button to select an agent.

   The Select an Agent page appears.
7. Select a SiteMinder Web Agent or Agent group, and click OK.
8. Specify the remaining resource properties.
9. Create new rules or delete existing rules.
10. Create new sub–realms or delete existing sub–realms.
11. Specify the session properties.
12. Specify the directory mappings and types of events the realm must process.
13. Specify the minimum confidence level to apply to all resources in the realm.

    Note: Applying a confidence level to a realm requires that you integrate SiteMinder with a supported risk analysis engine and enable confidence level support. For more information, see the *Implementation Guide*.
14. Click Finish.

    The Realm associated with the selected SiteMinder Web Agent or Agent group is created.

More information:

Duplicate Policy Server Objects (see page 54)

Configure a Realm Protected by a RADIUS Agent

You can configure a realm to protect a group of resources that users access through a Web Server.

Note: The Administrative UI lets you configure realms protected by a RADIUS Agent. These realms do not require all of the same information that is required for a SiteMinder Web Agent realm.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object.

To configure the realm

1. Click Policies, Domain.
2. Click Realms.

   The Realms page appears.
3. Click Create Realm.

   The Create Realm: Select Domain page appears.
4. Select a domain from the Domain list, and click Next.

   The Create Realm: Define Realm page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Type the name and a description of the realm.
6. Click the ellipsis button to select an agent.

   The Select an Agent page appears.
7. Select a RADIUS Agent or Agent group, and click OK.
8. Specify the remaining resource properties.
9. Create new rules or delete existing rules.
10. Specify the session properties.
11. Click Finish.

    The Realm associated with the selected RADIUS Agent or Agent group is created.

More Information:

Duplicate Policy Server Objects (see page 54)

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 about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

Delete a Realm

When you delete a realm, all nested realms that are associated with the realm are also deleted. In addition, all rules that are associated with the deleted realm and its nested realms are also deleted. Only delete the parent realm to remove all the associated nested realms. Do not select the individual nested realms for deletion.

Note: More information about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

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.

Note: You can only create nested realms under a realm that is protected by SiteMinder Web Agents.

To create a nested realm

1. Click Policies, Domain.
2. Click Realms.

   The Realms page appears.
3. Specify search criteria, and click Search.

   A list of realms that match the search criteria appears.
4. Click the name of a realm that you want to modify.

   The View Realm page appears.
5. Click Modify.

   The settings and controls become active.
6. In Sub-Realms, click Create Sub-Realm.

   The Create Realm page appears.
7. Type the name and a description of the realm.
8. Type the path of the resource filter.

   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 the session properties.
10. Specify the following advanced settings:
    - Authorization directory mapping
    - Events processing
11. Click OK.

    The newly created Sub-Realm is added to the parent Realm.
12. Click Submit.

    A Realm associated with the newly created sub-realm is created.

More information:

Nested Realms (see page 505)

Flush a Single Realm from the Resource Cache

SiteMinder caches realm information when users access protected resources. This allows SiteMinder to improve network performance by keeping track of recently used resources. However, if you change the security requirements or contents of a realm, you must flush the realm from the SiteMinder resource cache.

Note: If you have the Manage System and Domain Objects administrative permission, you can flush all realms from the resource cache using the Cache Management dialog. For more information, see the *Policy Server Administration Guide*.

To flush a single realm from the resource cache

1. Click Policies, Domain.
2. Click Realms.

   The Realms page appears.
3. Specify search criteria, and click Search.

   A list of realms that match the search criteria appears.
4. Click the name of a Realm that you want to modify.

   The View Realm page appears.
5. Click Modify.

   The settings and controls become active.
6. In Advanced, click Flush.

   SiteMinder flushes the realm from the resource cache.

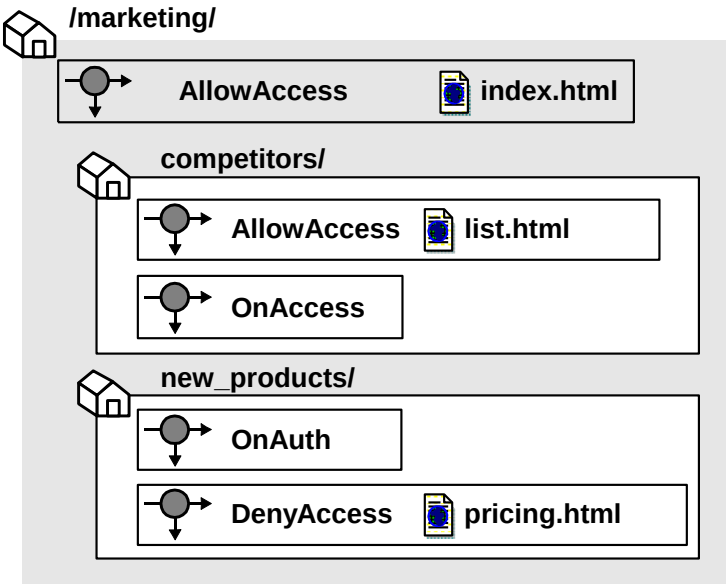
Chapter 17: Rules

This section contains the following topics:

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 525)

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 SiteMinder 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 SiteMinder 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 SiteMinder Agent. For OnAuthReject rules, the Policy Server only returns the responses associated with the rule in the deepest nested realm to the SiteMinder 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.

More information:

Nested Realms (see page 505)

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 an HTML page, 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 fires, the Policy Server processes the action specified in the rule based on the way the policy that contains the rule is configured. For example, if a user is not in a group specified in a policy, a rule that allows an HTTP Get action for an HTML page will not allow the user to access the resource.

More information:

Policy Overview (see page 561)

Web Agent Actions

Rules with a web agent action either allow or deny access to the resources specified by a rule when one of the HTTP actions specified in the rule occur.

When a rule that specifies Allow Access fires, SiteMinder allows the user to access the specified resource upon authenticating successfully. If a rule specifies Deny Access, SiteMinder denies access to the successfully authenticated user. Deny access rules can be added to policies to provide an additional layer of security by rejecting specific individuals or groups who cannot 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.

The Web Agent rule actions are:

- HEAD
- GET
- POST
- PUT
- DELETE
- TRACE
- OPTIONS

SOA Agent Actions

If you have purchased CA SOA Security Manager, two additional Web Agent rule actions are available for SOA Agent use:

ProcessSOAP

Supports incoming XML messages wrapped with a SOAP envelope.

ProcessXML

Supports incoming raw XML messages not wrapped with a SOAP envelope.

For more information, see the *CA SOA Security Manager Policy Configuration Guide*.

Affiliate Agent Actions

Affiliate Agents are SiteMinder Agents that communicate with SiteMinder Web Agents installed on the Web servers of a portal Web site.

Affiliate Agent rules are very simple, since they do not protect the resources of an affiliate Web site. The Affiliate Agent processes responses sent from the portal site, so that applications on the affiliate Web site may take advantage of the information gathered about users on the SiteMinder protected portal Web site.

Affiliate Agent actions are only available in the place of Web Agent actions for realms associated with an Affiliate Agent. There is only one possible action:

Visit

Allows a SiteMinder portal site to interact with an affiliate Web site

Note: For more information about Affiliate Agents, see the *Web Agent Configuration Guide*.

Authentication Events

Authentication events occur as SiteMinder tries to establish a user identity. As a rule action, an authentication event causes the Policy Server to fire a rule at a particular point in the authentication process.

Authentication events occur when a user accesses a resource protected by a rule that includes an On-Auth event. Unlike Web Agent actions or authorization events, authentication events always apply to the entire realm. You cannot create an On-Auth rule that applies to a portion of a realm.

The following is a list of possible authentication events:

OnAuthAccept

Occurs if authentication was successful. This event can be used to redirect a user after a successful authentication.

OnAuthAcceptCredentials

Occurs only during the login stage. The user credentials are presented and generate the creation of a new session.

OnAuthReject

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 their username and password) and at validation time (when the user 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 your Web Agent supports the EnforceRealmTimeouts option and that option is enabled, the user Idle and Max Timeouts remain at the values for the realm in which the user last authenticated. The values only change if the user has to reenter their credentials.

Redirects

Redirects are only allowed at authentication time to prevent the possibility of infinite redirect loops.

Access to the user 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).

Note: OnAuth event results are per realm. So for example, if a user goes from realm A to realm B and the user has an OnAuthAccept header in realm A, it is not available in realm B. When the user goes back to realm A, the header is set again.

OnAuthAttempt

Occurs if the user is rejected because SiteMinder does not know this user. For example, an unregistered user can be redirected to register first.

OnAuthChallenge

Occurs when custom challenge-response authentication schemes are activated (for example, a token code).

OnAuthUserNotFound

This event is only used to trigger Active Responses. Do not use this event to trigger any response other than an Active Response.

A rule with an authentication event action may be coupled with a SiteMinder response in a policy. When a user is authenticated (or rejected), the Policy Server passes any response that is associated with the applicable On-Auth rule back to the requesting Agent.

Note: You can optimize SiteMinder performance and can limit the number of times the Web Agent must retrieve static information from the Policy Server. To optimize performance, set up a rule that is based on the OnAuthAccept authentication event and create a response that returns the static information. When you bind the rule and response in a policy, the rule fires for users specified in the policy. The static response is only returned to users who successfully authenticate.

Authorization Events

Authorization events occur as SiteMinder verifies whether or not a user is authorized to access a resource. As a rule action, an authorization event causes the Policy Server to fire a rule at a particular point in the authorization process.

The following is a list of possible authorization events:

OnAccessAccept

Occurs as the result of successful authorization. This event may be used to redirect users who are authorized to access a resource.

OnAccessReject

Occurs as the result of failed authorization. This event may be used to redirect users who are not authorized to access a resource.

A rule with an authorization event action may be coupled with a SiteMinder response in a policy. 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.

Impersonation Events

Impersonation provides a method for a privileged user to assume the role of another user without ending the privileged user's session. Impersonation events are used to start impersonation sessions when resources are accessed.

Possible impersonation events:

ImpersonationStart

When included in an appropriate policy, a rule that includes this event allows an impersonation session to begin.

ImpersonationStartUser

When included in an appropriate policy, a rule that includes this event allows a set of users to be impersonated.

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.

More information:

Add Time Restrictions to Rules (see page 528)

Configure a Rule for Web Agent Actions

You can create a rule that fires in response to specified Web Agent actions and that allows or denies access to the resource that the rule is designed to protect.

Note: You can also create a rule for the CA SOA Security Manager XML Agent. When creating a rule for this Agent type, two additional rule actions are available: ProcessSOAP and ProcessXML. For more information, see the *CA SOA Security Manager Policy Configuration Guide*.

To create a rule

1. Click Policies, Domain.
2. Click Rules.

   The Rules page appears.
3. Click Create Rule.

   The Create Rule: Select Domain page appears.
4. Select a domain from the list, and click Next.

   The Create Rule: Select Realm page appears.
5. Select the realm that includes the resources that you want the rule to protect, and click Next.

   The Create Rule: Define Rule page appears.

   Note: If a realm does not exist for the resources that you want to protect, a rule cannot be created to protect those resources.

6. Type the name and a description of the rule.

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

7. Type the resource that you want the rule to protect.

   The Effective Resource updates to include the resource.

8. Specify whether the rules should allow or deny access to the protected resource.

9. Select Web Agent actions.

   The Action List is populated with HTTP actions.

10. Select one or more HTTP actions.

11. (Optional) In Advanced, specify time restrictions, an active rule, or both.

12. Click Finish.

    The Rule for Web Agent actions is created.

More information:

Regular Expressions for Resource Matching (see page 526)
Advanced Rule Options (see page 520)

Configure a Rule for Authentication Event Actions

You can configure a rule for authentication event actions to control actions that occur when users authenticate to gain access to a resource.

The realm in which the rule is to be created must be able to process authentication events. Verify that the Process Authentication Events option is selected.

To create a rule

1. Click Policies, Domain.

2. Click Rules.

   The Rules page appears.

3. Click Create Rule.

   The Create Rule: Select Domain page appears.

4. Select a domain from the list, and click Next.

   The Create Rule: Select Realm page appears.

5. Select the realm that includes the resources that you want the rule to protect, and click Next.

   The Create Rule: Define Rule page appears.

   Note: If a realm does not exist for the resources that you want to protect, a rule cannot be created to protect those resources.

6. Type the name and a description of the rule.

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

7. Select Authentication events.

   The Action List populates with authentication events.

   Note: The Resource field is disabled because an authentication event applies to the entire realm. The Allow Access and Deny Access options are also disabled as they do not apply to authentication events.

8. Select one or more authentication events.

9. (Optional) In Advanced, set time restrictions and or active rule settings.

10. Click Finish.

    The rule is saved and applied to the specified realm and resource.

More information:

Authentication Events (see page 517)
Configure a Realm (see page 507)
Advanced Rule Options (see page 520)

Configure a Rule for Authorization Event Actions

Authorization events occur after a user is authenticated. You configure a rule for authorization to let SiteMinder call responses based on whether a user is or is not authorized for the requested resource. When the user has been granted or denied access based on their privileges, the appropriate event is triggered.

The realm in which the rule is to be created must be able to process authorization events. Verify that the Process Authorization Events option is selected.

To create a rule

1. Click Policies, Domain.
2. Click Rules.

   The Rules page appears.
3. Click Create Rule.

   The Create Rule: Select Domain page appears.
4. Select a domain from the list, and click Next.

   The Create Rule: Select Realm page appears.
5. Select the realm that includes the resources that you want the rule to protect, and click Next.

   The Create Rule: Define Rule page appears.

   Note: If a realm does not exist for the resources that you want to protect, a rule cannot be created to protect those resources.
6. Type the name and a description of the rule.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
7. Type the resource the rule is to protect.

   Effective Resource updates to include the resource.
8. Select Authorization events.

   The Action List populates with authorization events.

   Note: The Allow Access and Deny Access options are disabled. These options do not apply to authorization events.
9. Select one or more authorization events.
10. (Optional) In Advanced, set time restrictions, an active rule, or both.
11. Click Submit.

    The rule is saved and applied to the specified realm and resource.

More information:

Authorization Events (see page 519)
Configure a Realm (see page 507)
Regular Expressions for Resource Matching (see page 526)
Advanced Rule Options (see page 520)

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, SiteMinder 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.

Configure a Rule for Impersonation Event Actions

You can configure a rule for impersonation events to start impersonation sessions when resources are accessed.

To create a rule

1. Click Policies, Domain.
2. Click Rules.

   The Rules page appears.
3. Click Create Rule.

   The Create Rule: Select Domain page appears.
4. Select a domain from the list, and click Next.

   The Create Rule: Select Realm page appears.

5. Select the realm that includes the resources that you want the rule to protect, and click Next.

   The Create Rule: Define Rule page appears.

   Note: If a realm does not exist for the resources that you want to protect, a rule cannot be created to protect those resources.

6. Type the name and a description of the rule.

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

7. Type the resource the rule is to protect.

   The Effective Resource updates to include the resource.

8. Select Impersonation events.

   The Action List populates with impersonation events.

   Note: The Allow Access and Deny Access options are disabled. These options do not apply to impersonation events.

9. Select one or more impersonation events.

10. (Optional) In Advanced, set time restrictions, an active rule, or both.

11. Click Finish.

    The rule is saved and applied to the specified realm and resource.

More information:

Impersonation (see page 663)
Regular Expressions for Resource Matching (see page 526)
Advanced Rule Options (see page 520)

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 SiteMinder 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
(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

Characters	Results
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 SiteMinder protects the specified resources. You disable a rule to prevent SiteMinder 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.

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 Rules

You configure time restrictions to specify when SiteMinder should fire the rule.

Configuring a time restriction from 9am - 5 pm, Monday - Friday, for example, specifies that SiteMinder 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 SiteMinder handles time across multiple time zones exists in How the Web Agent and Policy Server Calculate Time (see page 61).

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 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 in the Software Development Kit.

Note: For more information about shared libraries, see the *Programming Guide for C*.

To configure an Active Rule

1. Specify the library name, function name, and function parameters in the fields on the Active Rule group box.

   The active rule string is displayed in the Active Rule field.

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

2. 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 about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

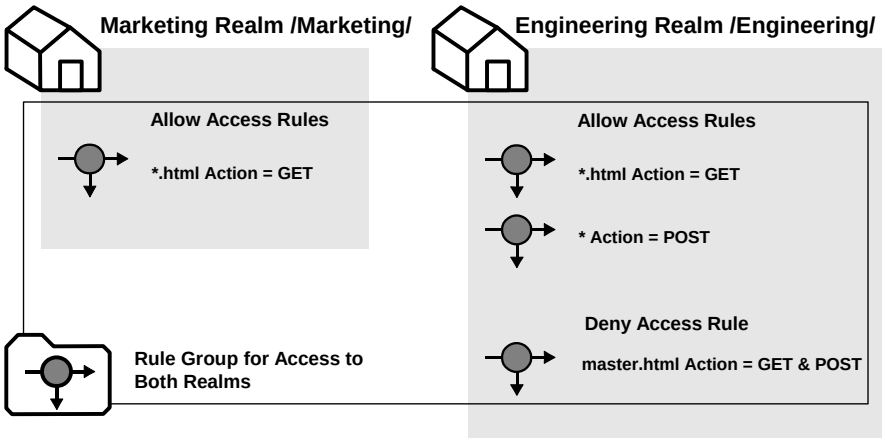
Chapter 18: Rule Groups

This section contains the following topics:

Rule Group Overview

A rule group is a set of rules that can be bound to SiteMinder 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 a GET action for different resources of a Web site, 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.



The previous diagram illustrates a rule group that contains rules for both the Marketing realm and the Engineering realm. The rule group can be used in a policy rather than including all four rules separately.

More information:

Policy Overview (see page 561)

Create a Rule Group

You can create a rule group and add it to a domain.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object.

To create a rule group

1. Click Policies, Domain.
2. Click Rule Groups.

   The Rule Groups page appears.
3. Click Create Rule Group.

   The Create Rule Group page appears. Verify that the Create a new object of type Rule Group option is selected
4. Click OK.

   The Create Rule Group: Select Domain page appears.
5. Select a domain and click Next.

   The Create Rule Group: Define Rule Group page appears.
6. Type the name and a description of the rule group.
7. Select Radius or SiteMinder and an Agent Type.
8. In Group Members, click Add/Remove.

   The Rule Group Members page appears.

   The Available Members column lists all rules that are defined in the specified domain and in the realms associated with the specified Agent type. When the Agent type is Generic RADIUS, the Available Members column lists all rules that the RADIUS Agents support.
9. 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 the additional members. To select a block of members, click the first member and then hold down the Shift key while you click the last member in the block.
10. Click OK.

    The selected rules are listed under Group Members.
11. Click Finish.

    The Rule Group is created.

More information:

Duplicate Policy Server Objects (see page 54)

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, Domain.
2. Click Rule Groups.

   The Rule Groups page appears.
3. Specify search criteria, and click Search.

   A list of rule groups that match the search criteria appears.
4. Click the name of a rule group that you want to modify.

   The View Rule Group page appears.
5. Click Modify.

   The settings and controls become active.
6. In Group Members, click Add/Remove.

   The Rule Group Members page appears.

   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. When the Agent type is Generic RADIUS, the Available Members column lists all rules that the RADIUS Agents support.
7. 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 the additional members. To select a block of members, click the first member and then hold down the Shift key while you click the last member in the block.
8. Click OK.

   The selected rules are listed under Group Members.
9. Click Submit.

   The selected rules are added to the rule group.

Modify a Rule Group

You can modify all of the properties of a rule group, except the Agent Type for SiteMinder 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 about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

Delete a Rule Group

Deleting a rule group only deletes the grouping. The rules contained in the grouping are not deleted.

Note: More information about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

Chapter 19: Responses and Response Groups

This section contains the following topics:

Responses

A response passes static text, user attributes, DN attributes, customized active responses, or the runtime values of defined variables from the Policy Server to a SiteMinder Agent. Responses can be used by servlets, Web applications, or other custom applications to display customized content, change SiteMinder settings, or redirect users to different resources. When working with Web applications, responses can be used as privileges or entitlements for fine-grained access control.

A policy 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 SiteMinder Agent.

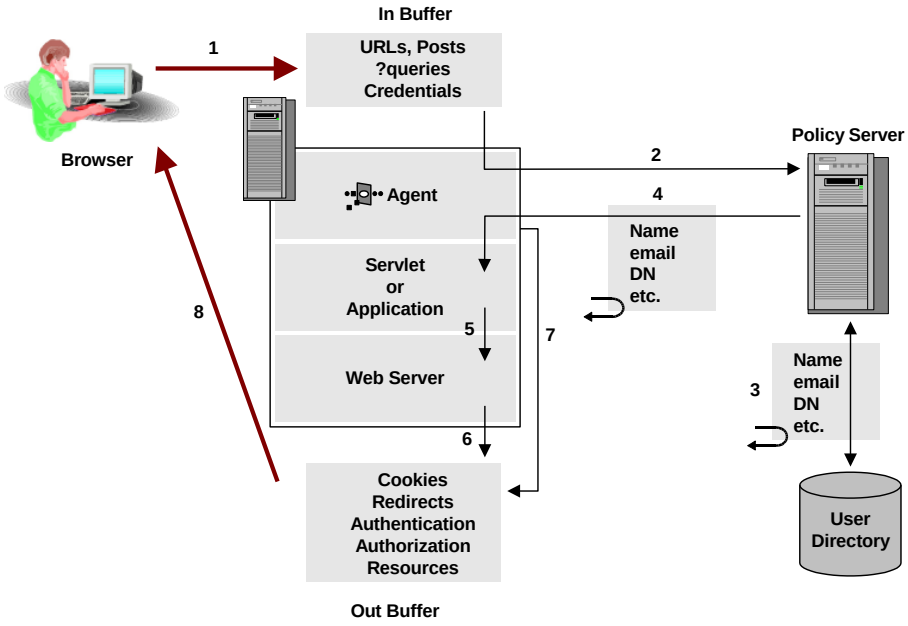
Responses take the form of name/value pairs. When a rule is triggered, the Policy Server returns the paired response to the SiteMinder Agent.

For example, if a user attempts to access a protected Web page, but is not authorized to view the contents of the page, a response can redirect the user to an HTML page that indicates the user does not have access, and provide details for contacting a system administrator.

For Web Agents, SiteMinder adds response attributes to HTTP header variables or HTTP cookie variables so that the responses are available to the Web resource or application named in the rule. In a RADIUS environment, the response is returned to the RADIUS client.

How SiteMinder Processes Responses

The following diagram illustrates how SiteMinder uses responses when processing a user's request for resources.



In the previous diagram, SiteMinder processes responses using the following steps:

1. A user requests a resource that is protected by a SiteMinder Agent.

   The In Buffer represents the Web Server buffer where the requested URL, Post data or query strings reside during Web Server processing.

2. The SiteMinder Agent intercepts requests for protected resources, and communicates with the Policy Server to authenticate and authorize the user.

   Part of the authentication process binds the user to a record in a user directory.

3. The Policy Server uses the binding to retrieve attributes specified in a SiteMinder response from the user's entry in the user directory.

4. The Policy Server passes user attributes specified in the response back to the Web Agent.

5. The attributes returned to the Web Agent may be used by a servlet or application that has been customized to use the attributes specified in the response.

   The servlet or application executes its processes and passes its results to the Web Server.

6. The Web Server's Out Buffer contains the resulting information that must be returned to the user.

7. The Web Agent adds any SiteMinder specific information to the Web Server's Out Buffer.

   The Web Agent may pass any of the following to the Out Buffer: SiteMinder cookies, URLs for redirection, and successful /unsuccessful authentications or authorizations.

8. The Web Server passes the contents of the Out Buffer to the user.

Response Types

A response is a container for one or more response attributes. The response attributes are what a SiteMinder Agent receives after the Policy Server processes a response. The available response attributes differ based on the type of response.

There are three types of SiteMinder responses:

- Web Agent responses
- Affiliate Agent responses
- RADIUS responses.

Note: You can create response types for custom Agents and response attributes using the SiteMinder APIs, which are available separately with the Software Development Kit. More information exists in the API Reference Guide for C.

Web Agent Responses

Web Agent responses are SiteMinder responses that provide name/value pairs usable by a SiteMinder Web Agent. These responses can contain attributes for HTTP header variables, cookie variables, and URLs for redirections.

Affiliate Agent Responses

Affiliate Agent responses are SiteMinder responses that provide name/value pairs usable by a SiteMinder Affiliate Agent. These responses can contain attributes for HTTP header variables or HTTP cookie variables.

More information:

Affiliate Agent Response Attributes (see page 540)

RADIUS Responses

RADIUS responses are SiteMinder responses that provide values usable by a RADIUS Agent. These responses can contain response attributes for all supported RADIUS attributes.

More information:

RADIUS Agent Response Attributes (see page 541)

Response Attributes

Each SiteMinder response contains one or more response attributes. These attributes differ based on the type of response. The following sections discuss the response attributes that are available for each type of response.

Web Agent Response Attributes

Web Agent response attributes are response attributes that SiteMinder Web Agents can interpret and pass on to other applications. The following list describes the generally available Web Agent response attributes:

WebAgent-HTTP-Authorization-Variable

Indicates an attribute that is reserved for future use.

WebAgent-HTTP-Cookie-Variable

Generates a SetCookie header, which then sets a nonpersistent cookie in a web browser. The cookies only exist in the cookie domain where the 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 agent does not include header variables in the responses that it sends back to a web browser. Instead, these responses reside in the request headers of the web server.

Consequently, the header variables are not 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 URLs, depending on the type of response in which it is used:

- In an authorization response, a URL to direct the user to if the user is allowed access to a resource.
- In an authentication response, a URL to direct the user to if the user was authenticated for a security realm.

To specify whether an authorization response 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.

Note: When configuring a Web Agent OnAcceptText response, set the FCC Compatibility Mode parameter (fcccompatmode) corresponding to the Web Agent to yes. This action ensures that user authentication takes place at the Web Agent and that the text in the response is available for display in the browser. If the FCC Compatibility Mode parameter is set to no, user authentication takes place at the Forms Credential Collector (FCC). The response is triggered, but the text in the response is lost.

WebAgent-OnAuthAccept-Session-Idle-Timeout

Overrides the number of seconds a user session can be idle. When this limit is reached, the user is forced to authenticate again. 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. When this limit is reached, the user session is terminated and the user is forced to authenticate again. 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 session ticket as the value of the SM_AUTHENTICATIONCONTEXT user attribute. The value 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-OnAuthAccept-Session-Variable

Stores a particular Session Variable in the session store when an administrator has decided against persisting all authentication data.

Limits:Used in accept responses. Persistent Sessions are enabled.

WebAgent-OnReject-Redirect

Defines *one* of the following URLs:

- In an authorization response, a URL to direct the user to if the user is denied access to a resource.
- In an authentication response, a URL to direct the user to if the user has failed to authenticate for a security realm.

To specify an authorization response 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.

Affiliate Agent Response Attributes

Affiliate Agent response attributes are response attributes that SiteMinder Affiliate Agent can interpret and pass on to other applications at an affiliate Web site.

The following is a list of Affiliate Agent response attributes:

- AffiliateAgent-HTTP-Header-Variable
- AffiliateAgent-HTTP-Cookie-Variable

Note: For complete descriptions of the response attributes, see the *Web Agent Configuration Guide*.

RADIUS Agent Response Attributes

RADIUS Agent response attributes are response attributes that RADIUS Agents can interpret. All of the response attributes supported by SiteMinder correspond to the attributes described in the Request for Comments (RFC) 2138, which describes attributes supported by the RADIUS protocol.

Responses and Directory Mappings

Directory mappings let you specify a separate authorization user directory in application object component or 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.

Configure a Response

You can create a response by specifying an 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, Domain.
2. Click Responses.

   The Responses page appears.
3. Click Create Response.

   The Create Response: Select Domain page appears.
4. Select a domain and click Next.

   The Create Response: Define Response page appears.
5. Type the name and a description of the response.
6. Select Radius or SiteMinder and an Agent Type.
7. (Optional) Click Create Response Attribute to create a response attribute and add it to the attribute list.

   The Create Response Attribute page appears.
8. Click Finish.

   The Response is created.

More information:

Configure a Web Agent Response Attribute (see page 543)
Configure an Affiliate Agent Response Attribute (see page 545)
Configure a RADIUS Response Attribute (see page 544)

Configure Response Attributes

Each SiteMinder response may contain one or more response attributes. Response attributes identify the pieces of information that the Policy Server passes to a SiteMinder Agent. Each SiteMinder Agent type can accept different response attributes.

Note: More information on configuring an smetssocookie Web Agent active response attribute, which is needed for enabling single sign-on from SiteMinder to CA Single Sign-On, exists in Configure an smetssocookie Web Agent Active Response Attribute.

Response Attribute Types

SiteMinder supports different types of response attributes. The type of response attribute determines how SiteMinder provides appropriate content for the attribute.

You can specify the following types of response attributes when you add response attributes to a SiteMinder response:

Static

Returns data that remains constant.

Use a static attribute to return a string as part of a SiteMinder 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 entry in a user 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, configure the user directories on the SiteMinder User Directory pane.

DN Attribute

Returns profile information from a directory object in an LDAP, Microsoft SQL Server, or Oracle user directory.

User groups and Organizational Units (OUs) that are part of a user DN are examples of directory objects attributes that can be treated as DN attributes.

For example, you can use a DN attribute to return a company division for a user that is based on the user membership in a division.

Note: In order for the Policy Server to return values from DN attributes as response attributes, configure the user directories on the SiteMinder User Directory pane.

Active Response

Returns values from a customer supplied library that is based on the SiteMinder 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 conforms to the Authorization API (available separately with the Software Development Kit).

Note: Make sure that the returned value is valid. 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.

Session Variable

Returns the value of a session variable.

SiteMinder retrieves the value from the session store, or from memory when the response is part of the authentication request.

Expression

Allows the administrator to provide an expression.

For example, the administrator can configure a Response Attribute to extract a certain string from the Certificate issuerDN attribute and store it as a new session variable.

Configure a Web Agent Response Attribute

You can create a response attribute for a SiteMinder Web Agent by selecting SiteMinder and Web Agent on the Attributes group box on the Response pane. Web Agent response attributes support HTTP header variables, cookie variables, redirections to other resources, text, and timeout values.

Note: If you have purchased and installed SOA Security Manager, you can create a WebAgent-SAML-Session-Ticket-Variable response attribute. For more information, see the *CA SOA Security Manager Policy Configuration Guide*.

To create a response attribute

1. Click Create Response Attribute.

   The Create Response Attribute page appears.

2. Select a response attribute.

3. Select an attribute type.

   The details in the Attribute Fields are updated to match the specified attribute type.

4. Complete the details in the Attribute Fields.

   Note: A list of automatically generated SiteMinder user attributes that you can use in responses exists in SiteMinder Generated User Attributes (see page 550).

5. (Optional) Edit the attribute in the Script field.

   Note: The Attribute Setup section closes when you edit the attribute on the Advanced section.

6. Specify Cache Value or Recalculate value every ... seconds.

   Note: The maximum time limit that can be entered is 3600 seconds.

7. Click Submit.

   The Create Response Attribute Task is submitted for processing, and the response attribute is added to the Attribute List on the Response page.

Configure a RADIUS Response Attribute

You can create a response attribute for a RADIUS Agent by selecting RADIUS and a RADIUS vendor on the Attributes group box on the Response pane. RADIUS response attributes support any of the attributes supported by the RADIUS protocol.

To create a response attribute

1. Click Create Response Attribute.

   The Create Response Attribute page appears.

2. Select a response attribute.

3. Select an attribute type.

   The details in the Attribute Fields are updated to match the specified attribute type.

4. Complete the details in the Attribute Fields.

   Note: A list of automatically generated SiteMinder user attributes that you can use in responses exists in SiteMinder Generated User Attributes (see page 550).

5. (Optional) Edit the attribute in the Script field.

   Note: The Attribute Setup section closes when you edit the attribute on the Advanced section.

6. Specify Cache Value or Recalculate value every ... seconds.

   Note: The maximum time limit that can be entered is 3600 seconds.

7. Click Submit.

   The Create Response Attribute Task is submitted for processing, and the response attribute is added to the Attribute List on the Response page.

More information:

Configure a Web Agent Response Attribute (see page 543)

Configure an Affiliate Agent Response Attribute

You can create a response attribute for a SiteMinder Affiliate Agent by selecting SiteMinder and Affiliate Agent on the Attributes group box on the Response pane. Affiliate Agent response attributes support HTTP header variables and cookie variables. More information on Agent types exists in the *Web Agent Configuration Guide*.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

To create a response attribute

1. Click Create Response Attribute.

   The Create Response Attribute page appears.

2. Select a response attribute.

3. Select an attribute type.

   The details in the Attribute Fields are updated to match the specified attribute type.

4. Complete the details in the Attribute Fields.

   Note: A list of automatically generated SiteMinder user attributes that you can use in responses exists in SiteMinder Generated User Attributes (see page 550).

5. (Optional) Edit the attribute in the Script field.

   Note: The Attribute Setup section closes when you edit the attribute on the Advanced section.

6. Specify Cache Value or Recalculate value every ... seconds.

   Note: The maximum time limit that can be entered is 3600 seconds.

7. Click Submit.

   The Create Response Attribute Task is submitted for processing, and the response attribute is added to the Attribute List on the Response page.

More information:

Configure a Web Agent Response Attribute (see page 543)

Use 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 SiteMinder variable object is a name-value pair. SiteMinder 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.

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. SiteMinder 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 section.
3. Click Create Response Attribute on the Attribute List section.

   The Create Response Attribute pane opens.

4. Select a response attribute from the drop-down list on the Attribute Type section.

5. Select the type of response attribute on the Attribute Kind section.
6. Type the name of the variable object in the Variable Name field on the Attribute Fields section.

   Note: When this field is required, SiteMinder 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 section:

   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 check box.

   Active Response

   Specify the name of your library, the name of a library function. Optionally, specify 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.

   Session Variable

   Specify the name of a session variable for which an administrator can retrieve the value.

   Expression

   Specify an expression that extracts a value from an attribute and stores it as a new session variable.

   Note: SiteMinder uses the information that you provide in the fields on the Attribute Fields section 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:

Response Attributes (see page 538)
Select a Variable Using Variable Lookup (see page 549)

Select Users for Inclusion in a Response Attribute

The User Lookup pane allows you to select one user directory and search a list of users and user groups in that directory, selecting one set of users or user group for inclusion in a response attribute.

To select users for inclusion in a response attribute

1. Select DN Attribute as the Attribute Kind on the Attribute Setup group box.

   The Attribute Fields group box expands to include the DN Spec field.

2. Click Lookup on the Attribute Fields group box.

   The User Lookup pane opens.

3. Select the name of one user directory from the list, and click Search.

   The User Search pane opens.

4. (Optional) Select a Search type, and click GO:

   Attribute-value

   Specify an attribute name and value in the fields on the Users/Groups dialog.

   Expression

   Specify a search expression in the Expression field on the Users/Groups dialog.

   Note: You can click Reset to clear the search results.

5. Select one set of users or user group from the list, and click OK.

   The User Lookup pane reopens.

6. Click OK.

   The Response Attribute pane reopens, and the set of users or user group is added to the DN Spec field in the Attribute Fields group box.

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 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 SiteMinder 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: SiteMinder 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.

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 about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

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 about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

SiteMinder Generated User Attributes

The following list contains user attributes that SiteMinder generates automatically. These attributes can be specified as response attributes for Web Agent responses and are available to named expressions.

%SM_USER

The web agent places the username in an SM_USER http header variable for all requests. The web agent does not set the value of the SM_USER header variable when one fo the following items are true:

- A user does not provide a user name, such as with certificate–based authentication.
- A user name is not known.

%SM_USER_CONFIDENCE_LEVEL

If a user is authenticated with an authentication scheme and the authentication scheme generates a confidence level, this attribute holds an integer (0–1000). The authentication scheme inserts the integer in to the session ticket of the user. A higher confidence level corresponds to a higher level of credential assurance. A confidence level of zero represents no credential assurance. No credential assurance results in SiteMinder denying access to the requested resource.

Note: For more information, see Confidence Levels Introduced (see page 502).

%SM_USERDN

For an authenticated user, the web agent populates this http header variable with the DN that the Policy Server determines. For certificate-based authentication, this attribute can be used to identify a user.

%SM_USERNAME

For an authenticated user, this attribute holds the user DN that SiteMinder disambiguates. For an unauthenticated user, this attribute holds the user ID that a user specifies during a login attempt.

%SM_USERIMPERSONATORNAME

If the authentication scheme performs impersonation, this attribute holds the user DN that SiteMinder that authenticates.

%SM_USERLOGINNAME

This attribute holds the user ID that a user specifies during a login attempt.

%SM_USERIPADDRESS

This attribute holds the IP address of the user at the time of authentication or authorization.

%SM_USERPATH

For an authenticated user, this attribute holds a string that represents the directory namespace and directory server (both as specified in the user directory definition), and user DN (as SiteMinder disambiguates). For example:

"LDAP://123.123.0.1/uid=scarter,ou=people,o=airius.com"

For an unauthenticated user, this attribute holds the same value as SM_USERNAME.

%SM_USERPASSWORD

This attribute holds the password that the user specifies in the login attempt. This attribute is only available after a successful authentication through the OnAuthAccept event. The value is returned only on authentication, not on authorization.

%SM_TRANSACTIONID

This attribute holds the transaction ID that the agent generates.

%SM_USERSESSIONSPEC

The session ticket of the user.

%SM_USERSESSIONID

This attribute holds the session ID of a user who has already been authenticated, or the session ID that SiteMinder is to assign to the user upon successful authentication.

%SM_USERSESSIONIP

This attribute holds the IP address that was used during the original user authentication (upon establishment of a session).

%SM_USERSESSIONUNIVID

This attribute holds the universal ID of the user. If no universal ID directory attribute is specified in the user directory definition, the value defaults to the DN of the user.

%SM_USERSESSIONDIRNAME

This attribute holds the name of the user directory that the Policy Server is configured to use.

%SM_USERSESSIONDIROID

This attribute holds the object ID of the user directory that the Policy server is configured to use.

%SM_USERSESSIONTYPE

This attribute holds the session type of the user. The value is one of the following values:

- 2 - session
- 1 - identity

%SM_USERLASTLOGINTIME

This attribute holds the time, using GMT, that the user last logged in and was authenticated. This response attribute is only available for an OnAuthAccept authentication event. This attribute has value only when both of the following conditions are true:

- Password Services is enabled.
- The user has logged in through SiteMinder at least once.

%SM_USERPREVIOUSLOGINTIME

This attribute holds the time, using GMT, of the successful login before the last. This response attribute is only available for an OnAuthAccept authentication event. This attribute has a value only when Password Services is enabled.

%SM_USERGROUPS

This attribute holds the groups to which the user belongs. If the user belongs to a nested group, this attribute contains the group furthest down in the hierarchy. For all nested groups to which the user belongs, use SM_USERNESTEDGROUPS.

Example:

If a user belongs to the group Accounts Payable and Accounts Payable is contained in the group Accounting, SM_USERGROUPS contains Accounts Payable. If you want both Accounting and Accounts Payable, use SM_USERNESTEDGROUPS.

%SM_USERNESTEDGROUPS

This attribute holds the nested groups to which the user belongs. For only the group furthest down in the hierarchy, use SM_USERGROUPS[.

Example:

If a user belongs to the group Accounts Payable and Accounts Payable is contained in the group Accounting, SM_USERNESTEDGROUPS contains Accounting and Accounts Payable. If you want only Accounting, use SM_USERGROUPS.

%SM_USERSCHEMAATTRIBUTES

This attribute holds the user attributes associated with the DN or properties that are associated with the user. If the user directory is a SQL database, then SM_USERSCHEMAATTRIBUTES holds the names of the columns in the table where the user data is stored. For example, using the SmSampleUsers schema, SM_USERSCHEMAATTRIBUTES holds the names of the columns in the SmUser table.

%SM_USERPOLICIES

When a user is authorized for a resource and there are policies exist to give the user authorization, this attribute holds the names of the policies.

Example: To purchase an item, you are required to be a user that is associated with the Buyer policy. If the Policy Server authorizes me to buy an item, then SM_USERPOLICIES contains Buyer.

%SM_USERPRIVS

When a user is authenticated or a user is authorized for a resource, SM_USERPRIVS holds all of the response attributes for all policies that apply to that user, in all policy domains.

%SM_USERREALMPRIVS

When a user is authenticated or a user is authorized for a resource under a realm, SM_USERREALMPRIVS holds all the response attributes for all rules under that realm.

Example:

A realm exists named Equipment Purchasing. Under that realm, there is a rule named CheckCredit. The rule is associated with a response that returns the credit limit of the buyer, as a response attribute such as:

```
limit = $15000
```

If the buyer attempts to purchase equipment worth $5000, rule fires. SM_USERREALMPRIVS would contain all of the response attributes for all of the rules under the Equipment Purchasing realm.

%SM_AUTHENTICATIONLEVEL

When a user is authenticated for a resource, this attribute holds an integer number (of 0 to 1000) that represents the protection level of the authentication scheme under which the user was authenticated.

%SM_USERDISABLEDSTATE

This attribute holds a decimal number that represents a bit mask of reasons that a user is disabled. The bits are defined in SmApi.h under the Sm_Api_DisabledReason_t data structure, which is part of the SDK.

For example, a user may be disabled as a result of inactivity, Sm_Api_Disabled_Inactivity. In Sm_Api_DisabledReason_t, the reason Sm_Api_Disabled_Inactivity, corresponds to the value 0x00000004. So, in this case, SM_USERDISABLEDSTATE is 4.

A user can be disabled for multiple reasons.

%SM_USER_APPLICATION_ROLES

If you have purchased CA Identity Manager, this attribute may be used in responses. It contains a list of all roles assigned or delegated to a user. If an application name is specified, only the roles associated with the application are returned in the response attribute.

The response attribute name is typed in the Variable Name field on the Response Attribute pane. The response attribute name has the following syntax:

SM_USER_APPLICATION_ROLES[:*application_name*]

where *application_name* is an optional name of an application defined in Identity Manager.

The value for *application_name* must be communicated to the Policy Server administrator. Application names are not automatically passed to the Administrative UI.

Note: For more information about Identity Manager roles, see the CA *Identity Manager Operations Guide*.

%SM_USER_APPLICATION TASKS

If you have purchased CA Identity Manager (Identity Manager), this attribute may be used in responses. It contains a list of all tasks assigned or delegated to a user. If an application name is specified, only the tasks associated with the application are returned in the response attribute.

The response attribute name is typed in the Variable Name field on the Response Attribute pane. The response attribute name has the following syntax:

SM_USER_APPLICATION_TASKS[:*application_name*]

where *application_name* is an optional name of an application defined in Identity Manager .

The value for *application_name* must be communicated to the Policy Server administrator. Application names are not automatically passed to the Administrative UI.

Note: For more information about Identity Manager tasks, see the CA *Identity Manager Operations Guide*.

More information:

Configure a Web Agent Response Attribute (see page 543)

Availability of SiteMinder-generated Response Attributes

The following table shows the availability of SiteMinder generated response attributes during authentication, authorization and impersonation events:

Response Attribute	Authentication and Authorization Events					Impersonation Events
	GET/PUT	On Auth Accept	On Auth Reject	On Access Accept	On Access Reject	Impersonate Start User
SM_USER_CONFIDENCE_LEVEL	Yes	Yes	Yes	Yes	Yes	No
SM_USERNAME	Yes	Yes	Yes	Yes	Yes	No
SM_USERPATH	Yes	Yes	Yes	Yes	Yes	No
SM_USERIPADDRESS	Yes	Yes	Yes	Yes	Yes	No
SM_USERPASSWORD	No	Yes	Yes	No	No	No
SM_TRANSACTIONID	Yes	No	No	Yes	Yes	No
SM_USERSESSIONID	Yes	Yes	No	Yes	Yes	No
SM_USERSESSIONSPEC	Yes	No	No	Yes	Yes	No
SM_USERSESSIONIP	Yes	Yes	Yes	Yes	Yes	No
SM_USERSESSIONUNIVID	Yes	Yes	No	Yes	Yes	No
SM_USERSESSIONDIRNAME	Yes	Yes	No	Yes	Yes	No
SM_USERSESSIONDIROID	Yes	Yes	No	Yes	Yes	No
SM_USERSESSIONTYPE	Yes	Yes	No	Yes	Yes	No
SM_USERLASTLOGINTIME	No	Yes	No	No	No	No
SM_USERGROUPS[	Yes	Yes	No	Yes	Yes	No

Response Attribute	Authentication and Authorization Events					Impersonation Events
	GET/PUT	On Auth Accept	On Auth Reject	On Access Accept	On Access Reject	Impersonate Start User
SM_USERNESTEDGROUPS	Yes	Yes	No	Yes	Yes	No
SM_USERSCHEMAATTRIBUTES	Yes	Yes	Yes	Yes	Yes	No
SM_USERLOGINNAME	No	Yes	Yes	No	No	No
SM_USERIMPERSONATORNAME	No	No	No	No	No	Yes
SM_USERDISABLEDSTATE	Yes	Yes	No	Yes	Yes	No
SM_USERPOLICIES	No	No	No	Yes	No	No
SM_USERREALMPRIVS	Yes	No	No	No	No	No
SM_USERPRIVS	Yes	No	No	No	No	No

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 that you are creating an object. You can also copy the properties of an existing object to create an object.

To configure a response group

1. Click Policies, Domain.
2. Click Response Groups.

   The Response Groups page appears.
3. Click Create Response Group.

   The Create Response Group page appears. Verify that the Create a new object of type Resource Group option is selected.
4. Click OK.

   The Create Response Group: Select Domain page appears.
5. Select a domain and click Next.

   The Create Response Group: Define Response Group page appears.
6. Type the name and a description of the response group.
7. Select Radius or SiteMinder and an Agent Type.

   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.
8. In Group Members, click Add/Remove.

   The Response Group Members page appears.

   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 the Radius agents support.
9. 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 the additional members. To select a block of members, click the first member and then hold down the Shift key while you click the last member in the block.
10. Click OK.

    The selected responses are added to the response group.
11. Click Finish.

    The Response Group is created.

More information:

Duplicate Policy Server Objects (see page 54)

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.

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 about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

Delete a Response Group

Deleting a response group only deletes the grouping, not the individual responses contained in the group.

Note: More information about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

Chapter 20: Policies

This section contains the following topics:

Policy Overview

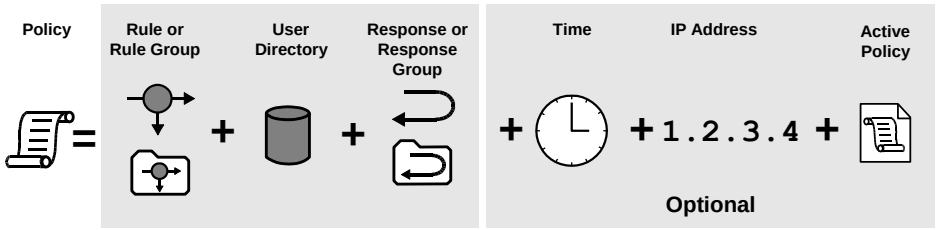
Policies define how users interact with resources. When you create policies in the Administrative UI, you link together (bind) objects that identify users, resources, and actions associated with the resources.

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.

SiteMinder 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 SiteMinder 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 customize content. 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 SiteMinder to be included in a policy definition. Active policies allow SiteMinder to interact with custom software created using the SiteMinder APIs.

More information:

Domains (see page 493)
Rules (see page 513)
Responses and Response Groups (see page 535)
User Directories (see page 157)
Allowable IP Addresses for Policies (see page 580)
Time Restrictions for Policies (see page 583)
Configure an Active Policy (see page 584)

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 SiteMinder 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 SiteMinder to authorize the user.

More information:

Authentication Schemes (see page 301)
Policy Binding Establishment (see page 585)

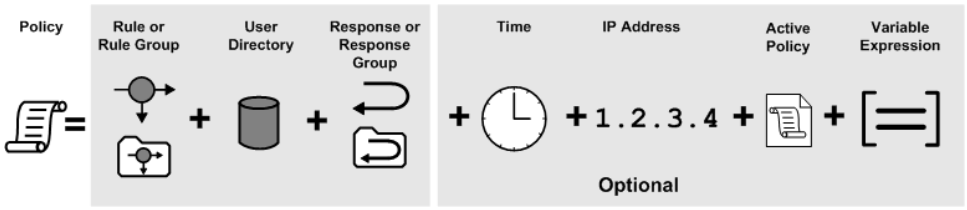
Expressions in Policies

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 578).

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 607)

Confidence Levels in Policies

If SiteMinder is integrated with a supported risk analysis engine, a confidence level is available for use in policies. Confidence levels extend policies to include the results of the risk evaluation that is completed as part of user authentication. The Policy Server can use these results when making authorization decisions.

You can apply a confidence level to the following objects:

- A policy realm.

  A confidence level that you configure in a realm applies to all resources (rules) that are associated with the realm. Applying a confidence level to a policy realm requires that you enable confidence level support. For more information, see the *SiteMinder Implementation Guide*.

- An active policy expression.

  A confidence level that you configure as an active policy expression applies only to those resources (rules) that are bound to the policy. The active policy expression allows for more granular authorization decisions. Using an active policy expression to apply a confidence level remains supported from previous releases and is enabled by default.

Note: SiteMinder supports an integration with an on–premise implementation of CA Arcot WebFort and CA Arcot Risk, and a hosted implementation of CA Arcot A–OK. For more information, see the *SiteMinder Implementation Guide.*

More information:

Confidence Levels Introduced (see page 502)

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.
2. Add users to the policy (see page 567).
3. Add one or more rules to the policy (see page 568).
4. (Optional) Associate responses or response groups with rules (see page 568).

5. (Optional) Associate global responses with rules (see page 569).
6. (Optional). Configure advanced policy options (see page 580).

More information:

Add Users to a Policy (see page 567)
Add Rules to a Policy (see page 568)
Associate a Rule with a Response or Response Group (see page 568)
Associate a Rule with a Global Response (see page 569)
Advanced Policy Options (see page 580)

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 Policies, Domain.
2. Click Domains.

   The Domains page appears.
3. Specify search criteria, and click Search.

   A list of domains that match the search criteria appears.
4. Click the name of the domain you want to modify.

   The View Domain page appears.
5. Click Modify.

   The settings and controls become active.
6. Click the Policies tab.

   The Policies page appears.
7. Click Create.

   The Create Policy page appears.
8. Type the name and a description of the policy.
9. Click the Users tab.

   The User Directories page appears.

10. Add users, user groups, or both to the policy, and click Submit.

    The Modify Domain: *Name* page re-appears.

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 242)
Policy Binding Establishment (see page 585)

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 you want to occur when the rule fires.

Note: Add at least one rule or rule group to a policy.

Follow these steps:

1. Click the Rules tab on the Policy pane.

   The Rules dialog 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 section 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 section. To create a 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 641)

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.

Add a Confidence Level to a Policy

Adding a confidence level to a policy lets you apply the results of an RiskMinder risk score evaluation to an authorization decision. Using an active expression limits the confidence level to only those resources (rules) bound to the policy. For RiskMinder risk scores, lower numbers indicate less risk and a safer transaction. For SiteMinder confidence levels, higher numbers indicate less risk and a safer transaction.

Follow these steps:

1. From the Administrative UI, click Policies, Domain, Domains.
2. Click the Edit icon for the policy domain you created for your RiskMinder environment.
3. Click the Policies tab.
4. Click Create.
5. Click the Policies tab.
6. Click the edit icon for the policy.
7. Complete the following steps in the Active Policy Expression area:
   a. Enter the following library name:

      `smriskactiveexpr`

   b. Enter the following function name:

      `CheckConfidenceLevel`

   c. Enter a confidence level in the Function Parameters field. The valid range is 1 through 1000.
8. Click OK.
9. Click Submit.

   The confidence level is applied to the resources (rules) bound to the policy.

More information:

Confidence Levels Introduced (see page 502)

Add CA Identity Manager Roles

If SiteMinder is integrated with a CA Identity Manager, a CA Identity Manager role is available for use in policies. Roles let the Policy Sever make authorization decisions for users who are members of CA Identity Manager roles.

Follow these steps:

1. Click the Users tab on the Policy screen.
2. Click Add Roles from the IDM Environment you want.
3. Select the roles you want and click OK.
4. Click Submit.

   The CA Identity Manager roles are added to the policy.

Exclude CA Identity Manager Roles

If a user who is a member of an excluded CA Identity Manager role tries to access a protected resource, the Policy Server:

- Verifies that the user is a member of the excluded role.
- Blocks access to the resource.

Follow these steps:

1. Click the Users tab on the Policy screen.
2. Locate the roles you want to exclude in the IDM Environments section.
3. For each role, click Exclude.
4. Click Submit.

   The CA Identity Manager roles are excluded from the policy.

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.

- 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 SiteMinder 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 SiteMinder API. The function call determines whether or not the policy fires.

More information:

Enable and Disable Policies (see page 579)

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.

More information:

How the Web Agent and Policy Server Calculate Time (see page 61)
Add Time Restrictions to Rules (see page 528)

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 SiteMinder 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	SiteMinder can be configured to associate Policies with custom directory objects.

Generally, you bind users or user attributes to policies on the SiteMinder 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 567)

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 SiteMinder Policy Users/Groups dialog. The second is by using the Search feature in the SiteMinder Policy Users/Groups dialog.

To bind users 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 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 576)
Add Users to a Policy (see page 567)

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 576)
Add Users to a Policy (see page 567)
Search User Directories (see page 242)

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 578)

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

SiteMinder 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 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 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 578)

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 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 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 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 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 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 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 578)

Binding Policies to Custom Object Classes

SiteMinder 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 SiteMinder 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 567)

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 576)
Add Users to a Policy (see page 567)

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 576)
Add Users to a Policy (see page 567)
Search User Directories (see page 242)

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 567)

Policy Bindings for Microsoft SQL Server and Oracle User Directories

When SiteMinder 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 567)

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 576)
Add Users to a Policy (see page 567)

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 576)
Add Users to a Policy (see page 567)
Search User Directories (see page 242)

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 567)

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 about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

Bind Policies to SQL Queries

SiteMinder 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 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 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 565)

Policy Processing

The Policy Server use policies to authorize access to resources and provide entitlements to authorized users. The resources protected by policies are usually stored in a hierarchy that mimics security, organizational, and business requirements and takes advantage of the SiteMinder's hierarchical or "nested" realms. The remainder of this chapter discusses how SiteMinder processes policies and gathers entitlements in a hierarchical structure.

More information:

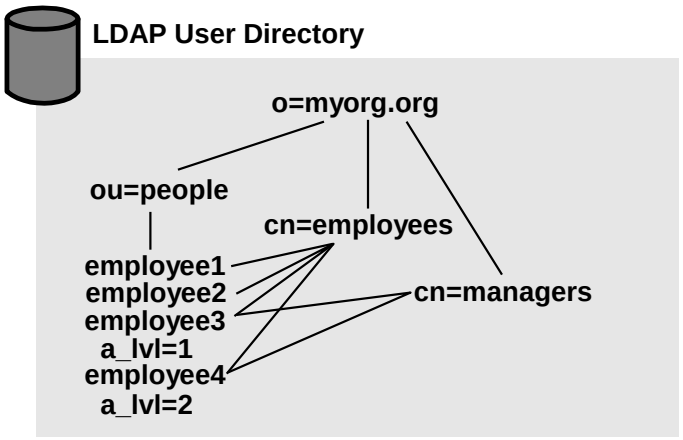
Nested Realms (see page 505)

Sample SiteMinder Configuration with Nested Realms

This section explains the sample configuration that will be used in examples in the remainder of this chapter.

User Directory

Assume that the policy domain that contains the policies and other relevant Policy Server objects includes a connection to the LDAP user directory in the following diagram.



The sample user directory contains the following:

o=myorg.org

This is an organization.

ou=people

This is an organizational unit that contains information for all employees.

employee<*n*>

These are directory entries for each employee. Note that a_lvl is a user attribute that indicates an access level. For the purpose of the examples in this section, assume that employee1 and employee2 have an access level of zero (a_lvl=0).

cn=employees

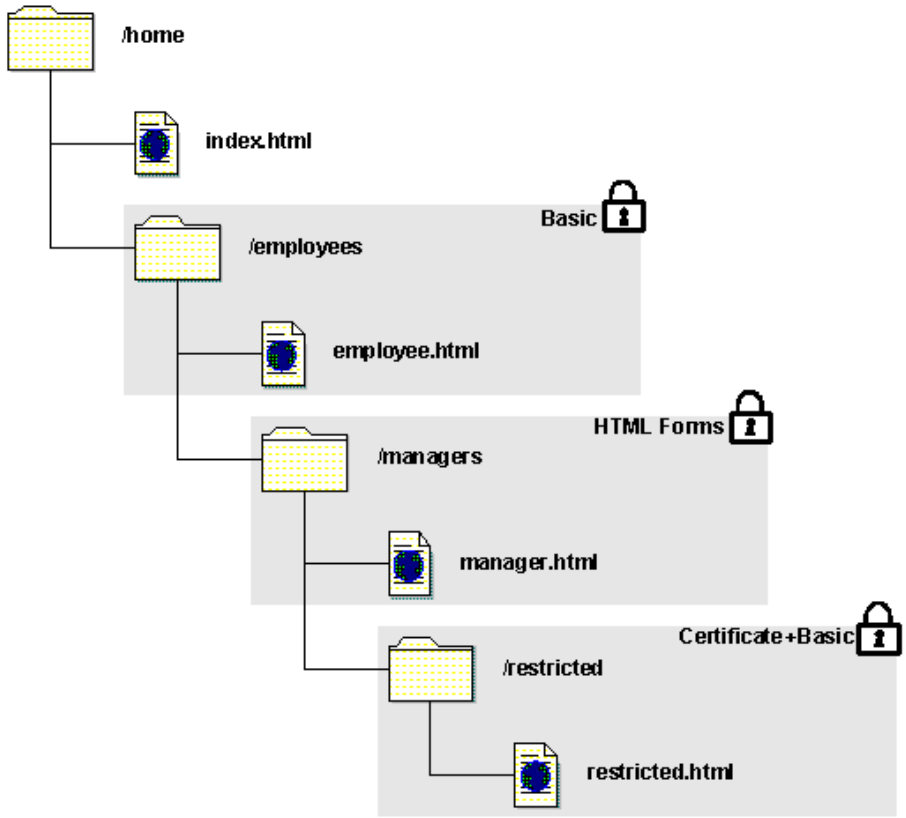
This is a group that contains all company employees as its members.

cn=managers

This is a group that contains all employees with a managerial title as its members. Note that employee3 and employee4 are the only employees in this group, and their respective access levels are greater than zero.

Nested Realms and Resources

The structure of nested realms and resources used in the examples in this section are illustrated in the following diagram. The shaded areas indicate realms that are protected by an authentication scheme.



The nested realms are made up of the following directories and resources:

/home

This directory is the top level of the sample nested realm. This realm specifies /home/ as its resource filter, and contains the unprotected resource of index.html. The index.html file is not protected by an authentication scheme.

/employees

This directory, contained in the /home directory, is protected by a Basic authentication scheme, which requires a user name and password as credentials. The realm associated with this directory uses a resource filter of employees/ and contains the employee.html file. The total Resource Filter for this realm is /home/employees/.

/managers

This directory, contained in the /employees directory, is protected by an HTML Forms authentication scheme which requires a user name, password, and additional personal information as credentials. The realm associated with this directory uses a resource filter of managers/ and contains the manager.html file. The total Resource Filter for this realm is /home/employees/managers/.

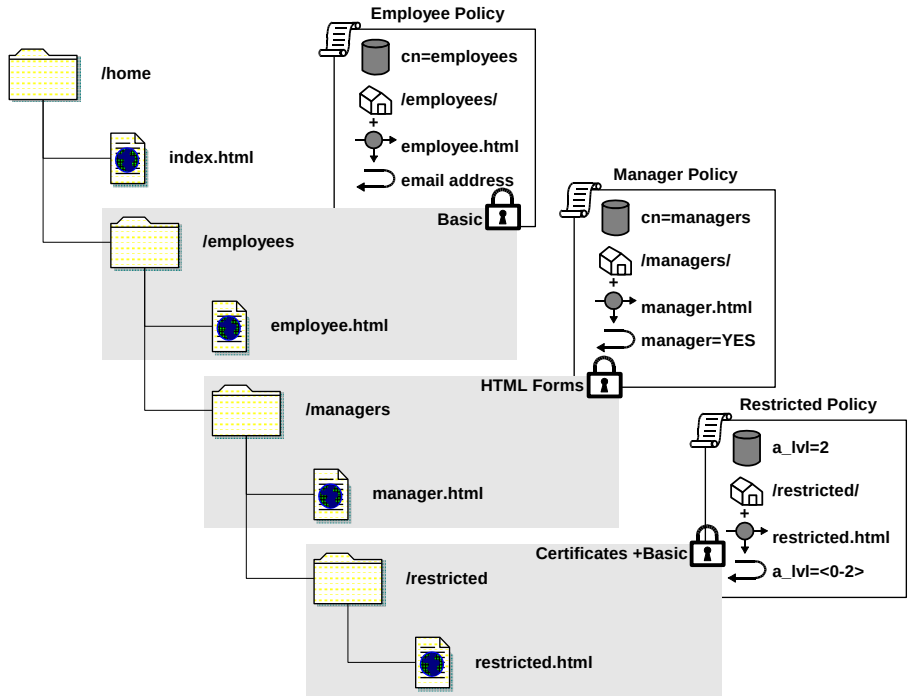
/restricted

This directory, contained in the /managers directory, is protected by an X.509 Certificate + Basic authentication scheme which requires user name, password, and a valid X509 certificate as credentials. The realm associated with this directory uses a resource filter of restricted/ and contains the restricted.html file. The total Resource Filter for this realm is /home/employees/managers/restricted/.

In these examples the protection level of the /employees realm is less than the protection level of the /managers realm, which is less than the protection level of the /restricted realm.

Policies and Responses

The policies and responses used in the examples in the remainder of the chapter are illustrated in the following diagram and described below.



The following is a description of each of the sample policies and the objects contained in each policy.

Employee Policy

This policy contains a Get rule that protects the employee.html resource. This resource is located in the /employees realm. The policy binds the user group cn=employees, so that all employees in the LDAP directory can access the resource once they are successfully authenticated. When an authenticated user is authorized by this policy, SiteMinder returns a response of the user's email address. For example, if employee1 attempts to access /home/employees/employee.html and is successfully authenticated, the Policy Server allows employee1 to access the resource and returns the email address:

employee1@myorg.org

A Web application can use this response for customization when accessing other company resources.

Manager Policy

This policy contains a Get rule that protects the manager.html resource. This resource is located in the /manager realm. The policy binds the user group cn=managers so that only employees contained in cn=managers group can access the resource once they are successfully authenticated. When an authenticated manager is authorized by this policy, SiteMinder returns a static response. In the example, if employee3 attempts to access /home/employees/managers/manager.html and is successfully authenticated, the Policy Server allows employee3 to access the resource and returns the following response:

manager=YES

An application can use this response to activate features that are only available to company managers.

Restricted Policy

This policy contains a Get rule that protects the restricted.html resource. This resource is located in the /restricted realm. The policy binds only the employees in the directory who have an access level user attribute of two (a_lvl=2). Managers with the correct access level can access the resource once they are successfully authenticated. When a user attempts to access the restricted.html resource, SiteMinder returns a response of a_lvl=*<0-2>*. For example, if employee4 attempts to access /home/employees/managers/restricted/restricted.html and is successfully authenticated, the Policy Server allows employee4 to access the resource and returns the following response:

a_lvl=2

An application can use this response to activate features that are only available employees with an access level of two.

Authentication Processing for Hierarchical Policies

Policies must contain rules. Rules can include authentication and authorization events. Based on how rules are configured, one of four authentication events can occur when the Policy Server attempts to identify a user based on credentials.

OnAuthAccept

The authentication is successful.

OnAuthAttempt

The authentication fails because the user is not found in any directory in the policy domain's search order.

OnAuthReject

The user *is found in a directory*, but the authentication fails because the credentials supplied by the user are incorrect. If this occurs, the Policy Server looks in the next directory in policy domain's directory search order. If the user's credentials cannot be verified in any of the directories in the search order, the Policy Server processes OnAuthReject events.

The user must be found in a directory for an OnAuthReject rule to fire. If the user is not found in any directory, an OnAuthReject rule will not fire.

OnAuthUserNotFound

The authentication server has a valid session ticket, yet it cannot find the user directory. This situation should only occur in a single sign-on environment that uses multiple directories, though it may take place if a user was idle long enough to be removed from the Agent's cache, and the user was removed from the directory. When this event occurs, the Policy Server evaluates the user's existence in the directory rather than relying on the session ticket.

The Policy Server attempts to authenticate users based on the longest matching realm. For example, if a user attempts to access /home/employees/managers/manager.html, the Policy Server uses the /managers realms to determine the required credentials. In the example in the previous figure, the user must complete a browser-based form required by the HTML Forms authentication scheme associated with the realm.

Note: The longest matching realm also determines the timeouts for the user's session. If a timeout is associated with the realm in which the user successfully authenticated, that timeout is used. You can use responses to override a realm timeout for a specific resource or group of resources.

More information:

Configure Response Attribute Caching (see page 549)

Successful Authentications

OnAuthAccept rules fire when all of the following are satisfied:

- A user enters credentials that satisfy the requirements of the longest matching realm.
- The Policy Server finds the user in a user directory.
- The Policy Server verifies the user's credentials.

At this point, the Policy Server collects OnAuthAccept entitlements from the longest matching realm, and any realms above it in the hierarchy of nested realms. In the example in the previous figure, if a user is successfully authenticated for the resource /home/employees/managers/manager.html, the Policy Server collects any OnAuthAccept entitlements for the /managers realm, then the /employees realm, and finally the /home realm.

Rejected Authentication Attempts

Policy domains are configured with a directory search order. When the Policy Server attempts to authenticate a user, it searches each user directory in the search order until it finds the user and verifies the supplied credentials. If the Policy Server locates a user in a directory, but the credentials supplied by the user do not match, the Policy Server looks at the next directory in the search order. If the Policy Server does not find a match for the user in any directory, the user's authentication attempt fails in the context of the realm that contains the requested resource.

For example, if a user attempts to access /home/employees/managers/manager.html, and the user is located in a user directory, but fails to provide valid credentials for any directory in the search order, the authentication event fails in the /managers realm. The Policy Server then processes any events for a rejected authentication attempt in that realm (OnAuthReject).

More information:

Domains and User Membership (see page 494)

Unsuccessful Authentication Attempts

If the Policy Server cannot find the user who attempts to authenticate in any of the directories in the directory search order, the authentication fails in the context of the realm that contains the requested resource. The Policy Server then processes any events for a failed authentication attempt (OnAuthAttempt).

Authorization Processing for Hierarchical Policies

Policies can contain rules that allow access to resources and may also include rules that trigger SiteMinder events. The possible authorization events include the following:

OnAccessAccept

This type of event occurs when a user is successfully authorized.

OnAccessReject

This type of event occurs when an authenticated user is denied access to a resource.

If a rule does not specify an authorization event, the rule either allows or denies access to the resource.

Policy Processing for Authorized Users

For the Policy Server to authorize a user to access a resource in a nested realm, the user must be authorized in each realm from the top of the hierarchy to the realm that contains the resource. Once a user is authenticated, the Policy Server checks the policies governing each realm above the resource to make sure the user is authorized. In the previous example, if employee3 wants to access /home/employees/managers/manager.html, the Policy Server verifies that employee3 is authorized in /home, /employees, and finally, /managers.

Once the Policy Server determines that the user is authorized, it collects entitlements for the user from each of the nested realms. Next, the Policy Server processes OnAccessAccept events starting at the top of the realm hierarchy and moving through the longest matching realm. In the previous example, the responses returned by the Policy Server would include employee3's email address, as well as the static response manager=YES.

Policy Processing for Unauthorized Users

Policy processing for unauthorized users is the same, whether or not the user is explicitly denied access by a policy, or simply not contained in a policy that allows access. Any entitlements collected so far for that user are lost and the Policy Server processes OnAccessReject events in the context of the rejecting realm where the requested resource is located.

In the previous example, if employee1, who is not a member of the cn=managers group, attempts to access /home/employees/managers/manager.html, the Policy Server only processes OnAccessReject events for the /managers realm.

Directory Mapping for Hierarchical Policies

In a SiteMinder configuration that contains nested realms, each realm may contain a directory mapping which specifies an authorization directory to be used which is different from the authentication directory. This allows companies to use a single directory for authenticating users, while distributing authorization directories throughout an enterprise. When the Policy Server processes entitlements in a group of nested realms, the it checks for directory mappings for each realm in the hierarchy. If there is no directory mapping, the Policy Server uses the authentication directory as the authorization directory.

Chapter 21: Variables

This section contains the following topics:

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 SiteMinder response objects.

Component Requirements for eTelligent Rules

The following components are required to use eTelligent Rules:

- A Web Agent
- (Optional) The Web Agent Option Pack

  The Web Agent Option Pack is required only if you plan on using POST variables.

Note: More information on installing a Web Agent exists in the *Web Agent Installation Guide*. More information on installing the Web Agent Option Pack exists in the *Web Agent Option Pack Guide*.

SiteMinder eTelligent Rules Benefits

- Reduce complexity and eliminate the need for custom code.

  Authorization access is defined by the SiteMinder 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 SiteMinder 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 SiteMinder 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 611)
Policies (see page 561)
eTelligent Rules Properties Files (see page 609)

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 609)
Modify the LoggerConfig.properties File (see page 609)

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 parameter 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 612)
- Request Context Variables (see page 612)
- User Context Variables (see page 612)
- Form Post Variables (see page 612)
- Web Services Variables (see page 613)

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 SiteMinder 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.

Web Services Variables

Web Services variables can be used to capture information retrieved from a Web Service for use in policies or responses. The key benefit of this type of variable is to allow a Policy Server administrator to define a policy based on the dynamic customer information provided in real time by a Web Service.

More information:

Web Service Variables (see page 616)

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 564)

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.

How the Policy Server Processes Variables

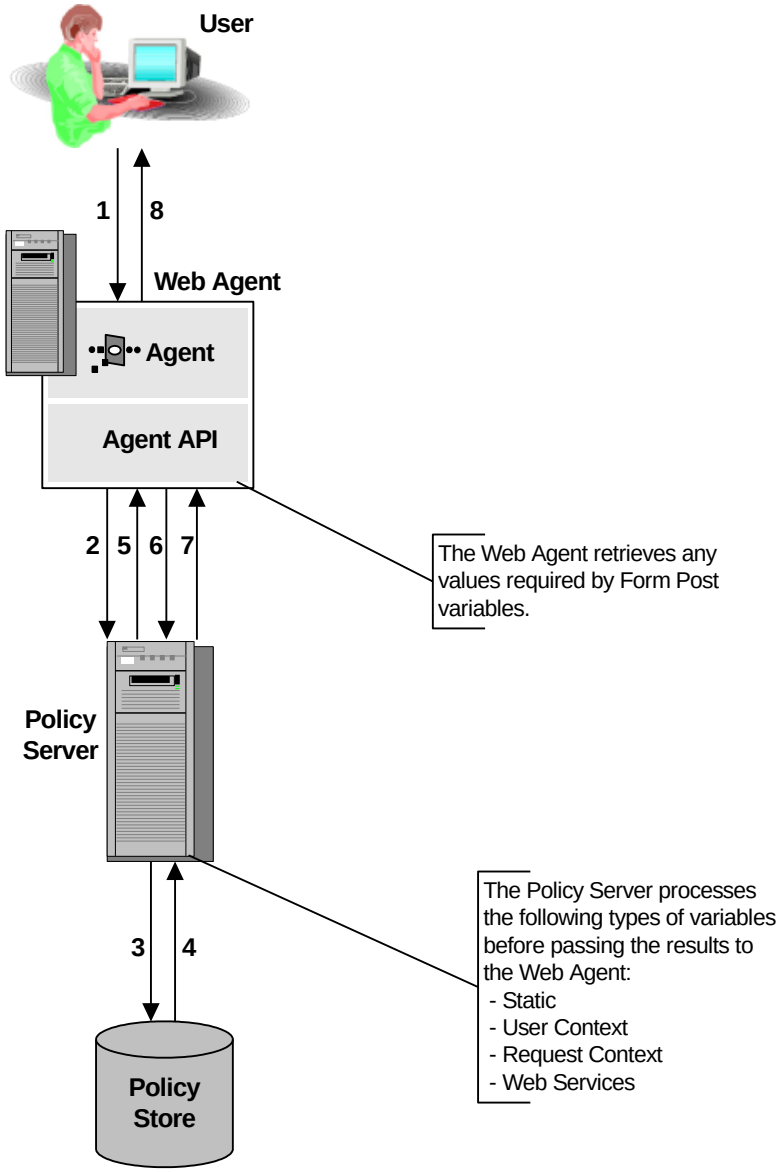
Variables are evaluated when the Policy Server processes an authorization for a request and determines that a user is authorized for the requested resource. The details of variable processing are slightly different based on whether the variable objects are contained in a response or in a policy expression.

How the Policy Server Processes Variables Contained in Policy Expressions

As part of policy evaluation, the variables contained in a policy expression are placed in an unresolved variable list during the authorization of a request. As the Policy Server resolves variables, they are moved to a resolved variables list. When all variables in a policy expression have been resolved, the Policy Server grants or denies access based on the entire policy.

The following figure illustrates how the authorization of a user's request is processed by a Web Agent and the Policy Server when the policy for the requested resource contains variables. This diagram does not include Web Service variables.

Note: The process in the following figure assumes that the user has already been authenticated by SiteMinder. For unauthenticated users, the authentication process must occur before the authorization.



1. The user requests a resource from a server that is protected by a SiteMinder Web Agent.
2. The Agent verifies that the resource is protected and the Policy Server begins authorization processing.
3. The Policy Server retrieves policy information from the Policy Store about the requested resource.

4. The Policy Server receives a list of unresolved variables contained in the policy expression associated with the requested resource. The Policy Server evaluates static, user context, and request context variables contained in the unresolved variables list.
5. If all variables and variable expressions have been resolved, the Policy Server indicates to the Web Agent whether or not the user may access the requested resource.
6. If the unresolved variables list still contains unresolved variables, the list is passed to the Agent API layer with a Not Resolved indicator. The values of any Form Post variables are resolved by the Web Agent and passed to the Policy Server in a new request that includes the Form Post variable value in the resolved variables list.
7. If the policy contained Form Post variables, the Policy Server processes the policy with the newly resolved values extracted from the POST data.
8. The user is either allowed or denied access to the requested resource.

More information:

Web Service Variables (see page 616)

How the Policy Server Processes Variables contained in Responses

SiteMinder processes variables contained in responses as described in the previous section. Since Form Post variables cannot be used for responses, all variables are resolved by the time a response fires at the Policy Server.

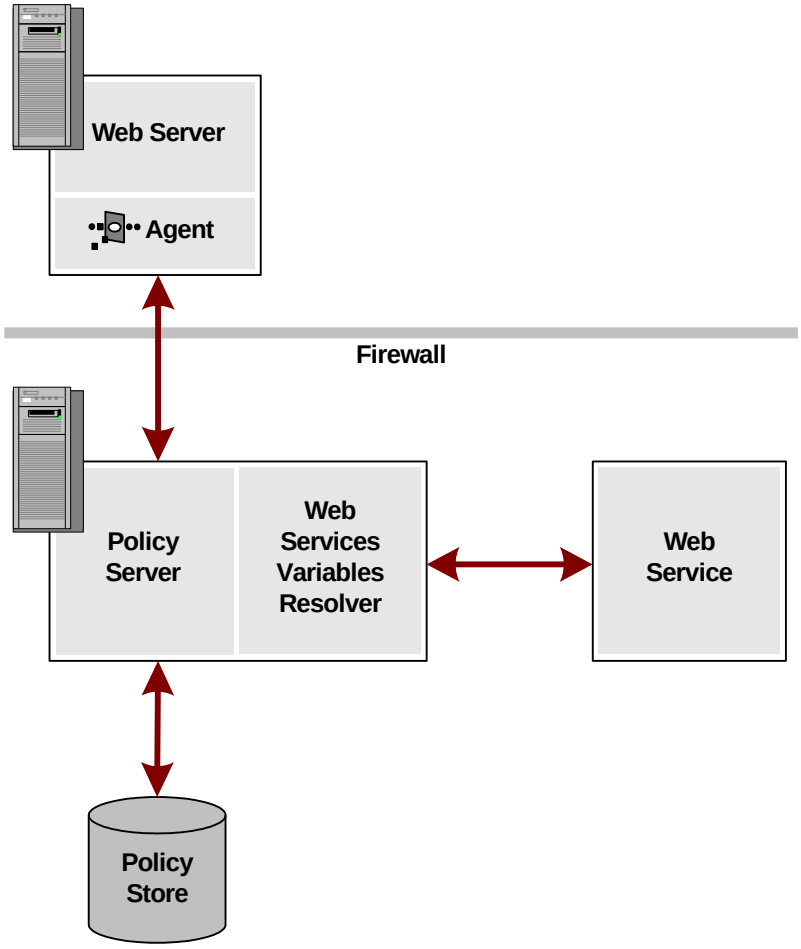
Web Service Variables

Web service variables provide a method for including dynamic data from a web service in a SiteMinder policy. Web service variables are resolved by calling a web service. The Policy Server sends a SOAP request document, as specified in the web service variable definition, and receives a SOAP response document as a reply. The Policy Server extracts the value of the web services variable from the SOAP response document.

The Simple Object Access Protocol (SOAP) is a lightweight, XML-based protocol that consists of three parts:

- An envelope that defines a framework for describing message contents and how to process it.
- A set of encoding rules for expressing instances of application-defined data types.
- A convention for representing remote procedure calls and responses.

The following figure shows how a SiteMinder deployment resolves a web services variable for a web service inside an Intranet. The web service is on the same side of the firewall as the Policy Server.



In this scenario, if a Web Service variable is associated with an authorization request, it is resolved on the Policy Server side by calling the Web Service Variables Resolver. The Web Service Variables Resolver runs in the same process space.

When defining the Web Service variable, the user specifies the SOAP document to send to the Web Service, the authentication credentials, and other parameters.

The resolver sends the specified SOAP document to the web service, extracts the value of the variable from the response and forwards it to the Policy Server to complete the authorization request.

Even if there is a firewall between the Policy Server and the web service, it can be configured to allow communication between the two. The Policy Server issues the request and reads the response, so the firewall is only required to allow outbound requests from the Policy Server to the web service.

A secure SSL connection can be configured between the Policy Server and the web service to allow for the inbound responses to come from the Web Service to the Policy Server. The SSL connection uses the server-side certificates on the web service and a list of trusted certificate authorities that are configured on the Policy Server side.

Component Requirements for Web Service Variables

Web service variables require a session store.

Note: More information about configuring a session store, see the *Policy Server Installation Guide*. For more information about upgrading a session store, see the *SiteMinder Upgrade Guide*.

Security Requirements When Resolving Web Services Variables

Security for Web Services Variables requires an SSL connection between the Policy Server and the Web Service. You can also include a WS-Security header with a username token that the Web Service has been configured to recognize. WS-Security is a standard set of SOAP extensions that provides security token propagation, message integrity and confidentiality through signing and encryption.

For a secure resolution of a Web Services Variable:

- The Policy Server must make a SOAP request to a Web Service on behalf of a known identity (a Web Service account). The model is similar to the model that SiteMinder uses to access attributes in a user directory.
- The WS-Security header containing the Web Services credentials can be configured to include a base-64-encoded SHA-1 digest of the password.
- A Web Service variable can be configured to use an SSL connection with a server-side certificate, which means that the Policy Server must be configured with the list of trusted CAs.

Note: For SSL connections, configure server–side certificates for the Web Service. Configure a list of trusted CAs on the Policy Server. To configure trusted CAs, use the certificate data store described in Certificate Authorities and Web Services Variables.

Configure the Web Service Variable Resolver

In order for the Policy Server to resolve a Web Service variable, you must configure the Web Service Variable Resolver to properly connect to the Web Service. The connection to the Web Service will fall into one of two categories:

- Both the Policy Server and the Web Service are on the same side of a firewall. There is no firewall between the two.
- There is a firewall between the Policy Server and the Web Service, but it is configured to allow one-way communication between the two (outbound requests from the Policy Server to the Web Service).

Before being able to use the Web Service Variables functionality, the Policy Server must be configured with a list of trusted CAs, using the SmKeyTool command line utility. If several Policy Servers are used in a load balancing or failover configuration, each of them must be configured with the same list of trusted CAs.

Default configuration settings are provided in the WebServiceConfig.properties file in the SiteMinder/Config/properties directory, and can be modified by the user.

Sample WebServiceConfig.properties Configuration File

```
# Netegrity Web Service Variable Resolver properties configuration file:
# This file must be in the classpath that is used when the policy server runs.
# ResolutionTimeout is the amount of time the resolver will at most wait to resolve
all Web Service variables related to a given request.
#
# This setting is intended to end sessions that are waiting on a web service that is
not responding. The time that the Web Agent will typically wait before responding is
typically 60 sec (but may be changed # in the future), which means this setting should
be 60000 or greater to cancel transactions that cannot be returned.
ResolutionTimeout=75000
# MaxThreadCount is the maximal number of active threads running within the Web Service
variables resolver.
MaxThreadCount=10
```

Certificate Authorities and Web Services Variables

To use SSL connections when resolving web services variables, configure a list of trusted Certificate Authorities (CAs). The Policy Server uses the CAs when it establishes a connection to a Web Service.

Use the Administrative UI to configure the list in the certificate data store.

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 493)

Create a Static Variable

You can create a static variable to make it available for use in policies or responses.

Note: The value of the resolved variable must not be greater than 1K.

To create a variable

1. Click Policies, Domain.
2. Click Variables.

   The Variables page appears.
3. Click Create Variable.

   Verify that the Create a new object of type Variable option is selected.
4. Click OK.

   The Create Variable :Select Domain page appears.
5. Select a domain from the list and click Next.

   The Create Variable: Define Variable page appears.
6. Type the variable name in the Name field.

7. 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.

8. Specify the data type and value of the variable in Variable Information.
9. 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 can create a request context variable to make it available for use in policies or responses.

Note: The value of the resolved variable must not be greater than 1K.

To create a variable

1. Click Policies, Domain.
2. Click Variables.

   The Variables page appears.

3. Click Create Variable.

   Verify that the Create a new object of type Variable option is selected.

4. Click OK.

   The Create Variable :Select Domain page appears.

5. Select a domain from the list and click Next.

   The Create Variable: Define Variable page appears.

6. Type the variable name in the Name field.

   Note: Request Context variable names must begin with the percent character (%).

   Example: %REQUEST_ACTION

7. Select Request Context from the Variable Type list.

   Request context settings open.

8. Select the variable value from the Property list.
9. 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 must not be greater than 1K.

To create a variable

1. Click Policies, Domain.
2. Click Variables.

   The Variables page appears.
3. Click Create Variable.

   Verify that the Create a new object of type Variable option is selected.
4. Click OK.

   The Create Variable :Select Domain page appears.
5. Select a domain from the list and click Next.

   The Create Variable: Define Variable page appears.
6. Type the variable name in the Name field.

   Note: User Context variable names must begin with the percent character (%).

   Example: %SM_USERPATH
7. Select User Context from the Variable Type list.

   User context settings open.
8. Select the portion of the user context that provides the value of the variable from the Item list.
9. (Required for Session Variable) Specify the type of data represented by the variable (Boolean, Number, String, or Date) in the Return Type field.

   For other Item list selections, the Return Type value is preset as String or Boolean as appropriate and not user-configurable.
10. (Required for User Property, Directory Entry, and Session Variable) Enter the name of the directory or user attribute that provides the variable value in the Property field.
11. (Required for User Property, Directory Entry, and Session Variable) Enter the size of the buffer (in bytes) that is to store the variable in the Buffer field.
12. (Required for Directory Entry) Enter the distinguished name of the directory entry in the DN field.
13. 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 Form Post Variable

You can create a Form Post variable to make it available for use in policies.

Note: The value of the resolved variable must not be greater 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 screen appears.
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.

Create a Web Services Variable

You create a Web Services variable to make it available for use in policies or responses.

Note: The value of the resolved variable must not be greater than 1K.

To create a variable

1. Click Policies, Domain.
2. Click Variables.

   The Variables page appears.
3. Click Create Variable.

   Verify that the Create a new object of type Variable option is selected.
4. Click OK.

   The Create Variable :Select Domain page appears.

5. Select a domain from the list and click Next.

   The Create Variable: Define Variable page appears.

6. Type the variable name in the Name field.

7. Select Web Service from the Variable Type list.

   Web Service settings appear.

8. Select the data type from the Return Type list.

9. Type the Web Service URL in the URL field.

10. Type the XPath query in the XPath field.

    Note: The Policy Server uses this query to extract the value of the Web Service variable from the SOAP document returned by the Web Service.

11. (Optional) Select the Require Credentials option in Web Service Credentials and specify the user name and password that the Policy Server is to use when connecting to the Web Service.

12. (Optional) Click the following button in the SOAP Document section to add existing variables to the SOAP message:

    Variable

13. (Optional) Click Add in HTTP Headers to associate an HTTP header with the Web Service variable.

14. Click Finish.

    The variable appears on the Variables tab of the domain and can now be used in policy expressions or responses.

Chapter 22: Key and Certificate Management

Certificate and Private Key Usage by SiteMinder

Private keys and certificates are required for the following tasks:

- Federation components use private key/certificate pairs for signing, verification, encryption, and decryption of entire assertions, or specific assertion content.
- Federation components employ client certificates for back-channel authentication for artifact single sign-on.
- For establishing SSL connections (SSL server certificates).

Private key/certificate pairs and single certificates for federation functions are stored in the certificate data store (CDS). The certificate data store is collocated with the policy store. All Policy Servers that share a common view into the same policy store have access to the same keys, certificates, CDS-configured certificate revocation lists (CRL), and OCSP responders.

SSL server certificates are stored on the web server where they are installed. SSL server certificates are not stored in the certificate data store.

Each key/certificate pair, client certificate, and trusted certificate in the certificate data store must have a unique alias. The alias lets SiteMinder reference any private key/certificate pair or single certificate in the certificate store. The certificate data store can store multiple key/certificate pairs and single certificates. In a federated environment, you can have multiple partners. For multiple partners, you can use a different pair for each partner.

If a signing alias is configured for signing assertions, the assertion generator uses the key that is associated with that alias to sign assertions. If no signing alias is configured, the assertion generator uses the key with the *defaultenterpriseprivatekey* alias to sign assertions. If the assertion generator does not find a default enterprise private key, it uses the first private key it finds to sign assertions.

Important! If you are going to store multiple keys, define the first key that you add with the defaultenterpriseprivatekey alias before adding subsequent keys.

A given Policy Server can sign or sign and verify responses. You can add keys and certificates for signing and validation to the same certificate data store.

You manage the contents of the certificate data store using the Administrative UI.

The following types of key/certificate pairs and single certificates are stored in the certificate data store:

Function	Private Key/Cert Pair	Certificate (public key)	CA Certificates	Client Certificate
Signs assertions, authentication requests, SLO requests and responses	X			
Verifies signed assertions, authentication requests, and SLO requests/responses		X		
Encrypts assertions, Name ID and attributes (SAML 2.0 only)		X		
Decrypts assertions, Name ID and attributes (SAML 2.0)	X			
Serves as a credential for client certificate authentication of the artifact back channel				X
Validates other certificates and certificate revocation lists			X	
Use SSL connections to resolve web services variables			X	

More information:

Signing and Verification Operations (see page 627)
Encryption/Decryption Operation (see page 627)
Certificates for SSL Connections (see page 627)

Signing and Verification Operations

The Policy Server uses a private key/certificate pair for signing and verification tasks. The private key/certificate pair signs the assertion, the assertion response, or authentication request. The specific message that is signed depends on the transaction taking place and the federation profile in use.

Before any signing transaction, the partner signing the assertion sends the certificate (public key) associated with the private key/certificate pair to the partner. This exchange is done as part of an out-of-band communication. The partner uses the certificate to verify the signature.

When a transaction occurs, the asserting party includes the certificate in the assertion, by default. During the verification process, however, the partner uses the certificate that it stores at its site to validate the signature.

For SAML 2.0 single logout, the side that initiates the logout signs the request, and the side receiving the request validates the signature. Conversely, the receiving side signs the SLO response and the initiator validates the response.

Encryption/Decryption Operation

For SAML 2.0, you can configure SiteMinder to encrypt an entire assertion, the NameID, or other attributes. If you enable encryption, the asserting party uses the certificate (public key) the relying party sends to encrypt data. Before any transaction, the relying party sends the certificate to the asserting party in an out-of-band exchange. The relying party uses the private key/certificate pair to decrypt the data.

Note: SAML 1.1 does not support encryption of assertion data.

Certificates for SSL Connections

The Policy Server uses SSL connections in the following ways:

- For federated communication.

  You can enable SSL for the SAML HTTP-Artifact back channel or for general federated communication. Establishing the SSL connection requires the relying party to associate the CA certificate with the signed SSL server certificate.

  The SSL server certificate secures the SSL connection. The CA certificate verifies that the SSL server certificate is trusted.

- For ICAS

  You enable an SSL connection to protect the forms credential collector file at the relying party. Import the SSL server certificate from the relying party website into the certificate data store. The server certificate secures the SSL connection.

- For web services variables.

  You can enable SSL connections for resolving web services variables.

Note: SSL server certificates are stored on the web server where they are installed. SSL server certificates are not stored in the certificate data store.

Certificates To Secure the Artifact Back Channel

To implement single sign-on using the artifact binding, the relying party sends a request for an assertion to SiteMinder at the asserting party. The assertion request goes to the Assertion Retrieval Service (SAML 1.1) or the Artifact Resolution Service (SAML 2.0). The retrieval service takes the artifact supplied by the relying party and uses it to retrieve the assertion. SiteMinder sends the response back to the relying party over a back channel. The back channel is a secured connection between the asserting and relying party. In contrast, web browser communication occurs over the front channel.

Secure the back channel and the retrieval service from unauthorized access using one of the following authentication methods:

- Basic
- Basic over SSL
- X.509 Client Certificate

  If you use X.509 client certificate as the authentication method, the relying party must provide a client certificate as its credential. This credential lets the replying party gain access to the service at the asserting party that retrieves the assertion.

Consider the following items when choosing an authentication method:

- Consider using an SSL connection for the back channel. Secure the SSL connection with an SSL server certificate signed by a trusted CA.

  A default set of common root and intermediate CA certificates are shipped with the certificate data store. To use another server certificate signed by a CA, import the CA certificate into the store as a trusted CA certificate.

  Federation uses an SSL-client when processing back channel requests. You can configure the web server at the asserting party to use SSL versions TLSV1_1 and TSLV1_2 with the following ciphers:

  - RSA_With_AES_128_CBC_SHA256
  - RSA_With_AES_256_CBC_SHA256

These ciphers are supported in both FIPS and non-FIPS mode. The determination whether to use SHA256 is made on the SP server side. Federation has no configuration for selecting the algorithm. Administrators must verify that the server at the asserting party is configured appropriately.

- If an X.509 client certificate is required to establish a connection, the relying party must have the key/certificate pair or client certificate authentication fails. Verify that the client certificate exists in the certificate data store at the asserting party. When the relying party sends the request for the assertion, the client certificate serves as the relying party credentials to access the retrieval service.

Check Certificate Validity with CRLs

A Certificate Revocation List (CRL) is issued by a Certificate Authority to its subscribers. The list contains serial numbers of certificates that are invalid or have been revoked. When a request to access a server is received, the server allows or denies access based on the CRL.

SiteMinder federation can leverage CRLs for its certificate functions. For SiteMinder to use a CRL, the certificate data store must point to a current CRL. If SiteMinder tries using a revoked partner certificate, you see an error message. For legacy federation, the error message is in the SAML assertion. The message indicates that authentication failed.

Note: Federation features implement the use of CRLs differently than SiteMinder X.509 authentication schemes. The authentication schemes use an independent LDAP directory that stores CRLs. The authentication schemes do not use the certificate data store.

SiteMinder supports the following CRL features:

- File-based CRLs or LDAP CRLs

  SiteMinder stores CRLs in the certificate data store. File-based CRLs must be in Base64 or binary encoding. LDAP CRLs must be in binary encoding. Additionally, LDAP CRLs must include CRL data in one of the following attributes:

  - certificateRevocationList;binary
  - authorityRevocationList;binary

  When a Certificate Authority publishes an LDAP CRL, it must return the CRL data in binary format, in accordance with RFC4522 and RFC4523. Otherwise, SiteMinder cannot use it.

- PEM and DER encoding formats for a file CRL
- DER encoding format for an LDAP CRL

SiteMinder does not validate an SSL server certificate against a CRL. The web server where the SiteMinder web agent is installed manages the SSL server certificate.

You are not required to have a CRL for each root CA in the system. If there is no CRL for the root CA, SiteMinder assumes that all certificates signed by that CA are trusted certificates.

Add a CRL for Certificate Management

Help ensure that only valid certificates are used for PKI functions by using CRLs. Verify the validity of certificates against a CRL.

Important! SiteMinder explicitly requests LDAP CRLs in binary encoding. Additionally, CRL data must be stored in the LDAP attribute named *certificateRevocationList;binary* or *authorityRevocationList;binary*. When a Certificate Authority (CA) publishes an LDAP CRL, it must return the CRL data in binary format, in accordance with RFC4522 and RFC4523. Otherwise, SiteMinder cannot use it.

For SiteMinder to use a CRL, the CRL location is required.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Infrastructure, X509 Certificate Management, Certificate Validity.

   A revocation list is displayed.
3. Click Add.

   The Configure Revocation List dialog opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Specify an alias for the CRL and the location (URL) of the certificate revocation list.

   The location has to be a file path for a file CRL and an LDAP search path for an LDAP CRL.
5. Click Save.

The CRL is added to the certificate data store.

Update a CRL

Update a CRL to verify that the certificate data in use is current.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Infrastructure, X509 Certificate Management, Certificate Validity.

   A revocation list is displayed.
3. Delete a CRL from the list.
4. Do one of the following steps to add a CRL:
   - Click Add and complete the settings for the new CRL file or LDAP location. Click Save.
   - Navigate back to the Certificate Management dialog. Specify a value for the Default CRL Update Period setting. At the time of the next CRL update, the certificate data store automatically goes to the configured file or LDAP location and reloads the CRL.

Check Certificate Validity with OCSP

SiteMinder provides features that require certificate validation for certificates in the certificate data store. In r12.5, federation features use the certificate data store. These features include protecting the HTTP-Artifact back channel, verifying SAML messages, and encrypting SAML messages.

To check the validity of certificates, the certificate data store can use an OCSP service. OCSP uses an HTTP service that a Certificate Authority (CA) provides to supply certificate validation on demand.

Note: Federation features implement the use of OCSP differently than SiteMinder X.509 authentication schemes. The authentication schemes use an independent LDAP directory to store OCSP responder certificates. The authentication schemes do not use the certificate data store.

By default, SiteMinder does not check the revocation status of a certificate in the certificate data store. To check the revocation status through an OCSP responder, use the OCSP updater utility (OCSPUpdater). When enabled, the OCSPUpdater checks the revocation status for configured OCSP responders every 5 minutes. This default frequency is configurable.

Configuration of the OCSPUpdater relies on the following components:

- SMocsp.conf File

  The OCSPUpdater uses the SMocsp.conf file for OCSP responder configuration. Each Certificate Authority (CA) that issues certificates has its own OCSP responder. In the SMocsp.conf file, include every OCSP responder for each CA certificate in the certificate data store.

  An SMocsp.conf file must exist to use the OCSPUpdater.

  Note: The SMocsp.conf file is the same file that the SiteMinder X.509 certificate authentication scheme uses to configure its own OCSP implementation.

- XPSConfig utility

  XPSConfig lets you customize the behavior of the OCSPUpdater, such as enabling it and setting the frequency of updates. The customization is local to the Policy Server running the OCSPUpdater. Enable an OCSPUpdater on only one Policy Server in a SiteMinder deployment.

Failover Between OCSP and CRL Checking

The certificate data store supports failover from OCSP to CRL validation. If you configure CRLs and OCSP checking, you can enable failover between the two.

SiteMinder federation features do not support certificate distribution point extensions with failover configured, even if the extensions are in a certificate.

More information:

Certificate Validity Checking (optional) (see page 411)

Formats Supported by the Certificate Data Store

The certificate data store supports the following formats:

- Private keys are supported in the following format and must be DER or PEM encoded:
  - PKCS1
  - PKCS5
  - PKCS8
  - PKCS12

  Only RSA keys are supported.

- Public certificates are supported in the following X.509 certificate format:
  - V1
  - V2
  - V3
- Public certificates are supported in the following encoding formats:
  - DER
  - Base64
  - PEM

Import Trusted Certificates and Key/Certificate Pairs

Key/certificate pairs and trusted certificates impact a number of federated SSO and other functions. To execute tasks that use keys and certificates, these items must be in the certificate data store.

If you do not have a key/certificate pair in the certificate data store, you have two options:

- Import a key/certificate pair from an existing file (.p12 or .pfx).
- Generate a key/certificate pair.

  To generate a new key/certificate pair, request a certificate from a trusted Certificate Authority and then import the signed certificate response that the authority returns.

More information:

Certificate and Private Key Usage by SiteMinder (see page 625)
Import a Key/Certificate Pair from an Existing File (see page 633)
How to Generate a Key/Certificate Pair (see page 634)

Import a Key/Certificate Pair from an Existing File

If you do not have a key/certificate pair in the certificate data store, import one from an existing .p12 or .pfx file.

The Policy Server treats an imported certificate as a trusted certificate. The exceptions are self-signed certificates, which get treated according to the following guidelines:

- The Policy Server identifies a V3 self-signed certificate as a CA certificate. In this case, it treats it as a CA certificate. This behavior occurs even though you initiate the import from the Certificate/Private Key dialog.

- The Policy Server treats the certificate as a trusted certificate when:
  - The Policy Server does not identify a V3 self–signed certificate as a CA.
  - The certificate is a V1 self–signed certificate.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Infrastructure, X509 Certificate Management, Trusted Certificates and Private Keys.

   The Certificate and Private Key List displays.
3. Click Import New and follow the wizard.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Be aware of the following items as you complete the wizard:
   - For a trusted certificate file in DER (binary) format, the file can contain one or more certificate entries. For a trusted certificate file in PEM (base 64) format, SiteMinder expects one certificate per file.
   - If you are using a .p12 file, you are required to fill in a password.
   - For each entry you plan to add to the certificate data store, enter the alias you want to associate with that entry. If you select multiple entries, each requires a unique alias.
5. At the Confirm step, review the information and click Finish.

The key/certificate pair is imported into the certificate data store.

How to Generate a Key/Certificate Pair

If you do not have a key/certificate pair in the certificate data store, you can generate a new key/certificate pair.

Perform the following steps:

1. Generate a certificate request and send the request to a trusted Certificate Authority.
2. Import the signed certificate response from the authority.

Generate a Certificate Request

If you do not have a key/certificate pair in the certificate data store, request one from a trusted Certificate Authority. When the CA returns a signed certificate response, import it into the certificate data store.

When you generate a certificate request, the Policy Server generates a private key and a self–signed certificate pair. The Policy Server stores this pair in the certificate data store. Using the generated request, contact a Certificate Authority and fill out the CA certificate request form. Paste the contents of the generated request into the form.

The CA issues a signed certificate response, usually in PKCS #7 format. You can import the signed certificate response into the certificate data store. After the signed certificate response is imported, the existing self–signed certificate entry of the same alias is replaced.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Infrastructure, X509 Certificate Management, Trusted Certificates and Private Keys.

   The Certificate and Private Key List displays.
3. Click Request Certificate.

   The Request Certificate dialog opens.
4. Complete the required fields.

   Note: Click Help for a description of fields, controls, and their respective requirements.
5. Click Save.

A file that conforms to the PKCS #10 specification is generated.

The browser prompts you to save or open the file, which contains the certificate request. If you do not save this file (or open it and extract the text), the Policy Server still generates the private key and self–signed certificate pair. To get a new request file for the private key, generate a new certificate signing request using the Generate CSR feature.

Import a Signed Certificate Response

After completing a certificate request and sending it to the Certificate Authority, the Certificate Authority issues a signed certificate response.

Import the signed certificate into the certificate data store to replace the existing self-signed certificate entry of the same alias.

Follow these steps:

1. Select Infrastructure, X509 Certificate Management, Trusted Certificates and Private Keys.

   The Certificate and Private Key List displays.

2. In the list, locate the self-signed certificate that you want to update.

3. Select Action, Update Certificate next to the self-signed entry.

   The wizard for importing certificates and keys displays.

   Note: Click Help for a description of fields, controls, and their respective requirements.

4. Browse to the file you want. You can use a:

   - .p7 or .p7b file that contains the signed certificate and the corresponding certificate chain.
   - .cer or .crt file (base64 PEM file) with the signed certificate without the certificate chain.

5. Select the appropriate entry.

6. At the Confirm step, review the certificate information and click Finish.

The signed certificate is imported into the certificate data store and the self-signed certificate is replaced.

Generate a New Certificate Signing Request

A certificate signing request (CSR) is a message that you send to a Certificate Authority to apply for a digital identity certificate. After you create a private key, you can generate a CSR. The CSR contains the public key.

You can generate a new CSR for a self-signed or CA-signed private key/certificate pair. The private key always generates an identical CSR without modifying the existing private key. You generate a new request for an existing private key for the following reasons:

- You no longer have the original request that was generated for the private key/self-signed certificate pair.
- You need a new certificate for an expiring one, which requires a new copy of a CSR to submit to a Certificate Authority.

To generate a new CSR:

1. Log in to the Administrative UI.
2. Select Infrastructure, X509 Certificate Management, Trusted Certificates and Private Keys.

   The Certificate and Private Key List displays.
3. Select Action, Generate CSR for the private key entry for which you want a new CSR.

   A file that conforms to the PKCS #10 specification is generated.
4. Save the CSR when prompted.
5. (Optional) If you require a CA-signed certificate, contact a Certificate Authority, and follow the procedure the Certificate Authority requires for submitting a request. Use the PKCS#10 file you saved in the previous step for the request.

After you complete the certificate request process, the Certificate Authority issues a signed certificate response that you import into the certificate data store. The Policy Server replaces the existing certificate entry of the same alias with the newly imported certificate.

Update Certificates in the Certificate Data Store

You can update key/certificate pairs and standalone certificates in the following ways:

- Update an expiring trusted certificate by deleting the existing certificate and importing a new trusted certificate. The new certificate must match the expiring certificate in the certificate data store.
- Update the certificate by importing a signed trusted certificate or a PKCS7-signed response. The new certificate must match the expiring certificate in the certificate data store.
- Update a certificate with a certificate from a PKCS#12 file. The new private key and certificate pair must match the expiring key/certificate pair in the certificate data store.

The new certificate must be valid before the Policy Server can use it to update an expiring certificate. Certificates are updated and become available immediately after they are imported. If the new certificate is not valid, as determined by its validity interval, the Policy Server cannot use the new certificate.

For importing only a trusted certificate, use a file containing the certificate in a PEM or DER encoding. The standard extension for files of these types is *.crt or *.cer. If the file ends in .p12 or .pfx, it is processed as a certificate data store file containing key/certificate pairs. Finally, if a file ends in .p7 or .p7b, it is processed as a signed response file. Anything else is treated as a certificate file, and SiteMinder tries to load a certificate from it.

Note: If you update certificates for a federated environment, you do not have to update any federation objects that use the expiring certificates.

Export Certificate and Key Data

You can export a private key/certificate pair and can send it to your federation partner. The partner can use the certificate to verify the digital signature created with the associated private key of the certificate.

Important! If you export the private key as part of a backup, never share it with anyone else.

Follow these steps:

1. Log in to the Administrative UI.
2. Select Infrastructure, X509 Certificate Management, Trusted Certificates and Private Keys.

   The Certificate and Private Key List displays.
3. Select Action, Export for the entry you want to export.

   The Export Key Store Entry dialog appears.
4. Select the format of the file you want to create from the exported data.

   Note: Click Help for a description of fields, controls, and their respective requirements.
5. Select the file format.
6. Click Export.

   You are prompted to open or save the file on the local system.

SiteMinder generates the encoded file content representing the key or certificate.

Certificate Authority (CA) Certificate Usage

The federation system uses Certificate Authority certificates to verify the following items:

- Whether an SSL server certificate for an SSL connection that secures the SAML HTTP-Artifact back channel is trusted.
- For HTTP-Artifact single sign-on, secure the back channel with an SSL connection. The embedded web server for the federation system can verify that the SSL connection is secured by a trusted certificate by validating the certificate of the Certificate Authority. This certificate must be stored in the certificate data store.
- Whether a certificate revocation list is valid.

  CRLs are acquired from a Certificate Authority. The certificate of the corresponding CA is required to validate the CRL before it can be trusted. The CRL is stored in the data store for use at runtime.

A default set of common root and intermediate CA certificates are shipped with the product for these purposes.

Import a CA Certificate

A set of common root and intermediate CAs are included with the product. To use CA certificates that are not in the certificate data store, import them.

Any certificate that you import is treated as a CA certificate. The exceptions are self-signed certificates:

- If the system identifies a V3 self-signed certificate as a non-CA certificate, the certificate is treated as a trusted certificate. This behavior occurs even though you initiated the import from the Import CA Certificate dialog.
- If the system identifies a V1 self-signed certificate, the certificate is treated as a CA certificate.

To import a CA certificate

1. Log in to the Administrative UI.
2. Select Infrastructure, X509 Certificate Management, Certificate Authorities.

   The Certificate Authorities List displays.
3. Click Import New.

   The Import CA Certificate dialog displays.

   Note: Click Help for a description of fields, controls, and their respective requirements.

4. Follow the wizard to import a new entry.
5. At the Confirm step, review the certificate information and click Finish.

The CA certificate is imported into the certificate data store. The change takes place directly after the import is complete.

Important! You cannot delete a CA certificate that is part of a trust chain for other certificates in use on the system. If you try to delete a CA certificate in use, an error message states that the certificate cannot be deleted.

Chapter 23: Global Policies, Rules, and Responses

This section contains the following topics:

Global Policies

Standard SiteMinder 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 561)
Authentication Events (see page 517)
Authorization Events (see page 519)

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 (nonglobal) counterparts.

Global response compared to 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.
- Cannot be added to response groups. There are no global response groups.

Similarities:

- Can use active expressions.
- Is only returned if it is specified in a particular policy.

Global rule compared to 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 a specific agent or agent group. The agent is explicitly specified when the rule is created.
- Available only for SiteMinder 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 compared to 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 is 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 processing, the responses that are defined for the fired global rules are added to the list of other responses. A global rule fires when the following conditions are true:

- The resource being accessed matches the absolute resource filter that is 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.

Important! The standard policy takes precedence over the global policy if Global policies processing is enabled for the domain and both standard rule and global rule are bound to the same agent or agent group.

More information:

Disable Global Policy Processing for a Domain (see page 500)

SiteMinder Global Policy Concept

SiteMinder uses a policy-based access control model. A SiteMinder policy defines the type of access a user has to a particular resource and what happens when the user accesses the resource. Each standard SiteMinder 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*. For information about realms, rules, and responses in standard SiteMinder policies, see the following:

- Configure a Realm (see page 507)
- Rule Groups (see page 531)
- Rules (see page 513)
- Responses and Response Groups (see page 535)
- Policies (see page 561)

SiteMinder objects can be of two types: system level and domain level. In a standard (non-global) SiteMinder 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 SiteMinder 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:

Policies Explanation (see page 563)

Global Policy Processing

Policies are evaluated as described in Policy Processing (see page 597). 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 or Create a Global Rule for Authorization Events
2. Configure a Global Response

3. Configure a Global Web Agent Response Attribute (see page 654)
4. Configure the Global Policy (see page 655)

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 SiteMinder 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 SiteMinder 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 652)

Create a Global Rule for Authentication Events

You can 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 Rules.

   The Global Rules page appears.
3. Click Create Global Rule.

   The Create Global Rule page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Enter the global rule name.
5. Specify agent and resource settings in Realm and Resource.

   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.
6. Select Authentication events.
7. Select an OnAuth event from the Action List.
8. Click Submit.

   The global rule is saved.

Global Rules for Authorization Events

Global rules that include SiteMinder authorization events allow SiteMinder 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, SiteMinder 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 authorization 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 Rules.

   The Global Rules page appears.
3. Click Create Global Rule.

   The Create Global Rule page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Enter the global rule name.
5. Specify agent and resource settings in Realm and Resource.

   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.
6. Select Authorization events.
7. Select an OnAccess event from the Action List.
8. Click Submit.

   The global rule is saved.

More information:

Responses and Response Groups (see page 535)
Resource Matching and Regular Expressions (see page 525)

Enable and Disable Global Rules

You enable a global rule to ensure SiteMinder fires the rule if a user accesses the specified resource and triggers the authentication or authorization event. You disable a global rule to prevent SiteMinder 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.

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.

More information:

Add Time Restrictions to Rules (see page 528)

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 in the Software Development Kit.

Note: For more information about shared libraries, see the *Programming Guide for C*.

To configure an Active Rule

1. Specify the library name, function name, and function parameters in the fields on the Active Rule group box.

   The active rule string is displayed in the Active Rule field.

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

2. 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 about modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 54).

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 can 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 Responses.

   The Global Responses page appears.
3. Click Create Global Response.

   The Create Global Response page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Enter the global response name.
5. Select a SiteMinder Agent type from the SiteMinder Agent type list.
6. Click Submit.

   The global response is saved. You can now add response attributes to the response. For more information about adding response attributes for each Agent type, see Response Attributes for Global Responses (see page 653).

Response Attributes for Global Responses

Each SiteMinder global response may contain one or more response attributes. Response attributes identify the pieces of information that the Policy Server passes to a SiteMinder Agent. Each SiteMinder Agent type can accept different response attributes.

Global Response Attribute Types

SiteMinder 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 SiteMinder Agent. Web Agent response attributes support HTTP header variables, cookie variables, redirections to other resources, text, and timeout values. More information on Web Agent response attribute types exists in the *Web Agent Configuration Guide*.

Note: If you have purchased CA SOA Security Manager, you can find information about the WebAgent-SAML-Session-Ticket-Variable response attribute type in the CA SOA Security Manager Policy Configuration Guide.

To create a response attribute

1. Click Create Response Attribute.

   The Create Response Attribute page appears.

2. Select a response attribute.

3. Select an attribute type.

   The details in the Attribute Fields are updated to match the specified attribute type.

4. Complete the details in the Attribute Fields.

   Note: A list of automatically generated SiteMinder user attributes that you can use in responses exists in SiteMinder Generated User Attributes (see page 550).

5. (Optional) Edit the attribute in the Script field.

   Note: The Attribute Setup section closes when you edit the attribute on the Advanced section.

6. Specify Cache Value or Recalculate value every ... seconds.

   Note: The maximum time limit that can be entered is 3600 seconds.

7. Click Submit.

   The Create Response Attribute Task is submitted for processing, and the response attribute is added to the Attribute List on the Response page.

More information:

Global Response Attribute Types (see page 653)

How to Configure Global Policy Objects

Configuring a global policy requires you to complete the following procedures:

1. Create the Global Policy
2. Add Global Rules to the Global Policy (see page 655)
3. (Optional) Associate a Global Rule with a Response (see page 656)

Create the Global Policy

You can create a global policy to define how users interact with resources.

To create a global policy

1. Click Policies, Global.
2. Click Global Policies.

   The Global Policies page appears.
3. Click Create Global Policy.

   The Create Global Policy page appears.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Enter the global policy name.
5. Add global rules and global responses.
6. Click Submit.

   The global policy is created.

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.

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 584)

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 580)

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:

How the Web Agent and Policy Server Calculate Time (see page 61)
Time Restrictions for Policies (see page 583)

Chapter 24: Impersonation

This section contains the following topics:

Impersonation Overview

Impersonation provides a method for a privileged user to assume the role of another user without ending the privileged user's session. This feature facilitates the following:

- Customer service representatives (CSRs) impersonate customers to investigate access problems.
- Helpdesk representatives impersonate employees to investigate access problems.
- Employees impersonate co-workers who are on vacation or out of the office.
- Any other situation in which one user must temporarily assume the identity, of another user.

Impersonation provides a secure solution in the preceding situations. In all cases, passwords are not disclosed to allow one user to impersonate another user.

The following terms are used when describing impersonation:

Impersonated session

A user session created for the purpose of assuming another user's identity.

Impersonatee

The user whose identity is assumed by a privileged user.

Impersonator

The privileged user that is able to assume the identities of other users.

Impersonation authentication scheme

A method for authenticating a user that allows a privileged user to assume the identity of another user while preserving the identity of the impersonator.

Session

Also known as user session. This term is the time between authenticating and logging out.

Session Specification

Also known as the Session Ticket or Session Spec, this term is the information held in a proprietary format on the Policy Server that describes a user and the characteristics of the current session.

SMSESSION

The name of the Web Agent cookie that contains the Policy Server's Session Specification.

Note: For information about sessions, see the *Policy Server Administration Guide*.

Impersonation Process

The process of impersonating another user consists of the following:

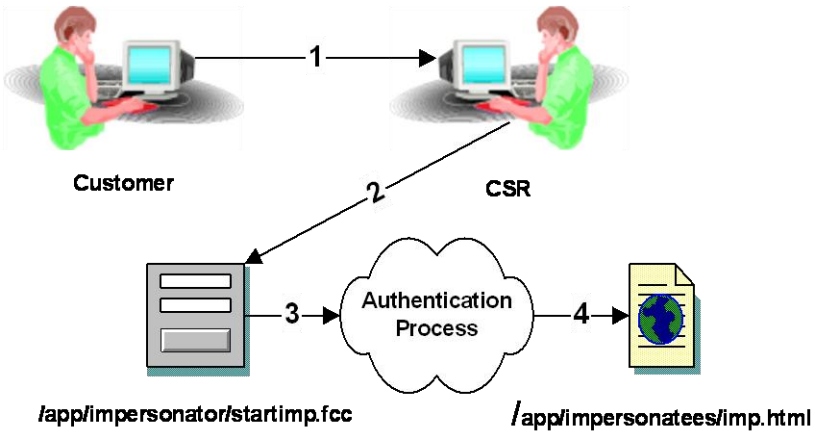
- A privileged user (impersonator) establishes a SiteMinder session by accessing a resource protected by SiteMinder. This can be the Administrative UI or some other application or resource protected by a SiteMinder policy.
- The impersonator, using the currently established identity, accesses a form which allows the impersonator to specify the user to be impersonated (impersonatee). The impersonator does not present the credentials of the impersonatee. The form is an .fcc file with special logic to enable the impersonated session.
- The impersonator submits information about the impersonatee.

The Policy Server determines whether or not the impersonator may be impersonated using a series of policies.

- Determines whether or not the impersonator has sufficient rights to impersonate the impersonatee.
- Uses the impersonator's established session when granting the impersonator access as the impersonatee.
- Audits all impersonator activity.

To begin an impersonated session, an impersonator accesses an .fcc file directly that has a target that points to a resource in the realm protected by the impersonation authentication scheme.

The following diagram shows an example of impersonation where a customer service representative (CSR) accesses an .fcc file directly to begin an impersonation session.



1. A customer with a problem calls his CSR. The CSR determines that the best way to solve the customer's problem is to impersonate the customer.
2. The CSR accesses the impersonation start page which in this case is named "/app/impersonators/startimp.fcc" to begin the impersonation process. The Forms Credential Collector (FCC) presents a form to the CSR. The CSR provides the name of the customer to be impersonated, and possibly other attributes for disambiguation. Within the .fcc file are directives (not shown) that collect the session specification of the CSR for use by the authentication scheme. The impersonation authentication scheme is invoked on the Policy Server because the target of the FCC resides in a SiteMinder realm that is protected by the impersonation authentication scheme.
3. The Policy Server takes the information gathered by the FCC and uses it to determine if the customer to be impersonated can be found in any of the configured user directories. If found, the Policy Server determines if the CSR may impersonate the customer, and if the customer may be impersonated. If both are allowed, the Policy Server authenticates the impersonator and the CSR acts as the customer for the duration of the impersonated session.
4. Finally, the Policy Server directs the CSR (impersonating the Customer) to the target of the FCC.

Security Considerations for Impersonation

While impersonating a customer, an impersonator's session specification will look to SiteMinder much like the session specification of any customer. The major difference is that the impersonator's distinguished name and the user directory in which the impersonator originally authenticated will be present as additional fields. This allows all impersonated access to resources to be passed through additional checks. It also allows the Policy Server to record impersonated activities for auditing.

The impersonated session specification is also used to prevent impersonation chaining. When the Policy Server determines that the fields for the impersonator DN and user directory are in use, it will not allow further impersonation and will reject the login attempt. This stops impersonators from stacking one impersonation on top of another to gain access to otherwise restricted resources.

Effects of Authentication Scheme Protection Levels

While impersonating a user, the protection level at which an impersonator originally authenticated will not be checked. Normally, when accessing resources in a new realm protected by an authentication scheme at a higher level, the user would be challenged for new credentials. However, since an impersonator should be a privileged user, these types of challenges will not occur during an impersonation session. Protection levels are meant to indicate the strength of credentials used to access resources in a realm. In the case of impersonation, there are no credentials specific to the user being impersonated, therefore protection levels are not considered.

Note: Once the impersonated session ends, protection levels are once again enforced normally.

Session Idle Timeouts

During an impersonated session, an impersonator's original session can possibly idle out. This depends on the idle timeout value for the realm in which the impersonator originally authenticated. If the impersonator impersonates a user for a longer period than their original idle timeout, the impersonation session ends with the impersonator's original session. To avoid this situation, increase the session idle timeout for realms in which impersonators commonly authenticate.

Chapter 25: Password Policies

This section contains the following topics:

Password Services Overview

Password Services provide an additional layer of security to protected resources. Password Services uses a forms credential collector (FCC) and customizable HTML forms to manage user passwords in:

- A user directory
- Part of a user directory

  Note: For more information about the stores Password Services supports, see the r12.5 SiteMinder Platform Support Matrix. For more information about agents and Password Services, see the *Web Agent Configuration Guide*.

You use password policies to manage user passwords and to, optionally, enable user–initiated password changes.

- Password policies define rules and restrictions that govern the following:
  - password expiration
  - composition
  - usage

  Note: You can apply multiple password policies to all or part of a user directory. SiteMinder applies the policies according to the priorities you specify for each.

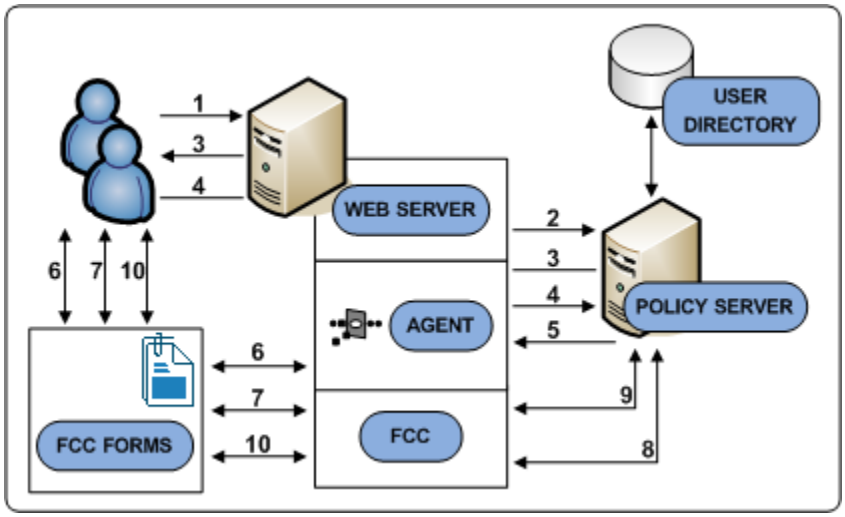
- SiteMinder invokes a password policy when a user attempts to access a protected resource and evaluates the credentials of the user. If the policy determines that the password is expired, SiteMinder can:
  - Disable the user account to prevent unauthorized access. If disabled, a SiteMinder administrator must re–activate the account.
  - Force the user to change the password.

You can use password policies to enforce password composition rules without administrative intervention. Password Services lets you:

- Force users to change a password at time you specify.
- Enable user–initiated password changes.

How Password Services Work

The following illustrates how Password Services works, including what happens when user password is expired and must be changed:



1. The user requests a resource.
2. The agent intercepts the request and checks with the Policy Server to see if the resource is protected.
3. If the resource is protected, the Policy Server requests user credentials from the browser.
4. The user sends the credentials to the Policy Server.

5. The Policy Server:
   a. Authenticates the user.
   b. Determines if the user is stored in a user directory that is bound to a password policy.
   c. Uses the password policy criteria to verify that the password is valid.
6. If the password has expired:
   a. The Policy Server sends a redirection URL to the agent. The URL points to the FCC.
   b. The agent uses the URL to redirect the browser to request the FCC.
7. The FCC determines which form to present to the user from a set of HTML templates. The FCC displays the form in the browser. A message provides a reason for the redirection and prompts the user to change the password.
8. The FCC passes the password to the Policy Server.

   The Policy Server:
   a. Uses the password policy to verify that the new password meets the composition criteria.
   b. Changes the password, if the password is valid.
9. The Policy Server sends a request to the agent to redirect the browser to the FCC.
10. The FCC displays a message that states that the password change has succeeded.

    After the user reads the message, the user is redirected to the requested resource.

Password Policy Considerations

If you plan to implement password policies in your enterprise, consider the following:

- SiteMinder requires read/write access to the user directory, including exclusive use of several attributes within that directory to store passwords and password–related information.
- Password policies can affect SiteMinder performance because of the additional user directory searches required to validate passwords. Password policies that are configured to search only part of a user directory, instead of the entire directory, can also affect performance.
- If your user directory has a native password policy, this policy must be:
  - Less–restrictive then the SiteMinder password policy or
  - Disabled

  Otherwise the native password policy accepts or reject passwords without notifying SiteMinder. Consequently, SiteMinder cannot manage those passwords.

- By default, if a user enters incorrect information when changing a password, SiteMinder returns a generic failure message. This message does not specify the failure reason. Create and enable the DisallowForceLogin registry key to change the default behavior and explicitly tell users why the change failed.
- If you use password policies on multiple Policy Servers, synchronize the system times of all servers. Synchronizing times helps to avoid the disabling of accounts or forcing password changes prematurely.

  Note: For more information, see the *SiteMinder Policy Server Configuration Guide*.

Create Password Policies

You can create a password policy to provide an additional layer of security to protected resources.

Note: The Password Services FCC is the default redirection URL. For more information about agents and the FCC, see the *Web Agent Configuration Guide*.

Follow these steps:

1. Click Policies, Password.
2. Click Password Policies.
3. Click Create Password Policy.

   Password policy settings appear.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Enter a policy name.
5. Select the user directory to which the policy applies from the Directory list.
6. Specify if the policy applies to the entire directory or part of the directory.
7. (Optional) If the policy only applies to part of the directory, click Lookup to specify which part.
8. Configure the policy to reflect the password logic you want by configuring expiration, composition, expression, restriction, or advanced settings.

Configure Password Expiration

You configure password expiration settings to define events, that when triggered, the Policy Server disables the user account and optionally redirects the user to a new Web page. Examples of such events include multiple failed login attempts and account inactivity.

Note: Expiration settings are optional. If you do not want to enable an expiration setting, leave the respective fields blank.

To configure password expiration

1. Click the Expiration tab.

   Password expiration settings open.

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

2. Specify user login tracking settings by selecting the Track successful logins, Track failed logins, and Authenticate on Login Tracking Failure check boxes in the Expiration group box.

   Note: You must select the Track successful logins check box if you want to disable accounts based on account inactivity. You must select the Track failed logins check box if you want to disable accounts based on failed login attempts.

3. Specify the settings that determine how often a password must be changed in the Password expires if not changed group box.

4. Specify the settings that determine how many incorrect password attempts are permitted in the Incorrect Password group box.

5. Specify the settings that determine how long a password can remain inactive in Password expires from inactivity group box.

   Note: If you do not need to configure passwords to expire from inactivity, we recommended that you do not set this option for performance reasons.

6. Click Submit to save the password policy or click another tab to continue working with the password policy.

Configure Password Composition

You configure password composition rules to control the character composition of newly created passwords.

Note: Composition rules are optional. If you do not want to enable a composition rule, leave the respective fields blank.

To configure password composition restrictions

1. Click the Composition tab.

   Password composition settings open.

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

2. Enter the minimum and maximum character length for passwords in the Minimum Length and Maximum Length fields.

3. Enter the maximum number of characters that can appear consecutively in a password in the Maximum field.

4. Specify the permissible characters types and the minimum requirements for each in the Content Minimum group box.

   Note: If you are using Netscape 4.1 Directory Server with Password Services, do not specify a non-printable characters minimum. Netscape 4.1 Directory Server does not accept non-printable characters.

5. Click Submit to save the password policy or click another tab to continue working with the password policy.

Password Regular Expressions

Regular expression matching for passwords allows you to specify text patterns used for string matching that each password must match or not match to be considered valid.

For example, if you require the first character in the password be a digit but not be the last character, you can configure a regular expression to enforce this requirement and all passwords will be checked against it.

Regular Expressions Syntax

The following table describes the characters you can use for constructing regular expressions for password matching. This syntax is consistent with the regular expression syntax supported for resource matching when specifying realms.

All closure operators (+, *, ?) are greedy by default, meaning that they match as many elements of the string as possible without causing the overall match to fail. If you want a closure to be reluctant (non-greedy), follow it with a '?'. A reluctant closure matches as few elements of the string as possible when finding matches.

The regular expression syntax is a s follows:

Characters	Results
\	Used to quote a meta-character (like '*')
\\	Matches a single '\' character
(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*?	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.

Configure Regular Expression Matching

You configure regular expressions to specify text patterns that are used for string matching. A password must match or not match the expression to be valid. Each regular expression entry is a name/value pair consisting of a descriptive tag and expression definition.

Regular expression matching for passwords is optional. If you decide to use regular expression, you only specify entries for expressions that passwords must match or must not match. If you have no expression matching requirements, do not create any regular expression entries.

To configure regular expressions for passwords

1. In the Password Policy dialog, select the Regular Expressions tab.

   You will see an empty table in the Regular Expressions group box.

2. Click Add to add an expression.

   The Password Regular Expression dialog opens.

3. Select one of the following radio buttons:

   - MUST Match

     If you select this option, define a regular expression that passwords must match.

   - MUST NOT Match

     If you select this option, add an entry for each regular expression that passwords must not match.

4. Enter values for the fields.

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

5. Click OK.

   The regular expression is added to the table. If you selected MUST NOT match, you will see a checkbox in the NO Match column.

Configure Password Restrictions

You configure password restrictions to place restrictions on password usage. Restrictions include:

- how long a user must wait before reusing a password
- how different the password must be from ones previously used

You can also prevent users from specifying words that you determine are a security risk or contain users' personal information.

Note: Restrictions are optional. If you do not want to enable a restriction, leave the respective fields blank.

To configure password restrictions

1. Click the Restrictions tab.

   Password restriction settings open.

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

2. Specify how much time must pass and/or how many new passwords must be created before an old password can be reused in the Reuse group box.

   Note: If you specify both criteria, each must be satisfied before a user can reuse a password.

   Example: A password policy requires users to wait 365 days and specify 12 passwords before reusing a password. After a year, if a user only supplied six passwords, the user would have to supply another six passwords before reusing the first password.

3. Specify how much a new password must differ from the previous password in the Changed Required group box.

4. Specify the number of consecutive characters the password policy compares to personal information stored in user profiles in the Profile Attributes group box.

5. Specify the path to a user-defined dictionary of forbidden passwords and the length of the string compared against values in the dictionary in the Dictionary group box.

6. Click Apply to save the changes or click OK to save the changes and return to the Administrative UI.

Configure Advanced Password Options

You configure advanced password policy options to specify that submitted passwords be pre-processed before validation and storage. Advanced password policies let you assign a priority to a policy, which allows the predictable evaluation of multiple password policies that apply to the same user directory or namespace.

Note: Pre-processing options are optional. You should specify a unique password policy evaluation priority for each password policy that may be assigned to the user directory or namespace.

To configure advanced password options

1. Click the Advanced tab.

   Advanced password policy settings open.

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

2. Specify options to process submitted passwords prior to evaluation and storage in the Password Pre-Processing group box.

   Note: You should specify identical pre-processing options for each password policy that is applied to the same user directory or namespace.

3. (Optional) If the password policy is one of multiple policies that applies to the same user directory or namespace, specify a the password policy priority in the Password Policy Priority group box.

   Note: Evaluation priorities range from 0-999, where 999 is the highest.

User-initiated Password Changes

User-initiated password changes allow end users to change their passwords without any intervention from an administrator. Users can elect to change their passwords by clicking a link to access the Password Change Request form.

Add a Change Password Link

To enable user-initiated password changes, the Policy Server administrator must add a Change Password link to an HTML page. For example, administrators might add this link to a login page so users can opt to change their password at login.

Note: For more information, see the *Web Agent Configuration Guide*.

Password Self-Changes

When users want to change their passwords they must:

1. Click Change Password.

   The Administrative UI displays the Password Change Request form.

2. Enter the requested information, then click the Change Password button.

   The Administrative UI displays another Password Change Information page, indicating that the user's password has been changed.

Remove the Login ID When Redirecting for Password Services

During password services processing, a user request is redirected multiple times. When the request is redirected, the login ID (typically the username) which was entered by the user is appended to the request URL by default. To modify the default behavior so that the login ID (username) is not appended to redirects, you can do one of the following procedures.

To remove the login ID when redirecting for password services in Windows

1. Add the following registry key:
   `HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\PolicyServer\DisallowUsernameInURL`
2. Set the DWORD value to one of the following values:
   - 0 — Applies the default behavior of appending the UID to the request URL.
   - 1 — Changes the default behavior so that the UID is not appended to the request URL.

To remove the login ID when redirecting for password services in UNIX

1. Navigate to:
   `<policy-server-install-dir>/registry/`
2. In a text editor, open the following file:
   `sm.registry`
3. Add the following registry key:
   `HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\PolicyServer\DisallowUsernameInURL`
4. Set the DWORD value to one of the following values:
   - 0 — Applies the default behavior of appending the UID to the request URL.
   - 1 — Changes the default behavior so that the UID is not appended to the request URL.

Enable Password Change Failure Messages

By default, if a user enters incorrect information when changing a password, SiteMinder returns a generic failure message. This message does not specify the failure reason.

You can change the default behavior and explicitly tell users why the change failed.

Follow these steps:

1. Access the Policy Server host system and do one of the following:
   a. (Windows) Open the Registry Editor and navigate to HKEY_LOCAL_MACHINE\Software\Netegrity\SiteMinder\CurrentVersion\Policy Server.
   b. (UNIX) Open the sm.registry file. The default location of this file is siteminder_home/registry.

      siteminder_home

      Specifies the Policy Server installation path.
2. Create DisallowForceLogin using the following settings:

   KeyType: REG_DWORD

   Value: 0 or 1

   0

   (default) SiteMinder returns a generic failure message. This behavior is consistent with the default SiteMinder behavior.

   1

   SiteMinder Returns an explicit failure reason.

   Note: A value other than 1 or 0 is not supported.
3. Do one of the following:
   - (Windows) Exit the Registry Editor.
   - (UNIX) Save the registry file.
4. Restart the Policy Server.

Password Policy Troubleshooting

The following sections describe and provide solutions to problems that may occur when implementing password policies.

New User Passwords are Rejected

Symptom:

User-specified passwords are always rejected.

Solution:

The password policy may be too strict or improperly configured. Check the content minimums and the password length composition settings for consistency.

User Accounts are Mistakenly Disabled

Symptom:

Users accounts that have not exceeded the number of permitted failed login attempts are becoming disabled.

Solution:

Check the incorrect password settings. The setting for disabling an account after a specific number of consecutive incorrect password attempts may be too low.

Setting this value too low causes a problem when two or more users, which are located in different user directories, have the same user name. When the Policy Server attempts to authorize a user, it checks all user names that correspond to the login and then attempts to match the password. If the Policy Server finds a user name that the password does not match, it records a failed attempt for that user. If this happens more than the number of times specified by the in the incorrect password settings, the account is disabled.

User Accounts are Prematurely Disabled

Symptom:

User accounts are prematurely disabled in a multi-Policy Server environment.

Solution

Check that there is no time differential between the Policy servers.

Password Changes are Forced

Symptom:

User accounts are forced to changed passwords too soon in a multi-Policy Server environment.

Solution:

Check that there is no time differential between the Policy servers.

LDAP Users Do Not Disable

Symptom:

Password policies do not disable LDAP users.

Solution:

Check the following:

- The LDAP directory to which the policy is bound is writable. You must be able to save passwords in each user's record in the LDAP directory.
- The user directory setting is valid for password attribute, password data, and disable flag user profile attributes. More information on configuring the password attribute, password data, and disabled flag user profile attributes exists in Directory Attributes Overview (see page 170).

Active Directory Users Cannot Change Passwords

Symptom:

Users stored in Active Directory user directories cannot change their passwords.

Solution:

Check the following:

- The Active Directory user directory to which the policy is bound is configured with a secure (SSL) connection.
- The Active Directory user directory to which the policy is bound is configured to use the unicodePWD Password Attribute.

Incorrect Password Message Does Not Appear

Symptom:

When a user submits a password change request that contains an invalid current password, the Password Change Information screen does not open with a message stating that the current password is incorrect. Rather, the Policy Server redirects the user to:

- The login screen without the message if an On-Auth-Reject-Redirect response is not bound to the policy configured with the user directory
- The URL associated with the On-Auth-Reject-Redirect response bound to the policy configured with the user directory

Solution:

Enable the DisallowForceLogin registry key, which is located at HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\PolicyServer.

DisallowForceLogin

Redirects users to the Password Change Information screen to re-enter the current password when the change request contains an invalid current password.

KeyType: REG_DWORD

Value: 0 (disabled) or 1 (enabled)

Default: 0 (disabled)

Note: If the registry key is enabled, values other than 0 or 1 are unsupported and have undefined behavior.

Chapter 26: SiteMinder Test Tool

This section contains the following topics:

Test Tool Overview

The SiteMinder Test Tool is a utility that simulates the interaction between Agents and Policy Servers. It tests the functionality of the Policy Server. During testing, the Test Tool acts as the Agent, making the same requests to the Policy Server as a real Agent. This allows you to test your SiteMinder configuration before deploying it.

The SiteMinder Test Tool is only available on Windows systems. It can be used to emulate Windows-based Web Agents and RADIUS Agents without limitations. The SiteMinder Test Tool can create connections with any Policy Server on all platforms, however, post 4.x Web Agents on Solaris cannot be tested with the SiteMinder Test Tool. To test policies for SiteMinder 5.x and later Web Agents, do one of the following:

- Use the Perl scripting interface for Web Agent registration and testing.
- Install and configure the SiteMinder Web Agent and test it manually.

The SiteMinder Test Tool performs three types of tests:

Functionality

Tests policies to ensure they are configured correctly.

Regression

Tests whether or not changes, such as migrating a policy store or implementing a new feature, affect SiteMinder.

Stress

Tests the performance of the Policy Server as it receives multiple requests.

Note: You can also test policies using the Scripting Interface. See the Programming Guide for Perl.

Configure Your Test Environment Agent

You can configure the Agent that the Test Tool simulates during a test in the SiteMinder Agent group box.

The Agent that the Test Tool simulates must be configured in the Administrative UI.

To configure Agent information, specify the following options

Agent Type

Specify one of the following Agent Types:

Version 4

Simulates SiteMinder 4.x Agents.

Version 5

Simulates SiteMinder 5.x Agents.

Note: If you want to use the Test Tool on a system to simulate a SiteMinder 5.x Web Agent, you must run the smreghost.exe application on the system where you will run the Test Tool. The smreghost.exe file is included with your Web Agent, and described in the Web Agent Installation Guide. The file is also located in <Policy Server install dir>/siteminder/bin.

RADIUS

Simulates RADIUS devices.

Agent Name

Enter the name of the Agent as it appears in the Administrative UI. This field is required for both Version 4 and Version 5 Agent simulations.

Secret

Enter the Agent's shared secret. This must match the shared secret entered when the Agent was created. A Secret is required for Version 4 and RADIUS Agent simulations.

(Optional) Server

Enter the full name of the server on which the Agent resides. For example, to test the Policy Server for http://www.myorg.org, enter www.myorg.org in this field. This field may be used for Version 4 Agent simulations.

SmHost.conf Path

Enter the path to the SmHost.conf file that contains the settings for the Version 5 Agent you want to simulate. You can use the Browse button to search for the SmHost.conf file.

Policy Server Identification

The test tool requires information about the Policy Server that will be used when simulating the interaction with the Agent described in the SiteMinder Agent group box. The required information differs slightly depending on the type of Agent you selected.

Set Up the Policy Server for Version 4 Agents and RADIUS Agent Simulations

For Version 4 Agents and RADIUS Agent simulations, you must specify the IP address and port information of the Policy Server(s) used in the test. If you want to simulate a multiple Policy Server environment, you can specify how those Policy Servers operate.

To set up Policy Server(s) for Version 4 Agent and RADIUS Agent simulations

1. Specify the following Policy Server options, as necessary:

   Policy Server

   Indicates whether you are specifying the primary or secondary Policy Server.

   IP Address

   Specifies the IP address of the Policy Server. By default, this field contains the IP address of the local system.

   Authorization, Authentication, and Accounting Ports

   Specifies the TCP ports used for authorization, authentication, and accounting requests. These fields are populated with the Policy Server's default port numbers.

   Timeout

   Displays the time (in seconds) that the Test Tool should wait for a response from the Policy Server.

2. Select one of the following operation modes:

   Failover

   Enables failover. During failover, the Test Tool directs requests to the initial Policy Server. If the initial Policy Server fails, the Test Tool redirects requests to the secondary Policy Server.

   Round Robin

   Enables round robin load balancing. Round robin load balancing divides requests between the primary and secondary Policy Servers. For each connection, the Test Tool alternates between Policy Servers.

3. Click Connect to make sure that the Test Tool can connect to the Policy Server.

   If the Test Tool makes a connection, the IsProtected and DoManagement stop lights turn green.

Note: You must specify an Agent before testing the Policy Server connection.

Policy Server Information for Version 5 Agents

For Version 5 Agents simulations, you may specify the IP address and port information of the Policy Server(s) used in the test, or you may use the Policy Server information contained in the Host Configuration Object contained in the policy store.

By default, the Policy Server information will be retrieved from the policy store when the Test Tool uses the SmHost.conf file to establish an initial connection to the Policy Server. To specify Policy Server information manually, select the Override check box and fill in Policy Server information as described in Set Up the Policy Server for Version 4 Agents and RADIUS Agent Simulations (see page 685).

You can configure the Agent that the Test Tool simulates during a test in the SiteMinder Agent group box.

Select a Test Mode

Use one of the test modes in the following list to determine how tests are run and results are displayed. Depending on the test mode that you select, you may also have to specify script information.

Interactive

Allows you to enter data, run tests, and see the results displayed immediately in the Server Response section.

Record

Combines Interactive operation with a script generation feature that writes test results to a plain-text command script file.

Basic Playback

Uses Command Script files created in the Record mode to automate sequential tests. Ideal for regression testing.

Advanced Playback

Uses a manually configured Thread Control File to automate complex tests. Ideal for stress testing.

More information:

Perform a Regression Test (see page 693)
Run a Stress Test (see page 694)

Specify Resource Information

You can specify the resource against which you want to conduct tests. Providing a resource simulates a user entering a URL in a browser.

To specify resource information, provide values for the following options

Resource

Enter the relative path of the resource that SiteMinder is protecting as it is configured in the realm. The path is relative to the Web server's publishing directory. For example, /protected/.

Action

Enter the Agent action, Authentication event, or Authorization event specified in the rule that you are testing.

You can configure the Agent that the Test Tool simulates during a test in the SiteMinder Agent group box.

Specify User Credentials

The Test Tool requires user credentials to test whether or not a policy can authenticate or authorize a user.

To specify user credentials, complete the following fields:

User Name

Enter the user name you want to use to access the resource.

Password

Enter the password for the user entered in User Name.

CHAP Password

If you are using a RADIUS CHAP authentication scheme, select this check box.

Certificate File

If the protected resource requires certificates to authenticate users, you must provide a certificate file so that the Test Tool can simulate certificate authentication.

You can configure the Agent that the Test Tool simulates during a test in the SiteMinder Agent group box.

Set the Encoding Spec

The encoding spec field allows you to specify a language encoding parameter. The Test Tool uses this parameter to encode headers in the same manner as a Web Agent. It then displays the encoded response attribute data in the Attributes field.

For more information about language encoding, see the Web Agent Configuration Guide.

To set the encoding spec, enter a value for the encoding spec as follows:

encoding_spec, wrapping_spec

where:

- encoding_spec is a text string that represents one of the following encoding types: UTF-8, Shift-JIS, EUC-J, or ISO-2022 JP
- wrapping_spec is the wrapping specification, which must be RFC-2047.

Note: If you leave this field blank, the default is UTF-8 with no wrapping.

You can configure the Agent that the Test Tool simulates during a test in the SiteMinder Agent group box.

Save and Load Test Configurations in a Test Tool Settings File

To avoid reentering user-supplied information, such as Agent, resource, and user information, you can save these values into a Test Tool Settings file. You can then reload those values at any time.

To save the current values that are specified in the Test Tool:

1. Click the Save Settings button.
2. Enter a location and name for the Test Tool Settings in the Save As dialog and click Save.

   The file is saved with a .ini file extension.

To retrieve the saved values from the Test Tool Settings file:

1. Click the Load Settings button.
2. Enter the location and name of the Test Tool Settings File in the Open dialog and click Open.

Note: You can also load the Test Tool Settings file from a Command Script.

Run a Functionality Test

The SiteMinder Test Tool allows you to test the functionality of policies in a simulated real-world environment. To perform a functionality test, you must have the following:

- A Policy Server that is configured and running
- A SiteMinder Agent that is configured in the Administrative UI

  Note: If the Test Tool is simulating a SiteMinder Agent v5.x, that Agent must have 4.x support enabled.
- A policy domain configured with any type of user directory
- A policy that pairs a rule with a user(s)

SiteMinder allows you to perform the following functionality tests:

IsProtected

Indicates whether or not a policy is protecting the resource you specified.

IsAuthenticated

Indicates whether or not the Policy Server can authenticate a set of user credentials against a user directory.

When user credentials are authenticated, the Policy Server compares the credentials to entries in a user directory. If the credentials match an entry, the Policy Server creates a session ticket and authenticates the user.

In a "real" SiteMinder deployment, SiteMinder confirms that a user's session ticket is valid instead of rechecking the user's credentials against a directory when an authenticated user makes additional requests. By default, the Test Tool authenticates the user each time the IsAuthenticated test is run, regardless of whether or not the user has a session ticket.

You can configure the Test Tool to validate a user's session ticket by entering Validate in the Comment field in the Test Tool before running an IsAuthenticated test; however, SiteMinder must authenticate the user before validating the session ticket.

Note: You can specify Validate when you run multiple tests in Interactive mode (using the Repeat count field), and in Playback mode.

IsAuthorized

Indicates whether or not the Policy Server can authorize a user based on a policy.

These tests must be run in the order they appear above. For example, you must run IsProtected before running IsAuthenticated. The order reflects the steps that SiteMinder uses to determine a user's access rights.

While running functionality tests, you can also use the Test Tool to perform the following tasks:

DoAccounting

Logs the most recent accounting server transactions.

DoManagement

Requests Agent commands, such as cache flush commands that clear the Agent cache. Running DoManagement ensures that the Test Tool receives current information from the Policy Server.

To run a functionality test

1. Configure a test environment.

   Note: You can also test policies using the Scripting Interface. See the *Programming Guide for Perl*.

2. (Optional) Specify the number of times you want the Test Tool to run your test in the Repeat Count field in the Command group box.

3. In the Command group box, select one of the following tests to run:

   - IsProtected
   - IsAuthenticated
   - IsAuthorized

4. If you are running an IsAuthenticated test and you want the Test Tool to validate an authenticated user's session ticket instead of authenticating the user's credentials against a user directory, enter Validate in the Comment field.

Note: Before validating a user's session ticket, the user must be authenticated. Once the user is authenticated, SiteMinder creates a session ticket for the user.

More information:

Configure Your Test Environment Agent (see page 684)
Calculate an Average Elapsed Time (see page 693)

Functionality Test Results

The tables in this section describe the results of each type of functionality test.

If isProtected...	Then...
Succeeds	The Test Tool displays Protected in the Message field. This means that the Test Tool made a successful connection to the Policy Server and a policy is protecting the resource. The Test Tool also populates the following fields with values returned by the Policy Server: **Realm Name** Name of the realm that contains the resource **Realm OID** The realm object identifier **Credentials** The authentication scheme used to protect the resource **Redirect** The redirect string used by the authentication scheme, if one is specified. All certificate and HTML forms-based schemes return this string, which typically instructs the Agent where to display a form.
Fails	The Test Tool displays Error or Not Protected in the Message field. Error indicates that the Test Tool could not connect to the Policy Server; Not Protected indicates that the specified resource is not protected by a policy. If the test fails: Make sure that the policy is configured correctly. Check the Authentication server log for debugging information.

If isAuthenticated...	Then...
Succeeds	The Test Tool displays Authenticated in the Message field and populates the following fields with values returned by the Policy Server: **Session ID** A unique SiteMinder-assigned session ID. The Policy Server uses this ID to identify the cookie where session information is stored. **Attributes** The attributes the Policy Server sends back in the response. For example: ID Length ASCII Format Hexidecimal format 4> id 215 len 005 'LDAP:' - '4c 44 41 50 3a' The response indicates the name of the user directory where the user was authenticated. **Note:** Click Reset to clear responses displayed in the Attributes field without removing user-supplied information. Reason The reason code associated with the outcome of the test. This field is used to supply information to developers using the SiteMinder SDK. Reason codes are listed in SmApi.h.
Fails	The Test Tool displays Not Authenticated in the Message field. If the test fails: Make sure that you are using valid user credentials. Check the Authentication server log for debugging information.

If IsAuthorized...	Then...
Succeeds	The Test Tool displays Authorized in the Message field and the SiteMinder-assigned Session ID in the Session ID field. This ID identifies the cookie where session information is stored.
Fails	The Test Tool displays Not Authorized in the Message field. If the test fails: Make sure that the policy is configured correctly. Check the Authorization server log for debugging information.

Calculate an Average Elapsed Time

After performing a test, the Test Tool displays the amount of time the test took to run in the Elapsed Time field of the Command group box. Because of fluctuations in the system, averaging the elapsed time of multiple tests provides more accurate results.

To get an average elapsed time

1. In the Repeat Count field, specify the number of times you want to run the test.

   The Test Tool runs the test the specified number of times and then displays the total elapsed time.

2. Divide the elapsed time by the number of times the test was run to determine the average elapsed time.

Perform a Regression Test

Regression tests allow you to test whether or not changes made to SiteMinder, such as upgrading the policy store or implementing a new feature, affect policies. To run a regression test, you run one test with the current environment, make changes, then run the test again. By comparing the results of the tests, you can determine if the changes affect SiteMinder.

To perform a regression test

1. Run functionality tests in Record mode. Optionally, enter a comment in the Comment field. The comment appears in the output file.
2. When the test is complete, select Basic Playback in the Mode group box.
3. In the Script Information group box, enter the name of the text file in the Input Script field.

   This file name should match the name of the Output Script file you created in Record mode.

4. In the Output Script field, specify an output file name.
5. In the Command group box, click Run Script.

   The Test Tool runs the input script and creates the output script file.

6. Compare the input and output script files.

Run a Stress Test

The Test Tool allows you to test SiteMinder's performance when the Policy Server receives more than one request at a time. Using stress tests, you can simulate multiple agents talking to the Policy Server simultaneously or a single agent communicating with the Policy Server on multiple threads.

Stress tests are run in Advanced Playback mode. The Test Tool receives instructions from a Thread Control file that specifies which tests to run and how many times to run them. After executing the instructions in the thread control file, the results of the test are written to an output file.

More information:

Configure a Thread Control File (see page 694)

Configure a Thread Control File

Configure a Thread Control file to define the Command Script files to run, number of repetitions, threads, and so on to automate complex tests for stress testing. A Thread Control file contains multiple instruction lines in the Test Tool's own scripting language and comments, indicated by the # symbol at the beginning of a line.

The basic instructions are in the following format:

command_script_file_name, *repetition_count*, *thread_count*, [*max_time_in_seconds*]

command_script_file_name

Specifies the pathname of the previously recorded command script file.

repetition_count

Specifies the number of times that the Test Tool runs the Command Script.

thread_count

Specifies how many simultaneous threads the Test Tool initiates to run the Command Script.

max_time_in_seconds

(Optional) Specifies a limit (in seconds) for the duration of the test. If elapsed test time exceeds this limit then playback halts, regardless of whether the configured number of repetitions are complete.

For example:

```
# c:\temp\test_data.txt, 8, 6, 120
```

This line specifies that:

- The input Command Script is c:\temp\test_data.txt
- The Test Tool runs the script eight times
- Six simultaneous threads run the script
- The test ends after 120 seconds, regardless of whether eight repetitions are complete

The Test Tool writes the output of the test to a file. The output file name is the name of the input file with _out# appended to it, where # is the incremented thread number. For example, the output files for the test above are c:\temp\test_data.txt_out1 to c:\temp\test_data.txt_out6.

The Test Tool scripting language includes the commands in the following table to control the script file output. See the following figure for a sample thread control file.

Command	Description
.report	Generates a final report (as an output file) summarizing the test results. The report does not include the status of each server request. This is the default.
.reportspread	Generates a file containing a response time spread report. The file name is *command_script*_stats_spread (for example, isprotected-record.txt_stats_spread).
.output	Generates a final report (as an output file) summarizing the overall results including the status of each server request.
.viewstats	Displays final statistics in a text editor.
.verbose	Generates output files containing the details of each server request.
.brief	Generates output files containing the brief results of each server request. This option is only valid when used with the .output command.
.sleep	Lets the Test Tool pause for a specified amount of time (in milliseconds). This simulates intermittent server requests.

Command	Description
.userselectionmode	Determines whether usernames are used sequentially or selected randomly. Valid values are 0 (sequential) and 1 (random). For more information, see (Optional) Configure How the Test Handles Usernames.
.randomseed	Allows the pseudo-random sequence of names to be repeated (up to thread ordering) for each test. For more information, see (Optional) Configure How the Test Handles Usernames.
.connect *settings_file*	Initializes the Test Tool using information from the Test Tool Settings file to set up a multi-threaded test with one simulated Agent. The default multi-threaded test comprises multiple simulated Agents with one simulated Agent per thread. You can also set up a test with one simulated Agent and multiple threads by using this option.
.disconnect	Un-initializes the Test Tool to indicate the end of one simulated Agent multi-threaded test.

Example Thread Control File

```
.output

.connect c:\test\smtest.ini
.brief
c:\temp\test_data1.txt, 2, 3
.verbose
.sleep 5000
c:\temp\test_data1.txt, 2, 2
.brief
c:\temp\test_data1.txt, 3, 4
.connect smtest.ini
c:\temp\test_data1.txt, 5, 6
.disconnect
```

Report Viewing

When you run a stress test, the Test Tool generates a report summarizing the results. This report contains the following information:

- Time the test started and finished
- Total elapsed time
- Minimum, maximum, and average request time

- Total number of requests
- Throughput
- The number of tests run and their results

The report is saved in the directory where the thread control file is located. The name of the report is the name of the thread control file with _stats appended to it. For example, the thread control file, thread.txt, yields a report named thread.txt_stats.

```
Control File:   C:\temp\control.txt
Started at:     0:02:05.481
Finished at:    0:03:22.672
Total Elapsed:  0:00:01.396

Minimum Request Time:   0:00:00.400
Maximum Request Time:   0:00:05.498
Average Request Time:   0:00:01.396

Total Requests          234
Throughput (Req/Sec):   3.241

Request         Count   Yes     No     Timeout   Error
--------------- ------  ------  -----  --------  -----
IsProtected     78       72     0      0         6
IsAuthenticated 78       78     0      0         0
IsAuthorized    78       72     0      6         0
------------------      ------  -----  -------   -----
Total:          234     222     0      6         6
```

Certificate-based Authentication Tests

The SiteMinder Test Tool simulates user authentication and authorization. Certificate-based authentication schemes require additional configuration.

Different certificate-generation tools sometimes affect the format of the Issuer DN and other attributes of a certificate.

For example, the Issuer DN for a certificate generated with certutil.exe on an IIS web server could use ST= to represent the state. However, the Issuer DN for a certificate generated with OpenSSL tools on an Oracle iPlanet web server could possibly use S= to represent the state.

Note: For more information about the actual values used by a specific certificate-generation tool, see the documentation provided by the vendor of your certificate-generation tool.

To test certificate-based authentication schemes, configure certificate mappings in the Policy Server to accommodate certificates created with different certificate-generation tools.

Certificate Attributes that Require Custom Mappings

Some common certificate attributes differ slightly according to the third-party tool (such as certutil.exe or OpenSSL) used to generate the certificate. Differences between the following attributes could possibly cause errors in the SiteMinder Test Tool:

Email Address

Represented by E or Email depending on the vendor of the certificate-generation tool.

US State

Represented by S or ST depending on the vendor of the certificate-generation tool.

User ID Number

Represented by UID or UserID depending on the vendor of the certificate-generation tool.

Note: For more information about the actual values used by a specific certificate-generation tool, see the documentation provided by the vendor of your certificate-generation tool.

Custom Attribute Mappings for Testing

Using the SiteMinder Test Tool for a certificate authentication scheme sometimes fails, even if it works typically (through a browser and the web server). The authentication log shows that the Test Tool expects a different format of the Issuer DN than the Issuer DN format used in the certificate.

This situation occurs when the Issuer DN and other attributes differ according to the type of certificate-generation tool used. For example, the certutil.exe program on an IIS web server could possibly use ST= to abbreviate the name of the state in the Issuer DN. The OpenSSL tools on an Oracle iPlanet web server, however, could possibly use S= to abbreviate the name of the state.

Note: For more information about the actual values used by a specific certificate-generation tool, see the documentation provided by the vendor of your certificate-generation tool.

The situation is similar for the other attributes listed in Certificate Attributes that Require Custom Mappings (see page 698).

To resolve this problem, have an administrator create mappings for each Issuer DN format in the Policy Server. Then, the Policy Sever can accept the Issuer DN formats created by different certificate-generation tools.

Issuer DN Mapping

Different certificate-generation tools (such as certutil.exe and OpenSSL) create the Issuer DN in slightly different ways. For example, one tool could possibly create an Issuer DN like the following:

```
CN=Personal Freemail RSA 2000.8.30, OU=Certificate Services, O=Thawte, L=Cape Town, S=Western Cape, C=ZA
```

Another tool could possibly create an Issuer DN like the following:

```
CN=Personal Freemail RSA 2000.8.30, OU=Certificate Services, O=Thawte, L=Cape Town, ST=Western Cape, C=ZA
```

To support multiple possibilities, have your administrator create mappings in the Policy Server for all Issuer DN formats in your environment.

Note: For more information about the actual values used by a specific certificate-generation tool, see the documentation provided by the vendor of your certificate-generation tool.

Create Custom Certificate Mappings

You can use the certificate-mapping feature of the SiteMinder Policy Server to provide custom mappings for certificates.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

To create and use a custom attribute in a certificate mapping

1. Click Infrastructure, Directory.
2. Click Certification Mapping, Create Certificate Mapping.

   The Create Certificate Mapping pane opens.
3. Verify that Create a new object is selected, and click OK.

   Certificate mapping settings open.

   Note: Click Help for descriptions of settings and controls, including their respective requirements and limits.
4. Enter the full issuer DN in the Issuer DN field.
5. Select the Custom radio button in the Mapping group box.

   The Mapping Expressions field opens.
6. Enter a custom mapping expression.

   This notation is used to specify two different attributes that are acceptable for a certificate mapping.

   - For Email:

     %{E/Email}
   - For ST:

     %{S/ST}
   - For User Id:

     %{UID/UserID}

   Note: More information about custom mapping expressions exists in Certificate Attributes that Require Custom Mappings (see page 698).
7. Click Submit.

   The custom mapping is saved. The Policy Server now handles requests from different types of certificate-generation tools (such as certutil.exe and OpenSSL) and the SiteMinder Test tool where the Email attribute is represented differently in the issuer DN. You can use this process for any of the other attributes mentioned in Certificate Attributes that Require Custom Mappings (see page 698).

More Information:

Certificate Mapping for X.509 Client Authentication Schemes (see page 403)

Appendix A: Troubleshooting SSL Authentication Schemes

This section contains the following topics:

Overview

Configuring the SSL Advanced Authentication Schemes requires Web Servers to be properly configured to use SSL. Most of the problems you may encounter configuring Authentication Schemes over SSL connections are likely to be SSL configuration issues. Therefore, the first step in troubleshooting Authentication Schemes over SSL is to verify that SSL is properly configured and working. This is done without the interaction of the SiteMinder Web Agent so that these components can be individually analyzed.

Determine SSL Connection Ability

The first step in troubleshooting Authentication Schemes over SSL is to verify that SSL is properly configured and working. This is done without the interaction of the SiteMinder Web Agent so that these components can be individually analyzed.

To determine whether you are able to establish an SSL connection

1. Disable the SiteMinder Web Agent protecting the realm for which you want to use an authentication scheme over SSL.

   Note: For information about disabling a Web Agent, see the *Web Agent Configuration Guide*.

2. Using your browser, go to one of the following URLs (using a browser with a certificate):

   - https://web_server_name:port (Netscape Web Servers)
   - https://web_server_name:port/<SSL Virtual Directory> (IIS Web Servers)
   - https://web_server_name:port (Apache Web Servers)

   If this SSL connection is configured to require certificates, you will be prompted to select a certificate.

If you are unable to successfully establish this SSL connection, then see SSL Configuration (see page 704) for more information on configuring SSL. If you were able to establish this connection, but have not been successful in configuring SiteMinder, see SSL Troubleshooting (see page 707).

SSL Configuration

It is imperative that SSL be configured and working properly before using SiteMinder. In order to make an SSL connection, you must be able to trust the certificate authority of an incoming certificate. For example, if a browser presents a certificate that was signed by VeriSign, you must have a VeriSign Certificate Authority installed and trusted in the Web Server. In addition to trusting client certificates that are presented, the server itself must have a certificate to present to the clients. The clients have to trust the Certificate Authority that issued the certificate. This allows for mutual authentication. Once these certificates have been installed, you can configure the Web Server to use SSL and *require* certificates, if desired.

For detailed SSL configuration information, see the documentation provided with your web server software. This section contains step-by-step instructions for configuring your Web Server and Web browser to successfully establish an SSL connection. If you have correctly configure SSL, but are still having problems making the connection, see the common problems at the end of the section.

Enable the Web Server to Trust Client Certificates in Netscape

If a certificate authority is already installed in the Web Server, go on to the next section. Otherwise, install a certificate for the Certificate Authority on the SSL Web Server.

To enable the Web Server to Trust Client Certificates in Netscape

1. Obtain the Certificate Authority's certificate and either keep it on your screen or save it to a file.
2. In Netscape Server Administration, select Keys & Certificates.
3. Click Install Certificate.
4. In the Certificate For field, fill out the Server Security Chain.
5. In the Certificate Name field, enter a description.
6. If you saved your Certificate Authority's certificate to a file, enter the file name in the Message is in this file field; otherwise, select the Message text (with headers) radio button and paste the certificate in the Message text (with headers) field.
7. Click OK and restart the Web Server.

Configure the Netscape Web Server to use SSL

After installing the Netscape Web Server Certificate, you must configure the Netscape Web Server to use SLL by requiring certificates.

To require certificates for your SSL Web Server

1. In Netscape Server Administration, click Admin Preferences.
2. Click Encryption On/Off and ensure that Encryption is on.
3. If you are running the Certificate or Certificate with Basic Authentication Scheme, you must require certificates. This is done under Encryption Preferences setting where Require Certificates must be set to On. From a browser that has a Certificate installed, verify that you can get to https://servername:port.

Note: Do not turn on Required Certificates for the Certificate or Basic Authentication Scheme.

Establish Trust for the Netscape Certificate Authority

If a certificate authority is installed in the Web Server, you can establish trust between the two.

To establish trust for the Netscape Certificate Authority

1. In Netscape Server Administration, select Keys & Certificates.
2. Select Manage Certificates.
3. Select the Certificate Authority. The system displays a dialog detailing the certificate.
4. Select Trust.
5. Click OK and restart the Web Server.

Enable the Web Server to Trust Client Certificates in Windows

You must trust your client certificates by installing the appropriate Certificate Authority Certificates.

SSL Web Servers must have certificates for each Certificate Authority. Major certificate authorities may already be installed. You can configure certificates in Windows operating systems by using the Certificates snap-in. For information, see your Windows documentation.

Configure the IIS Web Server to use SSL

Be sure that a secure port has been enabled on the Web Server. Generally this is port 443. You can verify this through the Management Console by right-clicking on the Web Server and in the Web site tab you will see an SSL Port. Be sure a port number has been installed.

The advanced authentication schemes will create virtual directories in the Web Server. These directories will automatically be configured to require SSL and certificates as required by the specific authentication scheme. However, for testing purpose, you may want to create a test virtual directory. You can configure this virtual directory to require certificates through the Directory Security tab, Secure Communications.

https://servername:port/virtual directory - Ensure that the browser is asked for a certificate.

Install the IIS Web Server Certificate

If you have not already done so, you will need to generate a key for your Web server. This is done through the Management Console, Key Manager. Access the Key Manager by doing the following:

Note: Note this process may be slightly different for IIS 3 and IIS 4.

To install the IIS Web Server Certificate

1. In the Management Console, right-click the Web Server and select Properties.
2. Click the Directory Security tab.
3. In the Secure Communications panel, click Key Manager.
4. Under Key, select Create New Key and a Wizard will guide you through the process.

   Once you create a key, you can request a certificate using the file created in the steps mentioned earlier. Go to the Certificate Authority and request a certificate for this server. You will need to paste the certificate request information generated in Step 1 in order to receive a certificate. Once you received a certificate, go back to Management Console, Directory Security and click Key Manager to install the certificate for the key described in the next step.
5. Right-click the key name and select Install Certificate.
6. Restart the Web Server.

Enable the Web Server to Trust Client Certificates in Apache

If a certificate authority is already installed on your web server, go on to the next section. Otherwise, install a certificate for the CA on the SSL Web Server as follows.

To enable the Web Server to trust client certificates in Apache

1. Download and build the following Apache components:
   - Apache
   - OpenSSL
   - Mod_SSL
   - Mod_so - This module is included as part of Apache, but it must be enabled during the build of the Web server.
   - RSAref-2.0

     See the *Policy Server Installation Guide* for details about installing the web server.
2. Copy the CA certificate into the apache/conf/ssl.crt directory in x509 b64 format.
3. Run make in the apache/conf/ssl.crt directory.
4. Restart the Web Server.

Installing the Apache Web Server Certificate

The process for installing a certificate on an Apache Web Server varies with individual configurations. Consult the documentation for Mod_SSL and OpenSSL for details about how to configure these components.

SSL Troubleshooting

The following sections detail the most common problems encountered when dealing with SSL authentication schemes.

There Was No Prompt for a Certificate

If you were not prompted for a certificate, verify that SSL is configured appropriately. If the Web Agent is installed, disable the Web Agent. The first step is to verify a simple SSL connection.

To determine whether you are able to establish an SSL connection

1. Disable the SiteMinder Web Agent protecting the realm for which you want to use an authentication scheme over SSL.

   Note: For information about disabling a Web Agent, see the *Web Agent Configuration Guide*.

2. Using your browser, go to one of the following URLs (using a browser with a certificate):

   - https://web_server_name:port (Netscape Web Servers)
   - https://web_server_name:port/<SSL Virtual Directory> (IIS Web Servers)
   - https://web_server_name:port (Apache Web Servers)

   If this SSL connection is configured to require certificates, you will be prompted to select a certificate.

After Following Previous Procedure, Still No Certificate Prompt

Perform the following five additional steps if you are still not receiving a certificate prompt.

- Verify that all Firefox browsers are configured to ask every time.
- Verify that all web servers are configured to use SSL and require certificates.
- Verify the following settings for each SiteMinder Virtual Directory.
- Verify the web server certificate expiration.
- Verify browser certificate validity.

Verify That All Firefox Browsers Are Configured to Ask Every Time

Firefox browsers can be configured to pass the same certificate automatically. This establishes the SSL connection using a certificate without prompting users to select a certificate.

Follow these steps:

1. In the Firefox browser, select Options from the Firefox menu.
2. Click Advanced.
3. Click the Encryption tab.
4. In the Certificates section, verify that the Ask me every time option is set.

Verify That All Web Servers Are Configured to Use SSL and Require Certificates

For Netscape Web Servers

1. In the Netscape Server Administration, click Admin Preferences.
2. Click Encryption On/Off and verify that the encryption is on, then click OK.
3. Click Encryption Preferences and verify that Required Certificates is set.
4. Restart the Web Server.

For IIS Web Servers

Verify that the virtual directories SMGetCredCert, SMGetCredCertOptional, SMGetCredNoCert are created and have the correct settings.

- SMGetCredCert - Require Certificates will be selected
- SMGetCredCertOptional - Accept Certificates will be selected
- SMGetCredNoCert - Do not accept certificates will be selected

Note: As part of the SiteMinder SSL Authentication setup, SiteMinder configures SSL virtual directories based on the type of SSL connection required by the authentication scheme.

Verify the Following Settings for each SiteMinder Virtual Directory

To verify the following settings for each SiteMinder Virtual Directory

1. In the Management Console, right-click a virtual directory and select Properties.
2. Click the Directory Security tab.
3. Click Edit Secure Communications.

For Apache Web Servers

In the httpd.conf file, be sure to set SSLVerifyClient as follows:

- For Basic over SSL: SSLVerifyClient none
- For Certificate or Basic: SSLVerifyClient optional
- For Certificate/Certificate and Basic: SSLVerifyClient require

  Note: For Apache Web servers where Certificates are required or optional, the "SSL Verify Depth 10" line in the httpd.conf file must be uncommented.

Check the Web Server's Certificate Expiration

Netscape Servers

1. In the Netscape Server Administration, click Keys & Certificates.
2. Click Manage Certificates.

3. Click ServerCert.
4. Verify that it is trusted, and has not expired. If it does not exist, or has expired, you will need to request a new certificate by following the steps in Install the Netscape Web Server Certificate.

IIS Servers

1. In the Management Console, right-click the Web Server and select Properties.
2. Click the Directory Security tab.
3. In the Secure Communications panel, click Key Manager.
4. Select a key to view its properties and verify that the key has not expired.
5. If you need to make any changes, restart the Web Server.

Apache Servers

If an Apache Web Server certificate expires, you will receive an error messages at server startup that indicates the certificate has expired.

Verify Browser Certificate Validity

A missing certificate or an invalid certificate can prevent you from receiving a certificate prompt.

Open your Web browser and verify the validity of the browser certificate.

Note: For more information about viewing certificate information, see your vendor–specific documentation.

After Certificate Prompt, Authentication Failure Received

Apache Web Servers

- Verify that the SSL Web Server contains the certificate authority of the certificate supplied.
- Verify that the SSL Web Server Trusts the certificate authority of that certificate.
- Ensure the SSL Verify Depth 10 is uncommented.

Netscape Web Servers

Verify that the Certificate Authority for the certificate is listed and that the Trust for the certificate has not expired. If it is not there or is not valid, install a new CA certificate.

IIS Web Servers

Verify that the certificate is listed and that it is valid. If it is not present or is not valid, install a new certificate. If you are able to get to the destination directory, then certificates are installed correctly.

Verify Correct Policy Server and Web Agent Configuration

After completing the steps in the previous topic based on your specific web server, verify your policy server and web agent configuration.

To verify correct policy server and web agent configuration

1. Check that the Policy Server is created correctly.
2. Check that the Web Agent contains the correct Policy Server information.
3. Verify that the Web Agent is enabled.
4. Restart the Web Agent and Policy Server.

SiteMinder Policy Should Allow Access, but SSL-Authentication Failed Message Received

In this situation, there is a Policy that is being called, but the user is incorrectly being denied access. This can result from a number of configuration errors. Common errors include:

- The SSL Server is not configured to Require Client Certificates. Therefore, the client is not passing a certificate; thereby disabling SiteMinder authentication process. You can verify this is the situation by enabling the logging option in the Web Agent. The log should indicate that the user is unknown. To correct this problem, turn on Require Certificates in the SSL Web Server.
- The Policy was not created properly. Check the Policy's users and be sure that the selection is correct.
- For Apache Web server, ensure the SSL Verify Depth is set properly and uncommented.

More information:

How to Configure a Policy Domain (see page 495)
Certificate Mapping for X.509 Client Authentication Schemes (see page 403)

Error Not Found Message Received

This is generally caused from the Authentication Scheme Parameter being configured improperly. The redirect is not configured properly so the Web Server is unable to find the SSL Web Agent component.

More information:

Authentication Schemes (see page 301)

Running Certificate or Basic but Cannot Enter Basic credentials.

On Netscape Web Servers, the *Certificate or Basic* scheme requires the Web Server to have encryption turned on, but does not require certificates. Be sure that in the Encryption Preferences section of the Netscape Server Administration, the Require Certificate setting is set to No.

Appendix B: LanMan User Directories

This section contains the following topics:

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:

  installation_directory\netegrity\siteminder\bin\

- You must configure a connection to the LanMan directory.

More information:

Configure a LanMan Directory Connection (see page 714)

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 714)
2. Configure a LanMan User Directory Connection

Configure Registry Keys for a LanMan Directory Connection

The first procedure in configuring a LanMan directory connection is configuring the appropriate registry keys.

Follow these steps:

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: For more information about starting and stopping the Policy Server, see the *Policy Server Administration 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 that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

To configure a LanMan user directory connection

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

   The User Directories page appears.
3. Click Create User Directory.

   The Create User Directory page 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 user directory.
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 157)
Configure Registry Keys for a LanMan Directory Connection (see page 714)

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 242)

Appendix C: CA SSO/WAC Integration

This section contains the following topics:

Overview

SiteMinder provides single sign-on from SiteMinder to the CA SSO environments. Users log in to a SiteMinder or CA SSO environment, and once authenticated by SiteMinder, are authenticated for both environments. Authenticated users can access protected resources in either environment without having to reenter credentials, as long as they are authorized. User authorization is based on the policies in effect within each environment.

When allowing users to access secure resources, SiteMinder and CA SSO each maintain user credentials in their own session stores. They also have their own proprietary session credentials that cannot be read by the other and, thus, user credentials are maintained separately. Since these credentials reside in different stores, to enable single sign-on, the SiteMinder Policy Server and CA SSO Policy Server must be part of the same cookie domain and must share the same user or authentication store.

In this single sign-on configuration, the SiteMinder and CA SSO Policy Servers can be on the same or on different machines. SiteMinder can contain a Web Agent, CA SiteMinder SPS, or both. You use a Web Agent or CA SiteMinder SPS based on your own SiteMinder environment. CA SSO uses the eTrust Web Access Control (WAC) Web Agent, and you do not need to modify your current environment to enable single sign-on with SiteMinder.

Note: You must be intimately familiar with SiteMinder and CA SSO before configuring single sign-on between the products. For a list of supported SiteMinder, CA SiteMinder SPS, CA SSO, and eTrust WAC versions, refer to the 6.0 SiteMinder and Agents Platform Matrix on the Technical Support site.

More information:

SiteMinder and CA SSO Integration Architectural Examples (see page 718)

SiteMinder and CA SSO Integration Architectural Examples

The following are three examples of single sign-on between SiteMinder and CA SSO environments:

1. A user authenticates to SiteMinder using a Web browser and then accesses an CA SSO-protected resource (see Example 1: User Accesses SiteMinder-Protected Resource Before CA SSO (see page 719)).
2. A user authenticates to CA SSO through a desktop CA SSO Client and then accesses a SiteMinder-protected resource using a Web browser (see Example 2: Authenticated CA SSO Client User Accesses SiteMinder Resource (see page 720)).
3. A user authenticates to CA SSO using a Web browser and then accesses a SiteMinder-protected resources (see Example 3: User Accesses CA WAC-Protected Resource Before SiteMinder (see page 722)).

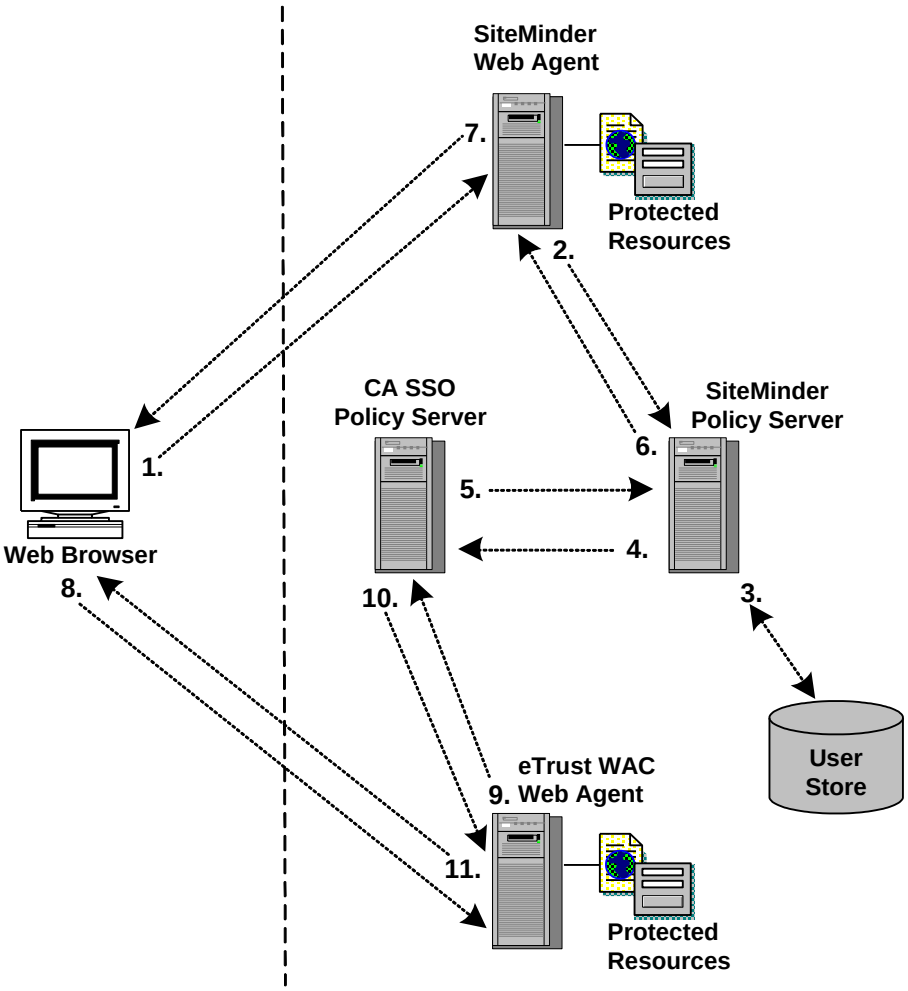
Note: In these examples, the SiteMinder and CA SSO Policy Servers authorization access steps to protected resources are omitted for clarity.

More information:

Configure Single Sign-On from CA SSO Client to SiteMinder (see page 727)

User Accesses SiteMinder-Protected Resource Before CA SSO

The following example illustrates a user accessing SiteMinder-protected resource before a WAC-protected resource:

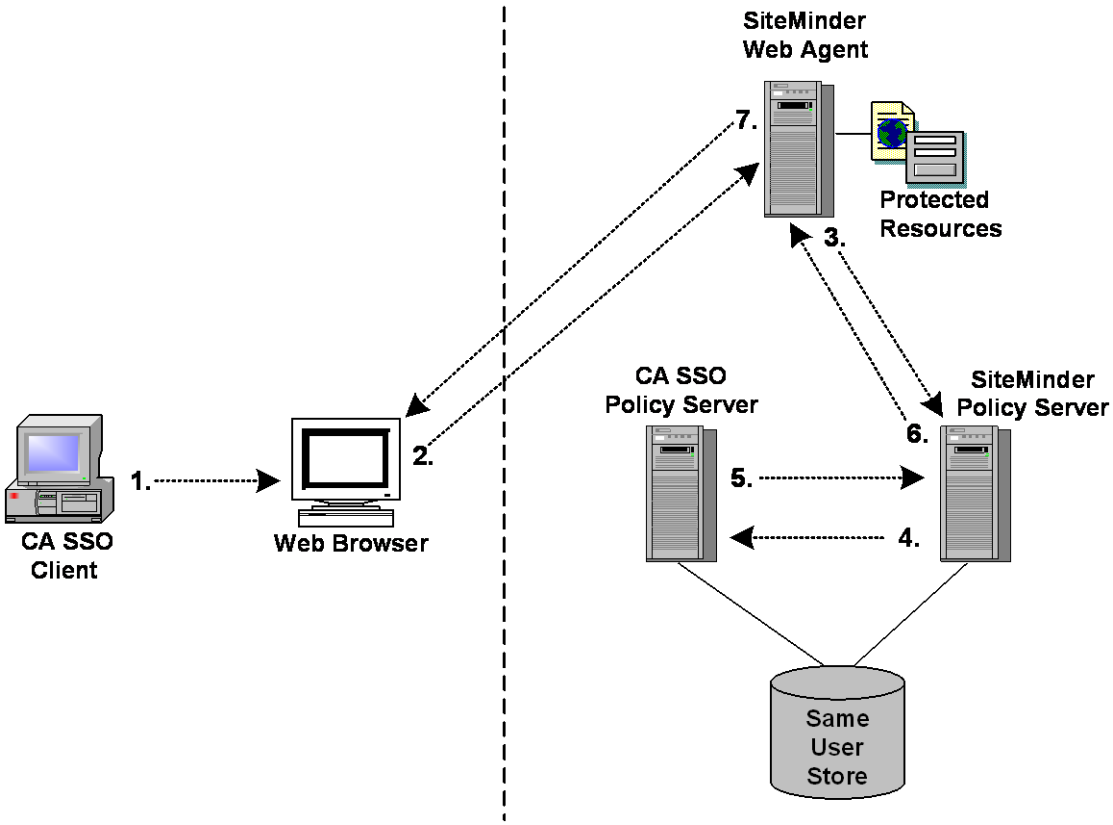


1. The user tries to access a SiteMinder-protected resource and the SiteMinder Web Agent/CA SiteMinder SPS intercepts the request. The user provides the Agent/SPS with authentication credentials.
2. The Web Agent/CA SiteMinder SPS forwards the credentials to the SiteMinder Policy Server for validation.
3. The SiteMinder Policy Server verifies that the user's credentials are valid in the user store.
4. After successful authentication, the SiteMinder Policy Server requests the CA SSO Policy Server to issue and return an CA SSO cookie for the SiteMinder user.
5. The CA SSO Policy Server validates the user and forwards the user's CA SSO web authentication credentials to the SiteMinder Policy Server.

6. The SiteMinder Policy Server forwards the CA SSO web authentication credentials to the SiteMinder Web Agent/CA SiteMinder SPS.
7. The SiteMinder Web Agent/CA SiteMinder SPS sets the CA SSO web authentication and SiteMinder cookies in the user's browser and returns the resource to the user.
8. The user tries to access an CA SSO resource and the eTrust WAC Web Agent intercepts the request.
9. The eTrust WAC Web Agent validates the user's CA SSO web authentication cookie credentials with the CA SSO Policy Server.
10. The CA SSO Policy Server tells the eTrust WAC Web Agent that the user has valid credentials.
11. The eTrust WAC Web Agent allows the user to access the CA SSO-protected resource.

Authenticated CA SSO Client User Accesses SiteMinder Resource

The following example illustrates an authenticated CA SSO client user accessing a SiteMinder protected resource:

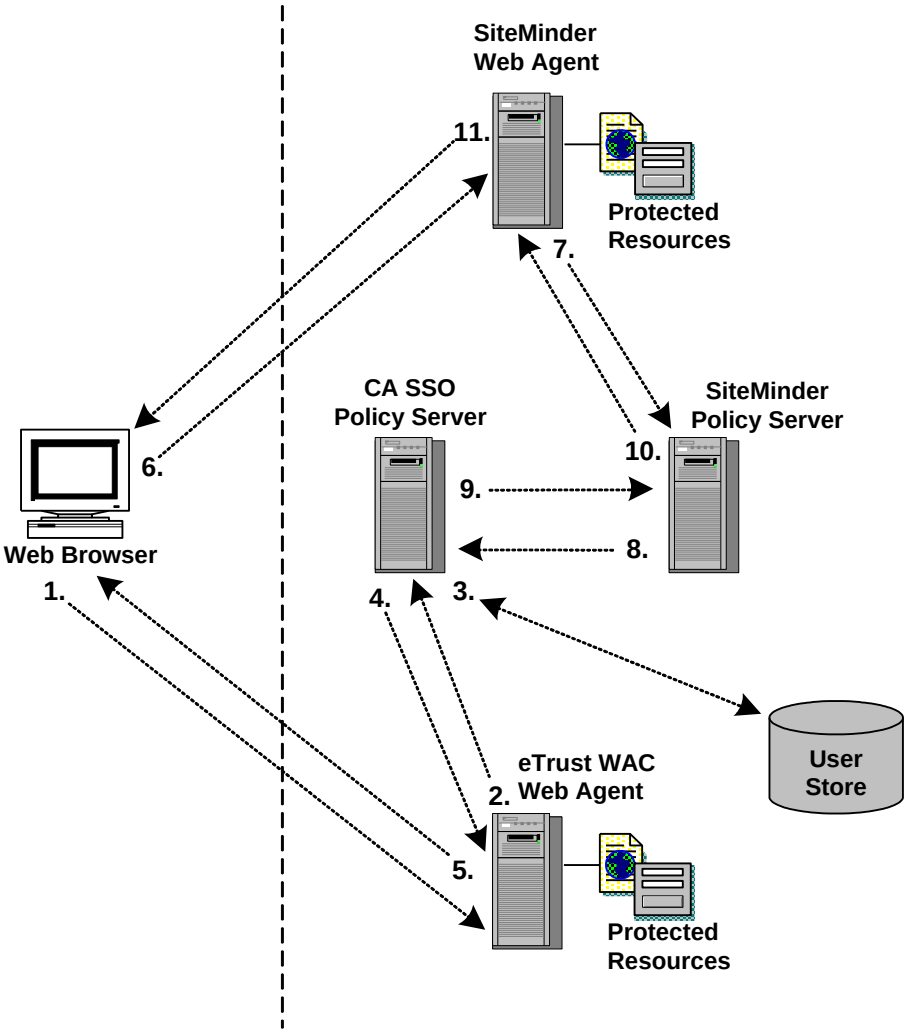


1. An authenticated CA SSO Client user launches a Web browser. While this is happening, the CA SSO Client places an CA SSO Web authentication cookie into the browser.
2. The user tries to access a SiteMinder-protected resource using the Web browser and the request is intercepted by the SiteMinder Web Agent/CA SiteMinder SPS.
3. The SiteMinder Web Agent/CA SiteMinder SPS forwards the CA SSO Web authentication cookie to the SiteMinder Policy Server.
4. The SiteMinder Policy Server forwards the CA SSO Web authentication cookie to the CA SSO Policy Server.
5. The CA SSO Policy Server validates the CA SSO Web authentication cookie and returns the user name to the SiteMinder Policy Server.
6. The SiteMinder Policy Server verifies the returned user name in the SiteMinder user store, then issues a corresponding SiteMinder cookie and returns it to the SiteMinder Web Agent/CA SiteMinder SPS.
7. The SiteMinder Web Agent/CA SiteMinder SPS returns the requested resource to the user, who now has the authentication cookie credentials necessary for SiteMinder and CA SSO environments.

User Accesses eTrust WAC-Protected Resource Before SiteMinder

The following example illustrates a user accessing a WAC-protected resource before SiteMinder.

Note: The example assumes the environment is using a IIS6 WAC Agent. An IIS6 WAC Agent is the only platform that the following example supports.



1. The user tries to access an CA SSO-protected resource and the eTrust WAC Web Agent intercepts the request. The user provides the Agent with authentication credentials.
2. The Web Agent forwards the credentials to the CA SSO Policy Server for validation.
3. The CA SSO Policy Server makes sure that the user's credentials are valid in the user store.

4. The CA SSO Policy Server forwards the user's eTrust SSO Web credentials to the eTrust WAC Web Agent.
5. The eTrust WAC Web Agent sets the user's CA SSO Web authentication cookie in the Web browser.
6. The user tries to access a SiteMinder-protected resource and the SiteMinder Web Agent/CA SiteMinder SPS intercepts the request.
7. The SiteMinder Web Agent/CA SiteMinder SPS forwards the user's CA SSO Web authentication credentials to the SiteMinder Policy Server.
8. The SiteMinder Policy Server forwards the user's CA SSO Web authentication credentials to the eTrust SSO Policy Server.
9. The CA SSO Policy Server validates the user's CA SSO Web authentication credentials and forwards the user name back to the SiteMinder Policy Server.
10. The SiteMinder Policy Server verifies the returned user name in the SiteMinder user store, then issues a corresponding SiteMinder cookie and returns it to the SiteMinder Web Agent/CA SiteMinder SPS.
11. The SiteMinder Web Agent/CA SiteMinder SPS sets the SiteMinder cookies in the user's browser and allows the user to access the requested resource.

SiteMinder and CA SSO Integration Prerequisites

Before configuring a single sign–on integration between SiteMinder and CA SSO:

1. Install and configure CA SSO.

   Note: When installing the CA SSO Policy Server, gather the following:

   - An SSO administrator name and password. The SiteMinder Policy Server uses the administrator name and password when authenticating to the CA SSO Policy Server through the smauthetsso authentication scheme.
   - The SSO ticket encryption key. The SiteMinder Policy Server smetssocookie active response requires this value.

2. Be sure that the SiteMinder environment and the CA SSO environment are operating in the same FIPS mode (AES encryption) of operation.

   Important! The integration fails if both environments are not operating in the same FIPS mode of operation.

3. Consider the following:
   - If the integration is to operate in FIPS–only mode, SiteMinder Policy Servers must be operating at r12.0 SP1 CR3 or later. If necessary, upgrade the SiteMinder Policy Servers that are to communicate with the CA SSO Policy Server.
   - Be sure that operating systems on which the Policy Servers are installed support the smauthetsso authentication scheme. The integration requires that you configure the smauthetsso authentication scheme. For more information, see the r12.5 SiteMinder Platform Support Matrix.

Configure Single Sign-On from SiteMinder to CA SSO

SiteMinder provides single sign-on from SiteMinder to CA SSO environments.

To enable single sign-on from SiteMinder to CA SSO using a SiteMinder Web Agent or CA SiteMinder SPS

Enable the SiteMinder SSO Plug-in installed with the Web Agent or CA SiteMinder SPS:

For the r12.5 IIS 6.0 or Apache 2.0 Web Agent

- Remove the comment (#) character from *one* of the following lines in the WebAgent.conf file:
  - (Windows operating environments)
    #LoadPlugin=*Path_to_eTSSOPlugin.dll_file*
  - (UNIX or Linux operating environments)
    #LoadPlugin=*Path_to_libetssoplugin.so_file*

Note: Restart the Web server after you modify the WebAgent.conf file so the new configuration settings take effect.

For the 6.0 CA SiteMinder SPSr

- Remove the comment (#) character from the following line in the WebAgent.conf file, located in <SPS_install_dir>\proxy-engine\conf\defaultagent\WebAgent.conf:

  #LoadPlugin=<Path to eTSSOPlugin.dll or libetssoplugin.so>

Note: Restart CA SiteMinder SPS after you modify the WebAgent.conf file so the new configuration settings take effect.

To enable single sign-on using the WAC Web Agent

1. Configure the domain in the WAC Web Agent's webagent.ini file by setting the following parameter:

   DomainCookie=<domain>

   where <domain> is the same domain (for example, test.com) for the CA SSO and SiteMinder Web Agents.

   The file is installed in the following location on the WAC Web Agent machine:

   C:\Program Files\CA\WebAccessControl\WebAgent\webagent.ini

2. Verify the following Web server and the authentication method settings in the webagent.ini file:

   - The "Authentication methods" and the "Default authentication method" parameters should be configured as SSO.
   - The WebServerName, PrimaryWebServerName, AgentName, NTLMPath and Secure should point to the machine where CA SSO Web Access Control is installed.
   - The ServerName attribute should point to the IP Address of the machine where the CA SSO Policy Server is installed.

     Note: For more information about configuring the WAC Web Agent, see the WAC documentation.

CA SSO Policy Manager Verification Steps

1. Ensure that the SiteMinder and CA SSO Policy Servers to use the same user or authentication store.
2. Make sure you have the following:

   - An SSO administrator name and password. The SiteMinder Policy Server uses the administrator name and password when authenticating to the CA SSO Policy Server through the smauthetsso authentication scheme.
   - The SSO ticket encryption key, as it is needed by the SiteMinder Policy Server's smetssocookie active response.

     Note: For more information about configuring the Policy Manager, see the CA SSO documentation.

SiteMinder Policy Server Configuration Steps

1. Create a Web Agent, Agent Configuration Object, and Host Configuration Object using the Administrative UI. For more information, see the *Policy Server Installation Guide* and the *Web Agent Installation Guide*.
2. Configure the SiteMinder and CA SSO Policy Servers to use the same user or authentication store.

   For SiteMinder user store configuration instructions, see the User Directories chapter in this guide.

   For the CA SSO authentication store, see the CA SSO documentation.
3. Configure an smetssocookie (certificate) custom active response.
4. Create a domain, realm, and rules using the Administrative UI to protect any resource with the SiteMinder Web Agent.

   Note: When creating the rules, append the smetssocookie custom active response to them.

Overall Verification Steps

1. Configure the user with credentials to access resources protected by the SiteMinder Web Agent and the WAC Web Agent.
2. Restart the SiteMinder Policy Server and Web server hosting the Administrative UI.
3. Access the resource protected by the SiteMinder Web Agent and provide this Web Agent with the appropriate user credentials.
4. After gaining access to this resource, in the same browser session, request a resource protected by the WAC Web Agent.

   You should gain access to this resource without being prompted for credentials.

More information:

Domains (see page 493)
Realms (see page 501)
Rules (see page 513)

Configure Single Sign-On from CA SSO Client to SiteMinder

SiteMinder provides single sign-on from the CA SSO Client to SiteMinder.

To enable single sign-on from an CA SSO Client to SiteMinder:

SiteMinder Policy Server Configuration Steps

1. Configure the smauthetsso custom authentication scheme using the Administrative UI.
2. Create a domain, realm, and rules using the Administrative UI to protect any resource with the SiteMinder Web Agent.
3. Configure the smauthetsso custom authentication scheme to protect a resource.
4. Create a policy that grants access to the protected resource to users who already have access the browser protected by the CA SSO Client.

CA SSO Client Verification Steps

Set the following in the CA SSO Client SsoClnt.ini file:

Note: The SsoClnt.ini file is installed in C:\Program Files\CA\CA SSO\Client on the CA SSO Client machine. More information on configuring the CA SSO Client exists in the CA SSO documentation.

DomainNameServer=<*eSSO_WA_FQDN*> <*SM_WA_FQDN*>

eSSO_WA_FQDN

(Optional) Specifies the fully qualified domain name for the WAC Web Agent

SM_WA_FQDN

Specifies the fully qualified name for the SiteMinder Web Agent

Overall Verification Steps

1. Restart the CA SSO Client, SiteMinder Policy Server, and Web server hosting the Administrative UI.
2. Access the protected browser through the SSO Client and enter the URL of the resource protected by the SiteMinder Policy Server.

   You should be able to access the resource without being rechallenged by SiteMinder.

More information:

Domains (see page 493)
Realms (see page 501)
Rules (see page 513)

Configure Single Sign-On from CA SSO to SiteMinder

SiteMinder provides single sign-on from CA SSO to SiteMinder.

To enable single sign-on from CA SSO to SiteMinder

SiteMinder Policy Server Configuration Steps

1. Configure the smauthetsso custom authentication scheme using the Administrative UI.
2. Create a domain, realm, and rules using the Administrative UI to protect any resource with the SiteMinder Web Agent.

   For more information, see Domains, Grouping Resources in Realms, or Rules.
3. Configure the smauthetsso custom authentication scheme to protect a resource.

WAC Web Agent Verification Steps

1. Configure the domain in the WAC Web Agent's webagent.ini file by setting DomainCookie=<domain>.

   Note: The value you specify for the domain must be the same for the CA SSO and SiteMinder Web Agents. The file is installed on the WAC Web Agent machine at C:\Program Files\CA\WebAccessControl\WebAgent\webagent.ini
2. Verify the following Web server and the authentication method settings in the webagent.ini file:
   - The "Authentication methods" and "The default authentication method" parameters should be configured as SSO.
   - The WebServerName, PrimaryWebServerName, AgentName, NTLMPath and Secure should point to the machine where SSO Web Access Control is installed.
   - The ServerName attribute should point to the IP Address of the machine where the CA SSO Policy Server is installed.
   - For more information about configuring the WAC Web Agent, see the CA SSO documentation.

   Note: For more information about configuring the WAC Web Agent, see the WAC documentation.

SiteMinder Web Agent or CA SiteMinder SPS Configuration Steps:

1. Enable the SSO plug-in installed with the Web Agent or CA SiteMinder SPS, so that SSO Client cookies can be authenticated, by removing the comment character (#) from the following line in the WebAgent.conf file:

   #LoadPlugin=*path_to_eTSSOPlugin.dll* | *path_to_libetssoplugin.so*

   Note: The WebAgent.conf file is located as follows:

   r12.5 IIS 6.0 or Apache 2.0 Web Agent

   See the Web Agent Configuration Guide.

   6.0 CA SiteMinder SPS

   SPS_install_dir\proxy-engine\conf\defaultagent\

   SPS_install_dir

   CA SiteMinder SPS installation directory

2. Restart the Policy Server.

Overall Verification Steps

1. Restart the WAC Web Agent, SiteMinder Policy Server, and Web server hosting the Administrative UI.
2. Access a resource protected by the WAC Web Agent and provide valid credentials.
3. Access a resource protected by the SiteMinder Web Agent in the same browser.

   You should be able to access the resource without being rechallenged by SiteMinder.

Configure an smetssocookie Web Agent Active Response Attribute

The smetssocookie Web Agent active response generates and sends an SSO cookie to a Web browser. The SSO cookie lets a SiteMinder-authenticated user access WAC or CA SSO protected content without having to reauthenticate.

To configure an smetssocookie Web Agent response attribute

1. Click Policies, Domain.
2. Click Responses.

   The Responses page appears.

3. Click Create Response.

   The Create Response: Select Domain page appears.

4. Select a domain from the list and click Next.

   The Create Response: Define Response page appears.

5. Define a Name and Description for the response.
6. Verify that the SiteMinder is selected and that Web Agent appears in the Agent Type list.
7. Click Create Response Attribute.

   The Create Response Attribute page appears.

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

   The Create Response Attribute: *Name* page appears.

9. Select WebAgent-HTTP-Cookie-Variable from the Attribute list.
10. Select Active Response in Attribute Kind.

    Additional fields appear in Attribute Fields.

11. In the Cookie Name field, type **SSOTK**.
12. In the Library Name field, type **smetssocookie**.
13. In the Function Name field, type **GenEtssoCookie**.

    Note: The function name is case-sensitive.

14. In the Parameters field, define the following ordered set of tokens :

    <*CA_PS_Host_Name*>;<*SSO_Auth_Host*>;<*SSO_AuthMethod*>;<*EncryptionKey*>

    CA_PS_Host_Name

    Specifies the host name of the CA SSO Policy Server.

    SSO_Auth_Host

    Specifies the SSO authentication host name in the CA Policy Manager. You can specify this host name by going to Web Access Control Resources, Configuration Resources, Authentication Host.

    Required value: SSO_Authhost

    SSO_AuthMethod

    Defines the SSO authentication method.

    Required value: SSO

    EncryptionKey

    Defines the ticket encryption key for the SSO authentication host name in the CA Policy Manager.

    The cookie script appears in the Script field.

    Note: To improve legibility, you can type a space before and after any token.

15. Click Submit.

    The Create Response Attribute task is submitted for processing, and the Create Response: Define Response page re-appears.

16. Click Finish.

    The Create Response task is submitted for processing. When the task is complete, the response can be added to an OnAuthAccept rule.

Configure an smauthetsso Custom Authentication Scheme

The CA SSO SiteMinder (smauthetsso) authentication scheme lets the SiteMinder Policy Server validate CA SSO authentication credentials so that a user already authenticated in a CA SSO/WAC environment does not need to re-authenticate to SiteMinder. This custom authentication scheme accepts a CA SSO Cookie as a login credential; has it validated by a CA SSO Policy Server; extracts the user name from it; and verifies that the name is present in the SiteMinder user store. You can set this authentication scheme in a cookie, cookieorbasic, or cookieorforms mode.

You can configure one CA SSO Policy Server to failover to another CA SSO Policy Server when it fails for some reason. To configure failover, specify a comma-separated list of CA SSO Policy Servers as parameter field in Scheme Setup on the Authentication Scheme page.

Note: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

Follow these steps:

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.

3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.

4. Click OK

   The Create Authentication Scheme page appears.

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

5. Select Custom Template from the Authentication Scheme Type list.

   Scheme-specific fields and controls open.

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

6. Enter **smauthetsso** in the Library field.
7. Enter and confirm the password of the CA SSO Policy Server administrator in the Secret and Confirm Secret fields.
8. Define an ordered set of tokens in the Parameter field with the following format:

   Mode [; <*Target*>] ; *AdminID* ; *CAPS_Host* ; *FIPS_Mode ; Identity_File*

   Note: Separate tokens with semicolons. You may enter a space before and after each token for improved legibility.

   Example: cookie ; SMPS_sso ; myserver.myco.com ; 0 ; /certificates/def_root.pem

   Example: cookieorforms ; /siteminderagent/forms/login.fcc ; SMPS_sso ; myserver.myco.com ; 1 ; /certificates/def_root.pem

   Mode

   Specifies the type of credentials the authentication scheme accepts. Accepted values include cookie, cookieorbasic, or cookieorforms.

   cookie

   Specifies that only CA SSO cookies are acceptable.

   cookieorbasic

   Specifies that a basic authentication scheme is used to determine the login name and password if a CA SSO cookie is not provided.

   cookieorforms

   Specifies that a forms authentication scheme is used to determine the login name and password if a CA SSO cookie is not provided.

   Target

   Specifies the pathname of the .fcc file used by the HTML Forms authentication scheme.

   Note: This value is only required for the cookieorforms mode.

   AdminID

   Specifies the user name of the CA SSO Policy Server administrator for the CA SSO Policy Server. SiteMinder uses the administrator's user name and password to request validation of CA SSO cookies when authenticating to the CA SSO Policy Server.

   CAPS_Host

   Specifies the name of the host where the CA SSO Policy Server resides.

   FIPS_Mode

   Specifies the FIPS mode of operation in which the Policy Server is operating. Zero (0) specifies non-FIPS mode. One (1) specifies FIPS mode.

Identity_File

Specifies the path to the CA SSO identity file. The Policy Server uses this file to communicate with the CA SSO Policy Server.

9. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

More information:

HTML Forms Authentication Schemes (see page 315)

Chapter 27: CA User Activity Reporting Module Integration

CA User Activity Reporting Module (CA UAR) provides SiteMinder connector guides, which detail how to configure a CA UAR integration with SiteMinder. The guide you use depends on whether SiteMinder is configured to store audit information in a text file (smaccess.log) or an ODBC database.

To locate the CA UAR connector guides

1. Go to the CA User Activity Reporting Module Integration Matrix.
2. Click Authentication Service that is located under Product Integrations.

   The SiteMinder connector guides are based on the type of logsensor that CA UAR is to use.

3. Do *one* of the following:
   - If SiteMinder stores audit information in a text file, use the connector guide for a File logsensor.
   - If SiteMinder stores audit information in an ODBC database, use the connector guide for an ODBC logsensor.

Each of these guides is also available from the CA UAR Administrative UI when you create the required connector. To access these guides when creating the connector, click Help.

Appendix D: Using the Policy Server as a RADIUS Server

This section contains the following topics:

Use the Policy Server as a Radius Server

Remote Authentication Dial-In User Service (RADIUS) is a protocol that enables you to exchange session authentication and configuration information between a Network Access Server (NAS) device and a RADIUS authentication server. You can use the Policy Server as the RADIUS authentication server.

The RADIUS protocol is often used by NAS devices that serve as:

- Proxy services for Internet Service Providers (ISP)
- Firewalls
- Corporate dial-up security services

The RADIUS Client/Server Architecture

RADIUS is designed to simplify security by separating the communication technology provided by a NAS device from the security technology provided by the authentication server. RADIUS security protects remote access to networks and network services using a distributed client/server architecture. The Policy Server is the RADIUS server. The RADIUS client is the NAS device.

A NAS device performs one of the following:

- Supports dial-in protocols, such as SLIP or PPP, authenticates users by using the RADIUS authentication server, and routes the user onto the network; or
- Supports direct connections to the network through a firewall, authenticates users by using the RADIUS authentication server, and grants network access.

The Policy Server can serve as the RADIUS authentication server when configured as described in this chapter. As the RADIUS server, the Policy Server authenticates RADIUS users using a RADIUS authentication scheme and a pre-defined user directory.

Note: To use RADIUS accounting, you must configure a separate RADIUS accounting server. The Policy Server will satisfy the NAS device by sending the ACK response to the accounting server. However, you can log accounting information to files.

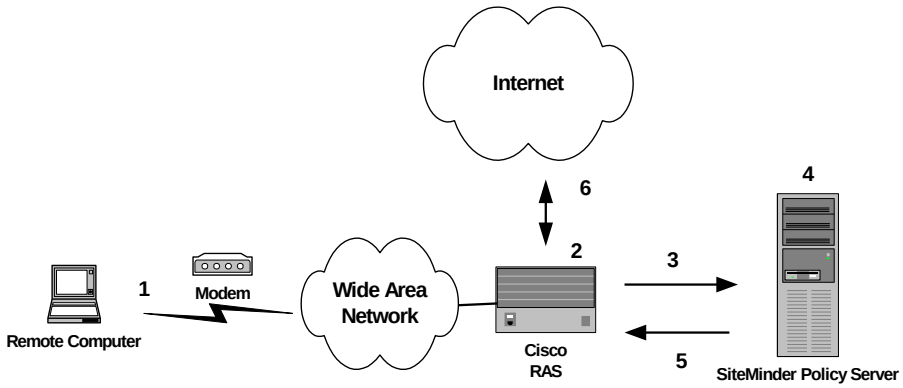
More information:

Generate RADIUS Logs for Accounting and Debugging (see page 764)

How RADIUS Authentication Works with the Policy Server

The Policy Server authenticates users through a series of communications with the NAS device. When SiteMinder authenticates a user, the NAS provides that user with access to the appropriate network services.

This authentication process is depicted in the following graphic:



1. A user dialing in from a modem attempts to open a connection to the Cisco RAS (a NAS device), which will enable the user to access the Internet.
2. The RAS determines that it must use a RADIUS user profile to authenticate the user.
3. The RAS sends the user connection request to the Policy Server.
4. The Policy Server obtains the user's name and password using one of the following methods:
   - Authenticates using Password Authentication Protocol (PAP)

     PAP is a PPP authentication protocol that provides a simple method for a host to establish its identity in a two-way handshake. Authentication takes place only upon initial link establishment and does not use encryption.
   - Authenticates using Challenge Handshake Authentication Protocol (CHAP)

     CHAP is also a secure PPP authentication protocol. CHAP provides a way to periodically verify the identity of a host using a three-way handshake and encryption. Authentication takes place upon initial link establishment. The RAS can repeat the authentication process any time after the connection takes place.
   - Authenticates using Security Dynamics ACE/Server or Secure Computing SafeWord server.
5. The Policy Server sends an authentication response to the RAS.
6. One of the following takes place:
   - If authentication is unsuccessful, the RAS refuses the connection.
   - If authentication is successful, the RAS receives a list of attributes from the SiteMinder response that fires upon authentication of the user. The attributes are used as a user profile, which configures the user's network session.

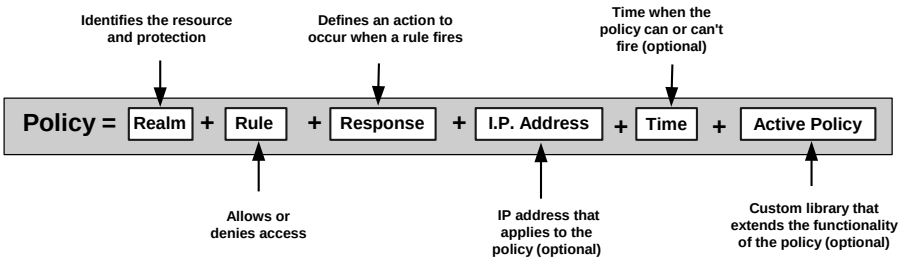
     The RAS notifies the Policy Server that the session has begun and when the session ends.

Policies in RADIUS Environments

A SiteMinder RADIUS policy is enforced by a RADIUS Agent and is created by binding the following elements together:

- An authentication rule
- A response
- A user or user group, and
- Optionally, an IP address, Time, and an active policy.

The basic structure of a policy is shown in the following diagram.

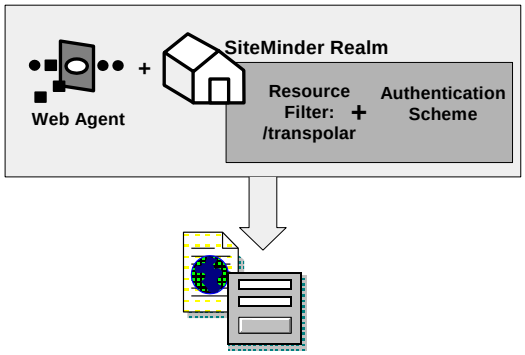


Although RADIUS policies are composed of the same elements that are contained in policies used by SiteMinder Agents, RADIUS Agents interpret the components differently. Rules, realms, and responses perform different functions, as shown in the following table.

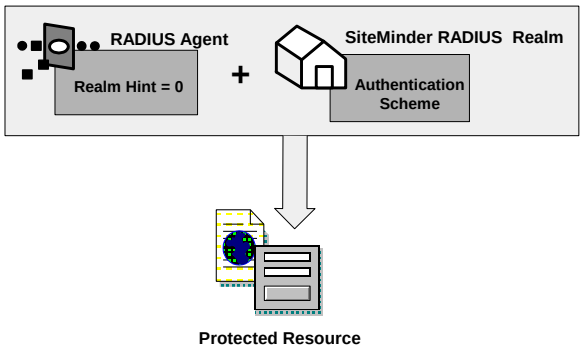
Policy Component	In a RADIUS Policy, this item:	In a SiteMinder Agent Policy, this item:
Realm	■ Identifies the Agent. ■ Identifies the authentication scheme. ■ Defines session timeouts.	■ Defines the resource filter (directory within the domain that the SiteMinder Agent will govern). ■ Identifies the Agent. ■ Identifies the authentication scheme. ■ Defines the state (protected or unprotected) of the resource. ■ Identifies which events (authentication or authorization) to process. ■ Defines session timeouts.
Rule	■ Authenticates only. ■ Allows or denies access. ■ Defines time or active rule restrictions.	■ Defines the resource filter. ■ Defines the action (Web Agent action, authorization event, or authentication event. ■ Allows or denies access. ■ Authorizes and authenticates. ■ Defines time or active rule restrictions.
Response	■ Defines the values to return for authentication events.	■ Defines the value to return for an authorization event. ■ Defines the values to return for authentication events. ■ Defines the values to return for authorization reject events. ■ Defines the values to return for authentication reject events.

RADIUS vs. Non-RADIUS Resources

The elements of a RADIUS policy are treated differently in part because of how resources are identified in a RADIUS environment. In a SiteMinder Agent environment, specific resources are identified using a resource filter in the definition of the realm. The resource filter identifies the directory location of the resources. The realm definition also identifies the Web Agent and the authentication scheme, as shown in the following diagram:



As shown in the following diagram, protected resources are located differently in a RADIUS environment. Instead of the realm identifying the resource using a filter, the RADIUS Agent identifies the resource using a *realm hint*. A realm hint is an attribute that enables the Policy Server to establish the domain in which to authenticate users. The realm hint either identifies a specific realm that the Agent protects or signifies that the Agent must protect the entire NAS device.



Use Realm Hints

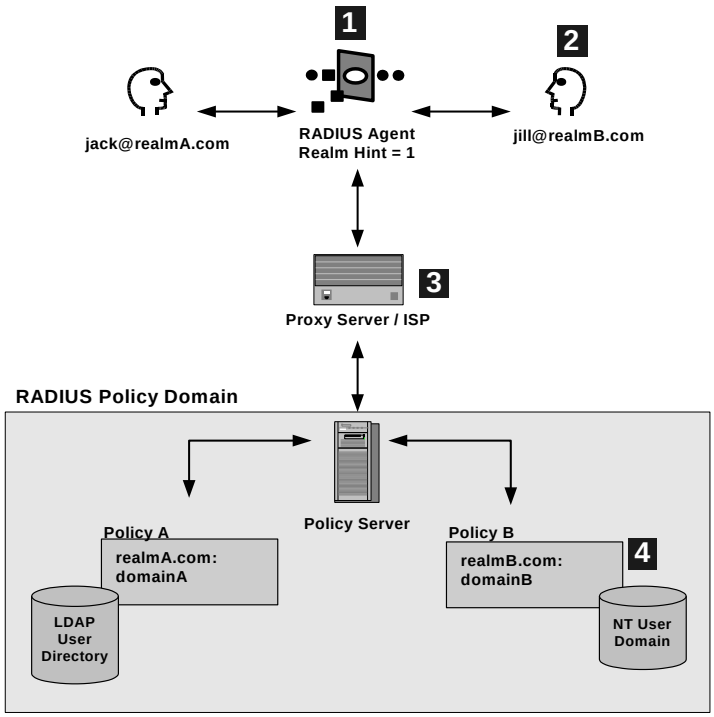
How does a RADIUS Agent protect a NAS device that must authenticate users in different domains, such as domainA and domainB? A realm hint is a RADIUS attribute that enables SiteMinder to determine the correct domain in which to authenticate a user. You must provide a RADIUS Agent with one of the following realm hint values:

- 0--(Default) Signifies that there is only one realm in the policy domain and therefore, a hint is not needed. The realm is bound to the NAS device directly.
- 1--(RADIUS User-Name attribute) SiteMinder parses the realm name from the user name in this attribute, then finds the associated domain, as explained below.
- An attribute that contains the actual name of the domain. This attribute is not available for all NAS devices. see your NAS device product documentation for more information.

When the realm hint is set to 1, the realm name is parsed from the user name attribute. The user_name-realm separator must be "@" or "/".

- If the separator is "@" then the element following the "@" is the realm name. For example, in jack@realmA.com, the realm is realmA.com.
- If the separator is "/" then the element preceding the "/" is the realm name. For example, in x5/jack, the realm is x5.

The following diagram and explanation shows how a proxy server determines the correct SiteMinder domain in which to authenticate a user.



1. One RADIUS agent protects both SiteMinder domains. The RADIUS Agent is configured with the realm hint value of 1.
2. When Jill tries to access the ISP's proxy server, the RADIUS agent intercepts the request and forwards Jill's user name attribute jill@realmB.com to the Policy Server.
3. The Policy Server parses the user_name and realm_name from the user name attribute.

   Example: jill@realmB.com, where jill is the user_name and realmB.com is the realm_name.

   The Policy Server identifies the domain associated with the realm_name. The domain associated with realmB.com is domainB.
4. The Policy Server authenticates the user_name in the appropriate directory. The user_name jill is authenticated in the NT user domain defined for Policy B: realmB.com:domainB.

Responses in RADIUS Policy Domains

SiteMinder responses can be used to return RADIUS attributes to the NAS device if the user is authenticated. Attributes configure the characteristics of the session once the user is authenticated and define the user profile of the authenticated user. The user profile can be used by the NAS device. For example, using attributes in a response, you can define time limits for the RADIUS user session.

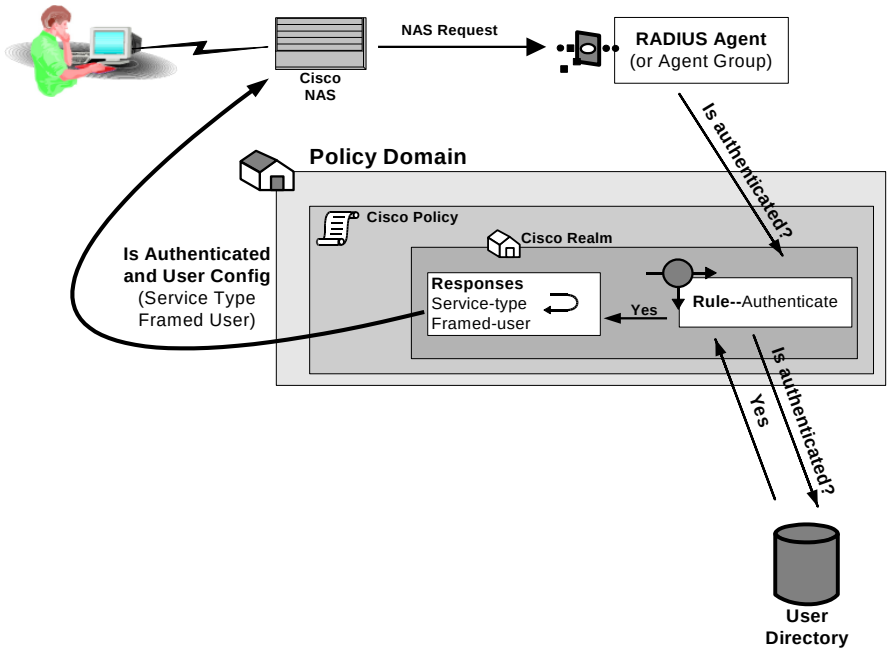
Using responses, you can provide the NAS device with user profile information that assigns privileges to the user. For example, you could allow one user unlimited access to a resource, yet limit another user's access to the same resource. Used in this way, responses give you the ability to authorize users even though RADIUS is primarily only a mechanism for authentication.

Note: If the NAS specifies authentication only, by default, SiteMinder does not return RADIUS attributes. To return RADIUS attributes when the NAS specifies authentication only, follow the instructions in Configure SiteMinder to Always Return RADIUS Attributes (see page 745).

How Responses Work

RADIUS responses are paired with rules that authenticate. If a rule authenticates a user successfully, the RADIUS response is triggered. If the rule does not authenticate the user, the response is not triggered.

If a response is triggered, the Policy Server sends the attributes contained in the response to the NAS device. This information is used to customize the user's session, as shown in the following diagram:



Attribute Types

You can use the following attributes in responses:

- User attributes
- DN attributes
- Active response attributes
- Radius attributes

User Attributes

These attributes return information associated with a user in an LDAP, WinNT, or ODBC user directory. User attributes are retrieved from the user directory and can used to modify the behavior of the RADIUS device.

DN Attributes

These attributes return profile information associated with an LDAP directory object related to the user. For example, the DN attribute could return information about LDAP objects such as the user's group or organizational unit (OU).

Active Response Attributes

These attributes return values from a custom library that was developed using the SiteMinder Authorization API. An active response is generated when SiteMinder invokes a function in the custom library.

RADIUS Attributes

These attributes return values defined by the following Agent type attributes:

RADIUS

Generic RADIUS attributes, as defined by the RADIUS Protocol specification, *Request for Comment (RFC) 2138.* The identifiers for these attributes include 1-25 and 27-63. Some of these attributes may be used multiple times in the same response.

Any RADIUS Agent type can return a response that includes generic RADIUS attributes.

RADIUS Extended

Attributes defined in the Dictionary file of the NAS device. These attributes define values that are not defined by generic RADIUS attributes and are specific to the type of NAS device in use. The unique identifiers for these attributes extend beyond the range reserved for generic RADIUS attributes, starting with 64. For example, Lucent provides an extended RADIUS attribute called *Ascend-Disconnect-Cause,* which uses the identifier 195.

Only Agent types that match the vendor type of the extended RADIUS attribute can use the attribute. For example, a Shiva Agent type can use the extended RADIUS attributes defined for Shiva, but a Cisco Agent type cannot use Shiva extended attributes in a response. The extended attributes that are used in a response must match the attributes defined in the Dictionary file of the RADIUS client.

By default, SiteMinder provides pre-defined RADIUS extended attributes for some Agent Types that use these attributes, such as Ascend (Lucent). You can also define additional RADIUS extended attributes for any of the RADIUS Agent types, if necessary.

Vendor-Specific

Attributes defined in the Dictionary file of the NAS device, which use 26 as an identifier. Vendor-specific attributes enable you to define attributes for values that are not provided by the generic RADIUS attributes. Some vendors use vendor-specific attributes in place of or in addition to RADIUS extended attributes. For example, Cisco does not use RADIUS Extended attributes; however, this NAS device supports several vendor-specific attributes, such as *Cisco AV–pair* and *Account-Info*.

You can use vendor-specific attributes to pass information to other protocols. For example, you can define a vendor specific attribute for the Cisco AV–pair attribute to pass TACACS+ information to a TACACS+ server.

Vendor-specific attributes can only be defined in responses that match the vendor type of the RADIUS client.

By default, SiteMinder provides pre-defined vendor-specific attributes for some Agent Types that use these attributes, such as the Network Associates' Sniffer Agent type. You can also define additional RADIUS extended attributes to any of the RADIUS Agent types, if necessary.

Note: For more information about RADIUS attributes, see *Request for Comment (RFC) RADIUS Protocol 2138.*

More information:

Create Attributes for Agent Types (see page 746)

Configure SiteMinder to Always Return RADIUS Attributes

Some NAS devices always expect RADIUS responses in the Access-Accept, even if the NAS specifies authentication only. If the NAS specifies authentication only, by default, SiteMinder does not return RADIUS attributes.

To always return RADIUS attributes to a NAS device, create a new registry value with the following parameters:

- Value type--DWORD
- Value Name--HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Authentication\AlwaysReturnRadiusAttrs
- Value Data--A numeric value greater than zero

Note: The install program does not create a registry entry for AlwaysReturnRadiusAttrs. Until you create and set the entry, SiteMinder uses the default value of 0.

After you set AlwaysReturnRadiusAttrs to a value greater than zero, the following message will appear in the Authentication Server's debug log:

> Radius Attributes will be returned regardless of RA_SERVICE_TYPE_AUTHENTICATE_ONLY

Create Attributes for Agent Types

Before you can use an attribute in a response, the attribute must be made available to the Agent type returning the response. Attributes are made available to Agent types by defining the attributes in Agent types. Although many Agent types are pre-configured with vendor-specific and RADIUS extended attributes, you can add additional extended RADIUS, generic RADIUS, and vendor-specific attributes to Agent types, as needed.

Define Multiple Instances of an Attribute

You can define multiple instances of a vendor-specific attribute for the same Agent type. When you define multiple instances of a vendor-specific attribute, you can send a different value to the NAS device for each instance of the attribute. For example, for a Cisco Agent, you could define the following vendor-specific attributes, all using the same identifier (26):

- Cisco-AVpair
- Account-Info
- Command-Code

The settings that define the number of times an attribute can be used within a response are located on the Modify Agent Type Attribute pane of the Administrative UI.

To configure the attribute to be used multiple times, the Access Accept value must be set to Zero or Many.

The type of attribute that you define must match the vendor type of the Agent returning the response. For example, a vendor-specific Cisco attribute can only be returned by a Cisco Agent.

When the response is returned by the Agent, the packet structure of the response reflects the type of RADIUS Agent that sent the response. For example, the packet structure of a response returned by a Cisco Agent would include the vendor ID and the length of the string.

To define an attribute for an Agent type

1. Log into the Administrative UI.
2. Click Infrastructure, Agent Type, Modify Agent Type.

   The Modify Agent Type pane opens.

3. Specify search criteria, and click Search.

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

4. Select an Agent type, and click Select.

   The Modify Agent Type: *Name* pane opens.

5. Click Create Agent Type Attribute on the Agent Type Attributes group box.

   The Create Agent Type Attribute pane opens.

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

   The Create Agent Type Attribute: *Name* pane opens.

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

8. Select RADIUS, RADIUS Extended, or Vendor Specific from the RADIUS Type list.

9. Select the type of data that the attribute contains from the Data Type list.

10. Type one of the following attribute identifiers in the Identifier field:

    - **Generic RADIUS**

      The attribute identifier is defined in the RADIUS protocol specification. Although it is possible to overwrite the identifier of a Generic RADIUS attribute, you should generally retain the pre-defined Generic RADIUS attribute definitions, which match the RADIUS specification (*RFC 2138*).

      Example: To create an attribute for the Callback-Id variable, type 20 in the Identifier field.

    - **RADIUS Extended**

      The attribute identifier is defined in the vendor documentation.

      Example: To create an attribute for the Ascend-Callback attribute, type 246 in the Identifier field.

    - **Vendor Specific**

      The attribute identifier is 26.

      Example: To create an attribute for a Cisco Agent that enables the Agent to use TACACS+, type 26 in the Identifier field.

    Note: For more information about attribute identifiers, see your RADIUS vendor documentation.

11. Select a RADIUS code for each field on the RADIUS Behavior group box. The RADIUS codes are:

    Not allowed

    Attribute cannot be used in a response.

    Zero or One

    One instance or no instances of the attribute can be returned in the same response. If this value is selected, and you use the attribute in a response, the attribute will be removed from the Attribute list after you have used the attribute in a response.

    Zero or Many

    Multiple instances or no instances of the attribute can be returned in the same response.

    One and Only One

    One instance of the attribute must be returned in a response. If this value is selected, and you use the attribute in a response, the attribute will be removed from the Attribute list after you have used the attribute in a response.

    The fields on the RADIUS group box are:

    Access Request

    Provides information used to determine whether or not a user is allowed access to a specific NAS. The Access Request packets also provide information for any special services requested for that user.

    Access Accept

    Provides specific configuration information necessary to begin delivery of service to the user.

    Note: You must set the Access Accept value to Zero or One, Zero or Many, or One and Only One in order to use the attribute in a response.

    Access Reject

    Sends information if any value of the received Attributes is not acceptable. This code is often used for reply messages.

    Access Challenge

    Sends information if the NAS device has been configured for challenge/response.

Accounting Request

Describes the type of service being delivered and the user to whom it is being delivered.

Accounting Response

Sends information if the Accounting Request was recorded successfully. A RADIUS Accounting-Response is not required to have any attributes in it.

12. If the data type is number, click Create on the Values group box.
13. Type the symbolic name of the attribute in the Symbolic Name field, type the actual numeric value of the attribute in the Numeric Value field, and click OK.

    The Modify Agent Type Attribute pane reopens, and the attribute name-value pair is added to the Values group box.

    Note: To create multiple attribute name-value pairs, repeat steps 12 and 13. By mapping symbolic names to values, you only need to remember names.

14. Click Submit.

    The Modify Agent Type pane reopens, and the Agent type attribute is added to the Agent Type Attributes group box.

15. Click Submit.

    The Modify Agent Type task is submitted for processing.

    Note: When the task is complete and you create a response for this Agent type, you can select the Agent type attribute that you just added to the Agent type from an attribute list.

Modify Existing Attributes

You can modify attributes that you created and attributes that have been pre-defined for a RADIUS Agent. For example, you can modify the pre-defined Ascend-PPP-Address attribute for the Ascend Agent type.

Note: When you modify an existing attribute, the attribute is not updated dynamically in responses that already use the attribute. If an attribute is used in a response, you must recreate the response using the updated attribute.

All RADIUS Agent types have been pre-configured to use the generic RADIUS attributes, as defined in *RFC 2138*. These attributes are available to be used by each RADIUS Agent type.

Important! If you overwrite a generic attribute or define a new attribute in the Generic RADIUS Agent, the change is applied to *all* RADIUS Agents. For example, if you modify the Filter ID attribute in the Generic RADIUS Agent, the modification is also made to all of the other RADIUS Agent types, such as Cisco, Shiva, Livingston, Ascend, and Checkpoint.

To modify agent type attributes

1. Log into the Administrative UI.
2. Select Agents from the Infrastructure tab.
3. Click Modify Agent Type.
4. Click Search.
5. Select an Agent type and click Select.

   The Modify Agent Type pane opens.
6. Modify the Agent Type values by clicking the Edit button on the left of the attribute
7. Click Submit to save the changes.

More Information:

Define Multiple Instances of an Attribute (see page 746)

Deploy SiteMinder in a RADIUS Environment

SiteMinder can be setup to provide authentication services in a variety of different RADIUS environments:

- A homogeneous environment composed of only one NAS device, such as a Cisco RAS, and only one user directory. This environment is discussed in How to Authenticate Users in a Homogeneous RADIUS Environment.
- A heterogeneous environment composed of multiple NAS devices, such as a Checkpoint firewall and a Cisco RAS, and one user directory. This environment is discussed in Authenticate Users in Heterogeneous RADIUS Environments with One User Directory (see page 753).
- A heterogeneous environment composed of multiple NAS devices, such as a Checkpoint firewall and a Cisco RAS, and multiple user directories. This environment is discussed in Authenticating Users in Heterogeneous RADIUS Environments with Two User Directories (see page 757).

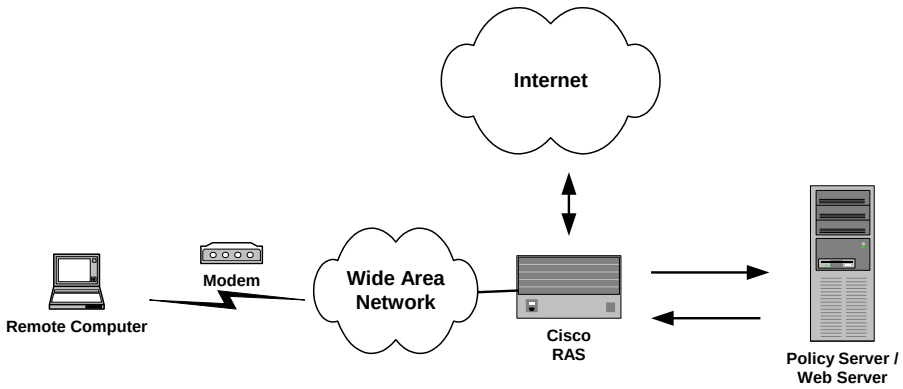
Guidelines for Protecting RADIUS Devices

Before deploying SiteMinder in a RADIUS environment, note the following guidelines:

- Realm names in the same policy domain must be unique.
- Only one type of RAS device can be protected within one policy. A single policy cannot protect more than one RADIUS device because each vendor uses a separate Dictionary file. The responses in a single policy must interpret return attributes identically. If the environment is heterogeneous and includes a variety of RAS devices, define a separate policy for each type of RADIUS device.
- Multiple user directories can be defined within one policy domain. When multiple user directories are defined, specify a search order.
- You can combine RADIUS Agents for different NAS vendors in a single generic RADIUS Agent group, and then use the same Agent group in a separate policy for each type of RADIUS Agent. For example, if the Agent group contained a Shiva Agent and a Cisco Agent, you would create a Shiva policy and a Cisco policy. The same rule and realm would be added to each policy, which saves time. However the response associated to each instance of the same rule would differ; the Cisco policy would associate a Cisco response to the generic rule and the Shiva policy would associate a Shiva response to the generic rule.

How to Authenticate Users in a Homogeneous RADIUS Environment

A homogeneous RADIUS environment is the most simple to protect. You can protect the RADIUS device using just one policy. This type of environment includes only one RADIUS device, such as a Cisco RAS, and one user directory, as shown in the following graphic:



Follow these steps:

1. Configure the system:
   a. Define the RADIUS Agent, as explained in Configure a RADIUS Agent.
   b. Setup a user directory against which to authenticate RADIUS users, as explained in Set Up the User Directory
   c. Optionally, you can also define administrative users and modify the authentication schemes.
2. Configure the policy domain:
   a. Create a RADIUS authentication scheme (CHAP or PAP), as explained in Create the Authentication Scheme.
   b. Define a realm that identifies the RADIUS Agent and the RADIUS authentication scheme, as explained in Configure a Realm Protected by a RADIUS Agent.
   c. Define a rule that enables authenticated users to access the realm protected by the RADIUS Agent, as explained in Configure a Rule for Authentication Event Actions.
   d. Define a response that provides the user profile to the NAS device and configures the characteristics of the session using response attributes, as explained in Configure a Response and RADIUS Agent Response Attributes (see page 541).
   e. Create a policy that binds the rule and response with the user directory, as explained in Configure a Policy (see page 565).

More Information:

How RADIUS Authentication Works with the Policy Server (see page 736)

Set Up the User Directory

You can authenticate RADIUS users using any user directory that is supported for the NT or UNIX platform you are using.

If the user directory contains information about user privileges, you can create responses using user attributes. When the user attributes are sent back to the RADIUS device, the attributes are used to configure the user session.

You can use the following directories:

- ODBC-enabled database
- NT Domain
- Netscape or NDS LDAP

Set Up the Policy Domain

The policy domain must identify one or more user directories that contain the names of the RADIUS users, the names of the Administrators who can modify the domain, and the realm that the RADIUS Agent is protecting.

Create the Authentication Scheme

You can use any of the following authentication schemes:

- Password Authentication Protocol (PAP)

  PAP is a PPP authentication protocol that provides a simple method for a host to establish its identity in a two-way handshake. Authentication takes place only upon initial link establishment and does not use encryption.

- Challenge Handshake Authentication Protocol (CHAP)

  CHAP is also a secure PPP authentication protocol. CHAP provides a way to periodically verify the identity of a host using a three-way handshake and encryption. Authentication takes place upon initial link establishment. The RAS can repeat the authentication process any time after the connection takes place.

- Security Dynamics ACE/Server or Secure Computing SafeWord server.

Note: For more information on creating an authentication scheme, see the Authentication Schemes chapter in this guide.

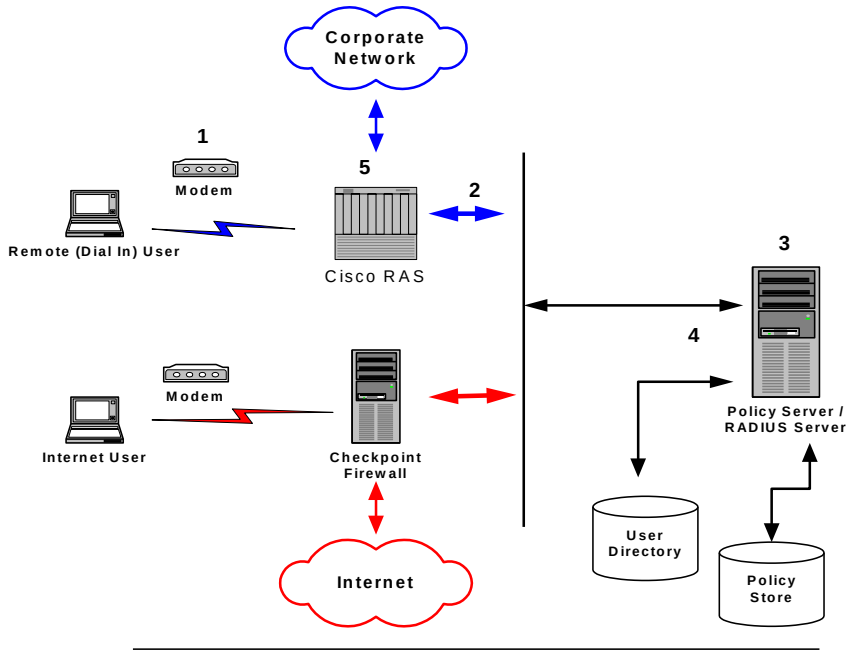
Authenticate Users in Heterogeneous RADIUS Environments with One User Directory

A more powerful and complex deployment of the Policy Server in a RADIUS environment is one that includes multiple realms administered by multiple NAS devices. In this scenario, the Policy Server can serve as the RADIUS authentication server for multiple RADIUS clients at once.

The advantage of using a heterogeneous configuration is that you save time by using the same RADIUS authentication server (that is, the Policy Server) for each RADIUS client.

How Users are Authenticated in Heterogeneous, Single Directory Environments

An example of a heterogeneous configuration is illustrated in the following graphic:



In the network topology shown in the previous diagram, the Policy Server authenticates users of two NAS devices: a Cisco RAS and a Checkpoint Firewall. The Policy Server uses one user directory to authenticate the users.

Each NAS device has its own RADIUS Agent, which has been configured with a realm hint. When the Policy Server receives a request to authenticate the user, it uses the RADIUS Agent's realm hint to determine the resource (domain) that the authenticated user can access.

The process of authentication when one user directory is used is as follows:

1. The remote user dials in from a modem and the Cisco RAS determines that it must use a RADIUS user profile to authenticate the user.
2. The RAS sends the user connection request to the Policy Server.
3. The Policy Server enacts the policy defined for the RAS, and the RADIUS Agent associated with the Cisco RAS does the following:
    a. Determines the user's domain using a realm hint.
    b. Obtains the user's name and password using the authentication scheme configured for the Agent.

4. The Policy Server evaluates the user information against the user directory and policy store.
5. The Policy Server sends an authentication response to the Cisco RAS and one of the following takes place:
   - If authentication is unsuccessful, the RAS refuses the connection.
   - If authentication is successful, the RAS receives a list of attributes from the user profile in the RADIUS server's database and establishes network access for the caller.

     The RAS notifies the Policy Server that the session has begun and when the session ends.

When the Internet user attempts to dial into the Internet Service Provider via the Checkpoint Firewall, a similar process of authentication occurs. Using the realm hint, the RADIUS Agent defined for the Checkpoint Firewall determines which domain the Internet user has access to. If the user is authenticated, the Policy Server passes the Firewall the correct attributes to establish the session.

User information for both NAS devices is stored in the same user directory. Each time the Policy Server receives an authentication request, it authenticates the user using the same data directory.

System and Policy Domain Configuration

This system configuration differs from the homogeneous environment; you must now create two Agents.

Within the policy domain there is one policy that includes rules and responses for the Cisco Agent and the Checkpoint Agent.

To setup SiteMinder in the heterogeneous, single directory environment described above, you must:

1. Configure the system:
   a. Define two RADIUS Agents, as described in Define Agents for a Heterogeneous, Single Directory Environment (see page 756).
   b. Setup a user directory against which to authenticate RADIUS users, as described in Configure the User Directory (see page 757).
   c. Create one policy domain, as described in Create the Policy Domain.
   d. Create an authentication scheme, as described in Create the Authentication Scheme.

2. Configure the policy domain:

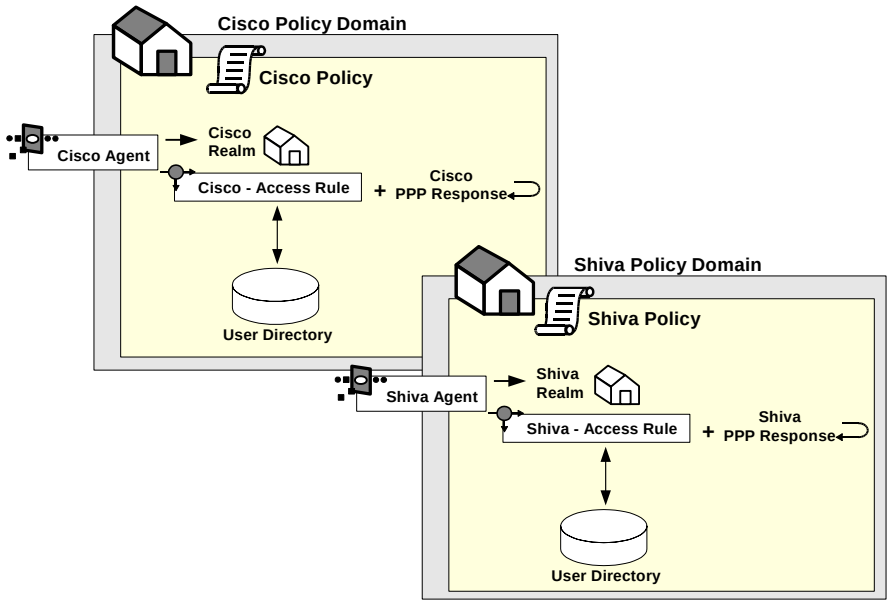
   a. Define two realms--one realm for the Cisco RAS and one realm for the Checkpoint firewall. Each realm binds a RADIUS Agent with a RADIUS authentication scheme.

   b. Define two rules that allow authenticated users to access the appropriate realm. Each rule binds a realm with an allow or deny access event.

   c. Define two responses that provide the user profile to the NAS device and configure the characteristics of the session using response attributes. A separate response must be defined for each NAS device because each device uses a different Dictionary file.

   d. Create one policy that binds the Cisco rule with the Cisco response and the Checkpoint rule with the Checkpoint response. This policy also binds the components of the policy domain (the rule and response groupings) with the RADIUS user directory.

A diagram of this policy domain is shown in the following graphic:



Define Agents for a Heterogeneous, Single Directory Environment

For this environment, you must configure two RADIUS Agents:

- One Agent must be associated with the Cisco RAS.
- One must be associated with the Checkpoint Firewall.
- Both RADIUS Agents must use 1 as the realm hint, which enables each Agent to identify the correct domain to protect.

Configure the User Directory

The Policy Server can authenticate users using the same user directory for both NAS devices.

Create the Policy Domain

The policy domain must identify the user directory that contains the names of the RADIUS users, the names of the Administrators who can modify the domain, and the realm that the RADIUS Agent is protecting. A RADIUS environment that uses only one user directory requires only one policy domain.

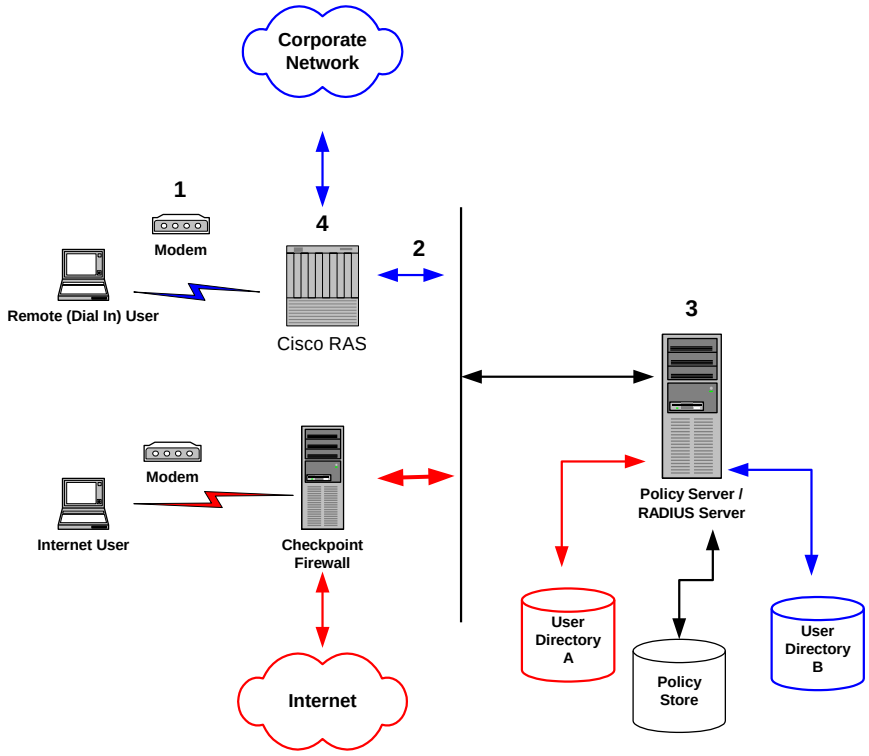
How to Authenticate Users in Heterogeneous RADIUS Environments with Two User Directories

The Policy Server can also be configured to authenticate users for multiple NAS devices when the user information for each device is located in separate user directories. The NAS devices can be of different vendor types.

There are several advantages to this configuration:

- Using two user directories in a single policy domain enables you to delegate the administration of each directory to a different person.
- By configuring the Policy Server to authenticate users for multiple RADIUS clients, you save time. You do not need to install and configure a separate authentication server for each RADIUS client.
- Using existing user directories is more efficient. You do not need to merge the user information into a single directory.

An example of a heterogeneous configuration that uses two user directories is illustrated in the following graphic:



Unlike the topology described in the previous section, this Policy Server uses *two* user directories to authenticate the users. User information for the Cisco RAS users is stored in User Directory A. User information for the Checkpoint firewall is stored in User Directory B. The Policy Server can authenticate users using both of these directories.

By dividing the configuration into two policy domains, the need for realm hints is eliminated. Each RADIUS Agent exists in a separate policy domain and is bound to only one realm.

The process of authentication when two user directories are used is as follows:

1. The remote user dials in from a modem and the Cisco RAS determines that it must use a RADIUS user profile to authenticate the user.
2. The RAS sends the user connection request to the Policy Server.
3. The Policy Server enacts the policy defined for the RAS, and the RADIUS Agent obtains the user's name and password using the authentication scheme configured for the Agent.

4. The Policy Server evaluates the user information against the user directory and policy store associated with the policy's domain.
5. The Policy Server sends an authentication response to the Cisco RAS and one of the following takes place:
   - If authentication is unsuccessful, the RAS refuses the connection.
   - If authentication is successful, the RAS receives a list of attributes from the user profile in the RADIUS server's database and establishes network access for the caller.

     The RAS notifies the Policy Server that the session has begun and when the session ends.

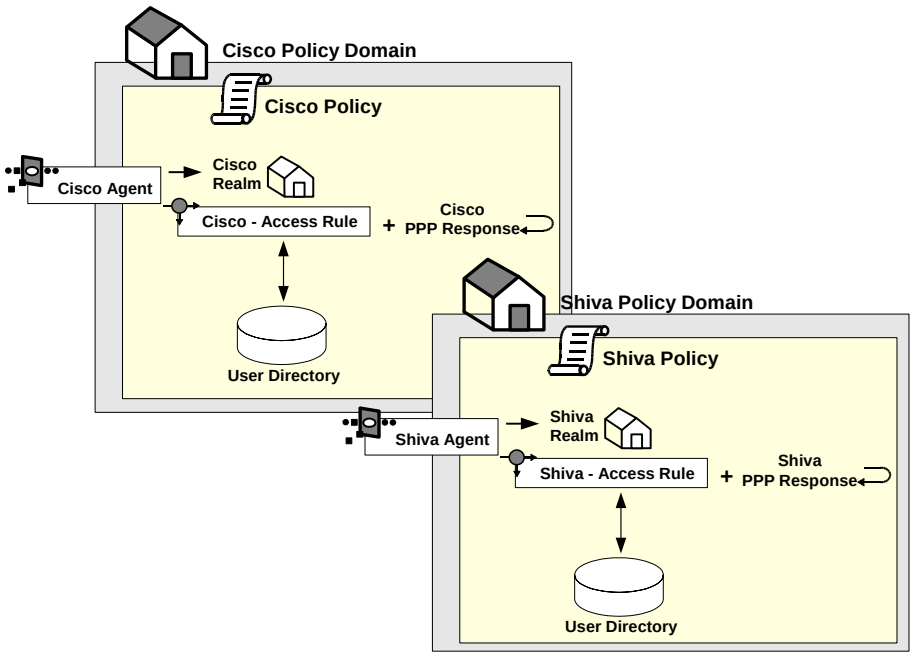
When the Internet user attempts to dial into the Internet Service Provider by using the Checkpoint Firewall, this same process of authentication occurs. However, the Policy Server evaluates the Internet user's authentication information against a different user directory.

How to Configure the System and Policy Domain

To configure the heterogeneous environment described above, which includes two user directories, you must:

1. Configure the system:
   a. Define two RADIUS Agents, as described in Define Agents for a Heterogeneous, Two Directory Environment (see page 760).
   b. Set up the user directories, as described in Set Up User Directories (see page 760).
   c. Create two policy domains, as described in Create Two Policy Domains (see page 761).
2. Configure the policy domain:
   a. Define one realm. The realm binds a RADIUS Agent with a RADIUS authentication scheme.
   b. Define a rule that enables authenticated users to access the realm. Each rule binds a realm with an allow or deny access event.
   c. Define a response that provides the user profile to the NAS device and optionally, configures the characteristics of the session using response attributes.
   d. Create a policy that binds the rule with the response. This policy also binds the rule and response with the RADIUS user directory.

A diagram of these two policy domains is shown in the following graphic:



Define Agents for a Heterogeneous Two Directory Environment

For this environment, you must configure two RADIUS Agents:

- One Agent must be associated with the Cisco RAS.
- One Agent must be associated with the Checkpoint Firewall.
- Neither RADIUS Agent requires a realm hint.

More information:

Define Agents for a Heterogeneous, Single Directory Environment (see page 756)

Set Up User Directories

Each of the user directories containing RADIUS user information must be configured in the Policy Server. Each directory will be associated with a separate policy domain so that separate administrators can be defined for each policy domain.

Create Two Policy Domains

One policy domain must be created for the Cisco RAS and one policy domain must be created for the Checkpoint firewall. When defining the policy domains, associate each domain with the appropriate user directory.

RADIUS Agents Group Overview

Creating a RADIUS Agent group enables you to manage multiple RADIUS Agents at once and eliminates the need to create and configure separate realms for each RADIUS Agent. Using one realm saves time because you can define the same session timeouts and the same authentication scheme for all RADIUS Agents simultaneously.

A group of RADIUS Agents could include Agents for different types of NAS devices. For instance, an Agent group could contain Agents for a Shiva LAN Rover RAS, a Checkpoint firewall, and a Cisco RAS. The Agent group containing all of these RADIUS Agents would be associated with a single realm, which defined the authentication scheme and session timeout requirements.

Agent groups are best suited for static environments that do not change often; Agent groups enable you to quickly configure SiteMinder to authenticate users of many NAS devices. If your environment is not static and frequently changes as new NAS devices are added or removed, you should probably avoid using Agent groups. Instead, it is usually easier to add and remove RADIUS Agents if they are not located in groups. If the Agents are separated and not grouped together, you can usually find specific Agents faster, and modify or remove policies more quickly.

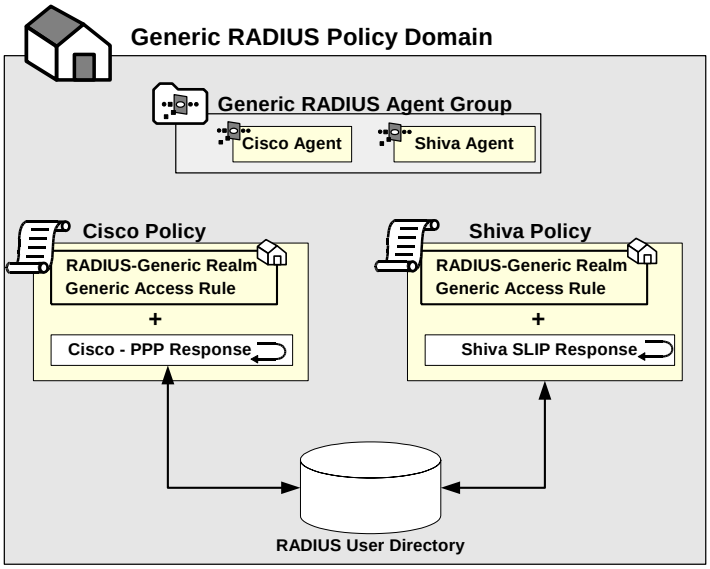
Set Up RADIUS Agent Groups

When using RADIUS Agent groups, you typically setup a separate policy for each type of RADIUS Agent. By using separate policies for each type of RADIUS Agent, you can share the common elements of the policy domain, such as the realm and a rule, in each policy. Sharing these common elements saves time.

Unlike the rule and realm, the response in each policy is not shared. Each policy has its own response, which corresponds to the device type of the RADIUS Agent in the policy. The attributes in a response match the attributes provided by the Dictionary file of the NAS device. For example, a response for a Cisco RAS would need to provide attributes that the Cisco RAS could interpret using the Cisco Dictionary file.

Note: All of the NAS devices represented in a RADIUS Agent group must share the same user directory. If they do not share the same user directory, they cannot exist in the same policy domain and therefore, they cannot share the same generic realms or generic rules.

The following example depicts one RADIUS Agent group that contains both an Agent for a Cisco RAS and an Agent for a Shiva RAS. The Agent group is shared by both the Cisco policy and the Shiva policy. Both of these policies share the same generic rule to allow access and the same generic realm, which binds the Agent group to the same authentication scheme. Notice, however, that the responses for each policy are unique.



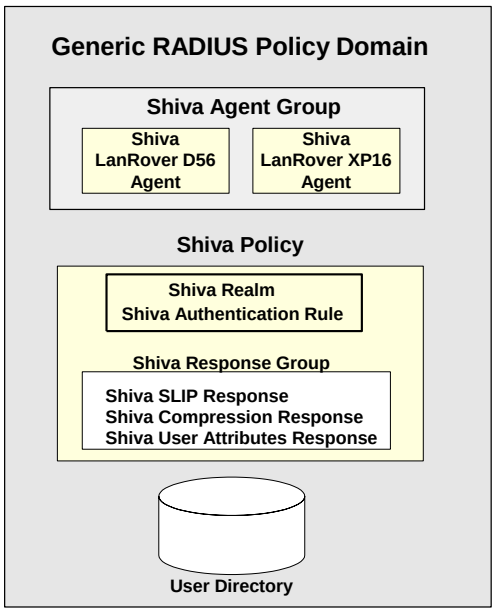
The procedure for setting up a RADIUS Agent Group is in Configure an Agent Group.

Group RADIUS Responses

A RADIUS response group is a collection of responses defined for the same type of Agent, such as Generic RADIUS. When a rule fires, all of the RADIUS responses paired with it in the response group are triggered.

The responses must be of the same Agent type in order to use the same Dictionary file. For example, you could combine two Cisco responses in the same group or two Generic RADIUS responses in the same group. However, you could not group a Generic RADIUS and a Cisco response in the same group.

The advantage of using RADIUS response groups is that it enables you to configure a policy domain using fewer policies. Instead of creating a separate realm and a separate policy for each RADIUS Agent, you could group RADIUS Agents of the same type, create one generic rule for authentication, and then group the responses for the rule. This type of configuration is shown in the following diagram:



Response groups also make it easier to add RADIUS devices to the environment. For instance, in an environment such as the one shown in the previous figure, you could quickly add another Shiva RAS to the RADIUS Agent group and this new RAS would automatically be configured with the appropriate rule and responses.

Troubleshoot and Test RADIUS

Once you have configured the Policy Server to act as a RADIUS authentication server, you can test and troubleshoot the policies using the tools described in subsequent topics.

Generate RADIUS Logs for Accounting and Debugging

RADIUS logs track debugging and accounting information generated by the Policy Server. Use the RADIUS log file to track the following:

- State of the Policy Server
- Connection attempts and session creations
- Information about each Policy Server action

Logs are turned on and off using the Policy Server Management Console from the Debug tab.

The Policy Server time stamps the log file with the date and time it was created. For example, "log.txt" can be specified as the name of the file. When the Policy Server is restarted and the Policy Server creates the log file, the date and time are added to the name.

If you are appending logging information to the same file, the date on the file reflects the date and time it was created. The timestamp is only updated if the Policy Server is restarted.

Read RADIUS Log Files With Smreadclog

This tool is used to read RADIUS log files generated by the Policy Server. It is useful for troubleshooting the Policy Server when used as a RADIUS authentication server. Options are provided to display individual RADIUS attributes that are exchanged between NAS and SiteMinder.

Smreadclog uses the following arguments to supply information required to read RADIUS log files:

-i*<input-file>*

Specifies the filename of the log file.

-o*<output-file>*

Specifies the filename of the output file.

-s*<secret>*

Specifies the shared secret that can be used to decode RADIUS passwords.

-r

Indicates that a hex dump of an entire RADIUS packet be displayed.

-a

Indicates that RADIUS attributes should be displayed individually.

-d

Indicates that RADIUS attributes should be displayed according to their definition in the policy store. This option displays actual attribute names as well as attribute values formatted based on their attribute type. Without this option, only the attribute name and value are displayed (as a hex string).

-p*<radius-server>*

Enables you to record and replay RADIUS activity of the Policy Server service against your RADIUS server.

-m*<authentication port>*

Specifies the port used for RADIUS authentication if that port is not the default port, 1645.

-n*<accounting port>*

Specifies the port used for RADIUS accounting if that port is not the default port, 1646.

To use smreadclog

1. Navigate to one of the following locations:
   - On NT, <*siteminder_installation*>\Bin

     where <*siteminder_installation*> is the installed location of SiteMinder.
   - On UNIX, <*siteminder installation*>/bin

     where <*siteminder_installation*> is the installed location of SiteMinder.
2. Enter the following command:

   ```
   smreadclog -i<input-file> -o<output-file>
   -s<secret> -r -a -d -p<radius-server> -m<portnumber>
   -n<portnumber>
   ```

   For example,

   ```
   smreadclog -iradiuslog.txt -oradiuslog2.txt
   -ssecret -r -a -d -p123.123.12.12
   ```

How to Test using the SiteMinder Test Tool

The SiteMinder Test Tool simulates the behavior of a RADIUS authentication server. Using the Test Tool, you can test policies that authenticate RADIUS users and ensure that the response attributes you configured are returning the appropriate data.

The process of testing RADIUS policies includes the following steps:

1. Create a RADIUS policy.
2. Configure the Policy Server Management Console to use RADIUS, as explained in Configure the Policy Server Management Console (see page 766).
3. Configure the SiteMinder Test Tool to test RADIUS policies, as explained in Test RADIUS Policies (see page 766).

Configure the Policy Server Management Console

To configure the Policy Server Management console

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. Use Administrator permissions even if you are logged in to the system as an Administrator. For more information, see the release notes for your SiteMinder component.

2. Select the Settings tab.
3. On the Settings tab, do the following:
   a. In the RADIUS UDP Ports group box, select the Enable check box.
   b. In the Authentication field, enter 1645.
   c. In the Accounting field, enter 1646.
4. On the Status tab, restart the Policy Server to enable the Policy Server configuration changes.

   You are now ready to test the RADIUS policies using the Test Tool.

Test RADIUS Policies

To test RADIUS policies

1. Start the Test Tool.

   Important! Before running a SiteMinder utility or executable on Windows Server 2008, open the command line window with administrator permissions. Open the command line window this way, even if your account has administrator privileges.

2. In the SiteMinder Agent group box, do the following:
   a. Select the RADIUS radio button.
   b. In the Secret field, enter the shared secret that was defined for the RADIUS agent in the SiteMinder Administration User Interface.
3. In the User Information group box, do the following:
   a. In the User field, enter the name of a user in the RADIUS user directory whose authentication will be tested.
   b. In the Password field, enter the user's password.
   c. Select the CHAP Password check box if you are using a RADIUS CHAP Authentication scheme.
4. In the Command group box, click IsAuthenticated.

The policy is tested and the response attributes appear in the Attributes group box.

Appendix E: Attributes and Expressions Reference

This section contains the following topics:

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 SiteMinder 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'

Iteration	Position	URL
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 816)
LOOP Function--Call a Virtual Attribute in a Loop (see page 833)

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 776)

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 775)

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 777)

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 777)

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 779)

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 778)

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 780)
Containment Operators (see page 780)

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 779)
Containment Operators (see page 780)

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 779)
Ends-with Operators (see page 780)

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 785)
Set Union Operators (see page 786)

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 781)
Set Union Operators (see page 786)

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 781)
Set Intersection Operators (see page 785)

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 787)
OR Operator (see page 787)
Exclusive OR Operator (see page 788)

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 786)
OR Operator (see page 787)
Exclusive OR Operator (see page 788)

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 786)
AND Operator (see page 787)
Exclusive OR Operator (see page 788)

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 786)
AND Operator (see page 787)
OR Operator (see page 787)

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 790)
Arithmetic Multiplication Operator (see page 790)
Arithmetic Division Operator (see page 791)

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 789)
Arithmetic Multiplication Operator (see page 790)
Arithmetic Division Operator (see page 791)

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 789)
Arithmetic Subtraction Operator (see page 790)
Arithmetic Division Operator (see page 791)

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 789)
Arithmetic Subtraction Operator (see page 790)
Arithmetic Multiplication Operator (see page 790)

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

Miscellaneous Functions

- EVALUATE

ABOVE Function--Is User Above Specified LDAP DN

The ABOVE function returns a TRUE if the specified user (user_DN) exists within a container that is *above* the specified container (root_DN).

Syntax

The ABOVE function has the following format:

```
ABOVE(root_DN, user_DN)
```

Parameters

The ABOVE function accepts the following parameters:

root_DN (string)

user_DN (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 801)
FIND Function--Return Position in String (see page 823)

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 842)
ORBITS Function--Perform a Bitwise OR Operation (see page 845)
XORBITS Function--Perform a Bitwise XOR Operation (see page 868)

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 specified user (user_DN) exists within the specified container (root_DN).

Syntax

The AT function has the following format:

```
AT(root_DN, user_DN)
```

Parameters

The AT function accepts the following parameters:

root_DN (string)

user_DN (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 797)
FIND Function--Return Position in String (see page 823)

BELOW Function--Is User Below Specified LDAP DN

The BELOW function returns a TRUE if the specified user (user_DN) exists within a container that is *below* the specified container (root_DN)

Syntax

The BELOW function has the following format:

```
BELOW(root_DN, user_DN)
```

Parameters

The BELOW function accepts the following parameters:

root_DN (string)

user_DN (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 844)
STRING Function--Convert to a String (see page 859)

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 834)
RPAD Function--Pad a String on the Right (see page 852)

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 820)
PARENTDN Function--Retrieve Parent in LDAP Tree (see page 846)
RDN Function--Retrieve First Component of LDAP DN (see page 849)
RELATIONDN Function--Compare Two Distinguished Names (see page 850)

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 816)
FILTER Function--Test Set Elements (see page 822)
SORT Function--Sort a Set (see page 857)

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

Code	Resulting Format
%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 814)
DOY Function--Return Day of Year (see page 815)

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 813)
DOY Function--Return Day of Year (see page 815)

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 813)
DOW Function--Return Day of Week (see page 814)

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 or integer depending on the attribute type that the function expects.

Example 1

Virtual User Attribute #GetCN set to RDN(STRING(%0),FALSE)

ENUMERATE(SM_USERGROUPS, #GetCN)

Example 2

ENUMERATE(SM_USERGROUPS, STRING(RDN(%0, FALSE)))

More information:

COUNT Function--Count the Elements in a Set (see page 808)
FILTER Function--Test Set Elements (see page 822)
LOOP Function--Call a Virtual Attribute in a Loop (see page 833)
SORT Function--Sort a Set (see page 857)
Pasting (see page 773)

ERROR Function--Write Error Message to Console Log

The ERROR function writes the specified error message to SiteMinder'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.

Remarks

Priviledged: Yes

Example

Return_value=ERROR('Invalid Access')
Return_value=''

More information:

INFO Function--Write INFO Message to Console Log (see page 828)
TRACE Function--Write Trace Entry to Console Log (see page 860)
WARNING Function--Write WARNING Message to Console Log (see page 868)

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 828)

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 807)
PARENTDN Function--Retrieve Parent in LDAP Tree (see page 846)
RDN Function--Retrieve First Component of LDAP DN (see page 849)
RELATIONDN Function--Compare Two Distinguished Names (see page 850)

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
*	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 808)
ENUMERATE Function--Test Set Elements (see page 816)
SORT Function--Sort a Set (see page 857)

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 797)
BEFORE Function--Find a String (see page 801)

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 827)

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 826)

INFO Function--Write INFO Message to Console Log

The INFO function writes the string argument to the SiteMinder 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 820)

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 847)
UCASE Function--Convert to Upper Case (see page 862)

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 851)
MID Function--Return Part of a String (see page 838)

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(#virtual_user_attribute, start_value, end_value, [step,] )
```

Parameters

The LOOP function accepts the following parameters:

#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.

start_value (number)

end_value (number)

step (number)

(Optional) The default is 1. Negative values are allowed.

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(#Padset ,1, 5)
Return_set="001^002^003^004^005"
Return_set=LOOP(#Padset ,1, 5, 2)
Return_set="001^003^005"
Return_set=LOOP(#Padset ,5, 1, -1)
Return_set="005^004^003^002^001"
```

More information:

Pasting (see page 773)
ENUMERATE Function--Test Set Elements (see page 816)
FILTER Function--Test Set Elements (see page 822)

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 805)
RPAD Function--Pad a String on the Right (see page 852)

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 854)

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 839)

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 830)
RIGHT Function--Retrieve Characters from a String (see page 851)

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 836)

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 826)
HOUR24 Function--Convert to Hour (see page 827)
SECOND Function--Return the Number of Seconds in a Date (see page 855)

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 813)
DOW Function--Return Day of Week (see page 814)
DOY Function--Return Day of Year (see page 815)
YEAR Function--Return the Year Component of a Numeric Date (see page 869)
YEAR4 Function--Return the Year Component of a Date (4 digits) (see page 870)

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 799)
ORBITS Function--Perform a Bitwise OR Operation (see page 845)
XORBITS Function--Perform a Bitwise XOR Operation (see page 868)

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 843)

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 843)

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 803)
STRING Function--Convert to a String (see page 859)

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 799)
NOTBITS Function--Perform a Bitwise NOT (see page 842)
XORBITS Function--Perform a Bitwise XOR Operation (see page 868)

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 863)

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 807)
EXPLODEDN Function--Convert LDAP DN to Set (see page 820)
PARENTDN Function--Retrieve Parent in LDAP Tree (see page 846)
RELATIONDN Function--Compare Two Distinguished Names (see page 850)

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 830)
MID Function--Return Part of a String (see page 838)

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 835)

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 840)

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 SiteMinder.

The SET function returns TRUE if SiteMinder returns success on the modification.

If the attribute is not visible to SiteMinder, 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 SiteMinder 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 825)

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 803)
NUMBER Function--Convert to a Numeric Value (see page 844)

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 SiteMinder 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.

Remarks

Priviledged: Yes

Example

```
Return_value=TRACE("Executing Code")
Return_value=TRUE
```

More information:

ERROR Function--Write Error Message to Console Log (see page 818)
INFO Function--Write INFO Message to Console Log (see page 828)
WARNING Function--Write WARNING Message to Console Log (see page 868)

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 830)
PCASE Function--Convert a String to Proper Case (see page 847)

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 865)

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 865)

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, SiteMinder 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
```

WARNING Function--Write WARNING Message to Console Log

The WARNING function writes the string argument to the SiteMinder 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 799)
NOTBITS Function--Perform a Bitwise NOT (see page 842)
ORBITS Function--Perform a Bitwise OR Operation (see page 845)

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.

Appendix F: SiteMinder Kerberos Authentication

This section contains the following topics:

Kerberos Overview

Kerberos is a standard protocol, designed at MIT, to provide a means of authentication between a client and a server on an open network. The Kerberos protocol protects messages from eavesdropping and replay attacks. Kerberos uses shared secrets, symmetric keys, and Kerberos services. Microsoft Windows operating environments use Kerberos V5 as the default authentication package. Solaris 10 also includes Kerberos V5.

In a Kerberos environment, user accounts and service accounts are named principals. Kerberos uses a trusted third party (the Key Distribution Center, or KDC) to mediate message exchanges between principals. The purpose of the Key Distribution Center is to reduce the risks inherent in exchanging keys.

Kerberos authentication is based on messages that request and deliver tickets. The Key Distribution Center processes two types of tickets:

- Ticket-Granting Ticket (TGT) — used internally by the KDC to transport a requestor's credentials to the ticket-granting service (TGS).
- Session Ticket — used by the ticket-granting service (TGS) to transport the requestor's credentials to the target server or service.

Kerberos uses keytab files for logging in to the KDC. Keytab files consist of pairs of Kerberos principals and encrypted keys derived from a Kerberos password.

The Kerberos protocol message exchange can be summarized in a simplified way as follows:

1. When a user logs in, the client contacts the KDC Authentication Service, requesting a short-lived message (the ticket-granting ticket) containing the user identity information.
2. The KDC authentication service generates the TGT and creates a session key that the client can use to encrypt communication with the ticket-granting service.

3. When a user requests access to local or network resources, the client presents the ticket-granting ticket (TGT), an authenticator, and the Service Principal Name (SPN) of the target server to the KDC.
4. The ticket-granting service examines the ticket-granting ticket and the authenticator. If these credentials are acceptable, the ticket-granting service creates a service ticket, which includes the user identity information copied from the TGT. The service ticket is sent back to the client.

   Note: The ticket-granting service cannot determine whether the user is granted access to the target resource. The ticket-granting service only authenticates the user and returns the session ticket.
5. After the client has the session ticket, the client sends the session ticket and a new authenticator to the target server, requesting access to a resource.
6. The server decrypts the ticket, validates the authenticator, and grants the user access to the resource.

How To Configure SiteMinder Kerberos Authentication

Kerberos authentication supports various configuration scenarios, depending on the host environments of the client and server. Although each scenario is slightly different, implementing Kerberos authentication in a SiteMinder environment requires the following administrator tasks:

- Configuration at the domain controller for the Kerberos Key Distribution Center (KDC)
- Configuration at the Policy Server
- Configuration on the SiteMinder administrative UI of a custom authentication scheme
- Configuration at the web server
- Configuration at the Windows workstation

Kerberos KDC Configuration at the Domain Controller

When using Kerberos, the domain controller is the key distribution center (KDC) for the Kerberos Realm. In a pure Windows environment, a Kerberos Realm is equivalent to a Windows Domain. The domain controller host provides storage for the user, service accounts, credentials, the Kerberos ticketing services, and Windows Domain services.

A keytab file is required for Kerberos authentication, which lets users authenticate with the KDC without being prompted for a password. The keytab file is created with the ktpass utility. The ktpass command tool utility is a Windows support tool. The ktadd utility is the equivalent on UNIX.

KDC configuration for SiteMinder on the domain controller host (Windows or UNIX) follows this general sequence:

1. Create a user account. This account is for logging in to the workstation.
2. Create a service account for the web server for logging in to the web server host.
3. Create a service account for the Policy Server for logging in to the Policy Server host.
4. Associate the web server account with a web server principal name.
5. Create a keytab file, which is transferred to the web server host.
6. Associate the Policy Server account with a Policy Server principal name.
7. Create another keytab file, which is transferred to the Policy Server host.
8. Specify that the web server and Policy Server accounts are Trusted for Delegation.

Important! For any service to use Kerberos protocol, be sure to create the Service Principal Name (SPN) in a standard format, that is, service/fqdn_host@REALM_NAME.

More information:

KDC Configuration on Windows 2003 Example (see page 877)
KDC Configuration on UNIX Example (see page 880)

Kerberos Authentication Configuration at the Policy Server

In addition to the standard Policy Server configuration, Kerberos authentication requires the following steps:

- Add these three parameters to the Agent Configuration Object:

Parameter	Value
KCCExt	.kcc
HttpServicePrincipal	Specifies the web server principal name. **Example:** HTTP/win2k3iis6.test.com@TEST.COM
SmpsServicePrincipal	Specifies the Policy Server principal name. **Example:** smps@win2kps.test.com

- Configure a Kerberos configuration file (krb5.ini) and place it in the system root path on Windows and in /etc/krb5/ on UNIX.
- Deploy the keytab file created on the KDC that contains the Policy Server principal credentials to a secure location on the Policy Server.

Important! If the Policy Server is installed on Windows and the KDC is deployed on UNIX, be sure to perform the additional configuration on the Policy Server host using the Ksetup utility (see page 876).

More information:

Kerberos Configuration at the Policy Server on Windows Example (see page 881)
Kerberos Configuration at the Policy Server on UNIX Example (see page 883)

Kerberos Authentication Configuration at the Web Server

Configuring a Windows or UNIX web server to support Kerberos authentication follows these general steps:

1. Install a SiteMinder Web Agent with SiteMinder Kerberos Authentication Scheme support.
2. Register a trusted host with the Policy Server and configure the Web Agent.
3. Configure a Kerberos configuration file (krb5.ini):
   - Configure the KDC for the Kerberos realm (domain).
   - Configure krb5.ini to use the keytab file containing the credentials of the web server principal.
   - Place krb5.ini in the system root path on Windows and in /etc/krb5/ on UNIX.
4. Deploy the keytab file (created on the KDC) containing the web server credentials to a secure location on the web server.

Important! If the web server is installed on Windows and the KDC is deployed on UNIX, be sure to perform additional configuration on the web server using the Ksetup utility (see page 876).

Kerberos Authentication Configuration at the Windows Workstation

To support Kerberos authentication, several Internet Explorer settings are required, and the workstation host is added to the KDC domain.

Important! If the KDC is deployed on UNIX, be sure to perform the additional required configuration on the workstation using the Ksetup utility (see page 876).

To configure the Windows workstation to support Kerberos authentication

1. Add the host for the Windows workstation to the KDC domain.
2. Log in to the host using user account created on the KDC.
3. Configure Internet Explorer to pass credentials automatically:
    1. Initiate an instance of the IE web browser.
    2. Select the Internet options menu.
    3. Select the Security tab.
    4. Select Local intranet tab.
    5. Click Sites and select all three checkboxes.
    6. Select the Advanced tab and add http://*.*domain.com* to local intranet zone.
    7. Select the Custom level tab under security settings and select Automatic logon only in intranet zone under the User Authentication tab.
    8. Select the Advanced tab from Internet options and select the Enable Integrated Windows authentication (requires restart) option.
    9. Close the browser.

Configure a Kerberos Authentication Scheme

An authentication scheme is required to support Kerberos authentication in the SIteMinder environment. Associate this authentication scheme with any realm whose protected resources use Kerberos authentication.

Refer to these instructions for information about creating a Kerberos authentication scheme.

Configure Kerberos External Realm on Windows Host

For the Windows workstation to use a Kerberos KDC deployed on UNIX, you must configure both the Kerberos KDC server and the workstation.

In the Kerberos realm, create a host principal for the Windows host. Use the following command:

```
kadmin.local: addprinc host/machine-name.dns-domain_name.
```

For example, if the Windows workstation name is W2KW and the Kerberos realm name is EXAMPLE.COM, the principal name is host/w2kw.example.com.

Because a Kerberos realm is not a Windows domain, the KDC operating environment must be configured as a member of a workgroup, which happens automatically when you follow this process:

1. Remove the host from the Windows domain.
2. Add the test user, for example, testkrb, to the local user database.
3. Add the Kerberos Realm:

   ```
   ksetup /SetRealm EXAMPLE.COM
   ```
4. Restart the host.
5. Add the KDC :

   ```
   ksetup /addkdc EXAMPLE.COM rhasmit
   ```
6. Set a new password:

   ```
   ksetup /setmachpassword password
   ```

   Note: The password used here is same as the one used while creating the host principal account in the MIT KDC.
7. Restart the host.

   Note: Whenever changes are made to the external KDC and realm configuration, a restart is required.
8. Set the Realm Flag

   ```
   ksetup /SetRealmFlags EXAMPLE.COM delegate
   ```
9. Run AddKpasswd:

   ```
   ksetup /AddKpasswd EXAMPLE.COM rhasmit
   ```
10. Use Ksetup to configure single sign on to local workstation accounts by defining the account mappings between the Windows host accounts to Kerberos principals. For example:

    ```
    ksetup /mapuser testkrb@EXAMPLE.COM testkrb
    ksetup /mapuser * *
    ```

The second command maps clients to local accounts of the same name. Use Ksetup with no arguments to see the current settings.

Kerberos Configuration Examples

The configurations that follow include examples of the specifics, such as keytab file creation, required to implement Kerberos authentication in a SiteMinder environment. Note that additional configuration is required when the KDC is deployed in a UNIX operating environment and the Policy Server, or web server, or workstation is in a Windows operating environment.

KDC Configuration on Windows 2003 Example

The steps listed following exemplify how to configure a Windows 2003 domain controller to support SiteMinder Kerberos authentication.

1. Promote a Windows 2003 SP 1 Server to a domain controller (named test.com in this example) using Windows dcpromo utility.
2. Raise the domain functional level:
    1. Open the Active Directory users and computers dialog from Administrative tools.
    2. Right-click the test.com drop-down on the left side of dialog.
    3. Click Raise domain functional level.
    4. Raise the domain functional level of Active directory to Windows Server 2003.

       Important! This step is irreversible.
3. Create a user account (for example, testkrb). Provide a password for this account. Clear the option, User must change password at next logon. Add this account to the domain administrators group so that the user has permissions to login. The Windows workstation uses this account to log in to test.com.
4. Create a service account for the web server (for example, wasrvwin2k3iis6). Create a password for this account. Clear the option, User must change password at next logon. Add this account to the domain administrators group so that the user has permissions to login.The Windows 2003 IIS web server host (win2k3iis6) uses this account to log in to test.com.
5. Create a service account for the Policy Server (polsrvwin2kps). Provide a password for this account. Clear the option, User must change password at next logon. Add this account to the domain administrators group so that the user has permissions to login.The Win2k3 Policy Server host (win2kps) uses this account to log in to test.com.
6. Join the web server (win2k3iis6) and the Policy Server (win2kps) hosts to the test.com domain using their service accounts created in Steps 4 and 5.

7. Associate the web server account (wasrvwin2k3iis6) with a web server principal name (HTTP/win2k3iis6.test.com@TEST.COM) and create a keytab file using the Ktpass utility.The syntax differs depending on whether the Policy Server is on Windows or on UNIX.

   Note: The Ktpass command tool utility is a Windows support tool. You can install it from MSDN download or an installation CD. Always verify the version of support tools. The default encryption type must always be RC4-HMAC. The encryption type can be confirmed by running ktpass /? at the command prompt.

   When the Policy Server is on Windows:

   ```
   ktpass -out c:\wasrvwin2k3iis6.keytab -princ HTTP/win2k3iis6.test.com@TEST.COM
   -ptype KRB5_NT_PRINCIPAL -mapuser wasrvwin2k3iis6 -pass <<password>>

   Targeting domain controller: winkdc.Test.com
   Using legacy password setting method
   Successfully mapped HTTP/win2k3iis6.test.com to wasrvwin2k3iis6.
   Key created.
   Output keytab to c:\wasrvwin2k3iis6.keytab:
   Keytab version: 0x502
   keysize 67 HTTP/win2k3iis6.test.com@TEST.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 2
   etype 0x17 (RC4-HMAC) keylength 16 (0xfd77a26f1f5d61d1fafd67a2d88784c7)
   ```

   The password is the same as the one used for creating the service account for the web server.

   When the Policy Server is on UNIX:

   ```
   ktpass -out d:\sol8sunone_host.keytab -princ host/sol8sunone.test.com@TEST.COM
   -pass <<password>> -mapuser sol8sunone –crypto DES-CBC-MD5 +DesOnly -ptype
   KRB5_NT_PRINCIPAL –kvno 3

   Targeting domain controller: winkdc.test.com
   Successfully mapped host/sol8sunone.test.com to sol8sunone.
   Key created.
   Output keytab to d:\sol8sunone_host.keytab:
   Keytab version: 0x502
   keysize 52 host/sol8sunone.test.com@TEST ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
   0x3 (DES-CBC-MD5) keylength 8 (0xb5a87ab5070e7f4a)
   Account sol8sunone has been set for DES-only encryption.
   ```

8. Associate the Policy Server account (polsrvwin2kps) with a Policy Server principal name (smps/win2kps.test.com@TEST.COM) and create another keytab file destined for the Policy Server host (win2kps).

   When the Policy Server is on Windows

   ```
   Ktpass -out c:\polsrvwin2kps.keytab –princ smps/win2kps.test.com@TEST.COM
   -ptype KRB5_NT_PRINCIPAL -mapuser polsrvwin2kps -pass <<password>>
   Targeting domain controller: winkdc.Test.com
   Using legacy password setting method
   Successfully mapped smps/win2kps.test.com to polsrvwin2kps.
   Key created.
   Output keytab to c:\polsrvwin2kps.keytab:
   Keytab version: 0x502
   keysize 72 smps/win2kps.test.com@TEST.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 2 etype
   0x17 (RC4-HMAC) keylength 16 (0xfd77a26f1f5d61d1fafd67a2d88784c7)
   ```

   The password is same as the one used for creating the service account for Policy Server.

   When the Policy Server is on UNIX:

   ```
   ktpass -out d:\sol8polsrv.keytab -princ host/sol8sunone.test.com@TEST.COM -pass
   <<password>> -mapuser sol8sunone –crypto DES-CBC-MD5 +DesOnly -ptype
   KRB5_NT_PRINCIPAL –kvno 3

   Targeting domain controller: winkdc.test.com
   Successfully mapped host/sol8sunone.test.com to sol8sunone.
   Key created.
   Output keytab to d:\sol8polserv.keytab:
   Keytab version: 0x502
   keysize 52 host/sol8sunone.test.com@TEST ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
   0x3 (DES-CBC-MD5) keylength 8 (0xb5a87ab5070e7f4a)
   Account sol8sunone has been set for DES-only encryption.
   ```

9. Specify that the web server and Policy Server service accounts are Trusted for Delegation as follows:

   1. Right-click the service account (polsrvwin2kps/wasrvwin2k3iis6) properties.
   2. Select the Delegation tab.
   3. Select the second option, Trust this user for delegation to any service (Kerberos only)

      Or, select the third option, Trust this user for delegation to specified service. Select the Use Kerberos only option button, and add the corresponding service principal name.

The domain controller is ready for SiteMinder Kerberos authentication.

KDC Configuration on UNIX Example

The process listed following exemplifies how to configure a KDC Kerberos Realm on a UNIX host to support SiteMinder Kerberos authentication.

1. Install MIT Kerberos, if necessary.
2. Use the kdb5_util command to create the Kerberos database and an optional stash file. The stash file is used to authenticate the KDC to itself automatically before starting the kadmind and krb5kdc daemons as part of the host auto-boot sequence.

   Both the stash file and the keytab file are potential point-of-entry for a break-in. If you install a stash file, it must be readable only by root, must not be backed up, and must exist only on the KDC local disk. If you do not want a stash file, run the kdb5_util without the -s option.

   This example generates the following five database files in the directory specified in kdc.conf file:

   - Two Kerberos database files: principal.db and principal.ok
   - One Kerberos administrative database file: principal.kadm5
   - One administrative database lock file: principal.kadm5.lock
   - One stash file: .k5stash

   ```
   [root@rhasmit init.d]# kdb5_util create -r EXAMPLE.COM –s
   Initializing database '/var/kerberos/krb5kdc/principal' for realm
   'EXAMPLE.COM',
   master key name 'K/M@EXAMPLE.COM'
   You will be prompted for the database Master Password.
   It is important that you NOT FORGET this password.
   Enter KDC database master key:
   Re-enter KDC database master key to verify:
   ```

3. Create a user principal (testkrb).
4. Create a user principal (for example, testwakrb), a host principal (host/win2k3iis6.example.com@EXAMPLE.COM, and a service principal (HTTP/win2k3iis6.example.com@EXAMPLE.COM) for the web server host. The password used for creating host account must be same as the password specified when using the ksetup utility on the web server host.
5. Create a user principal (testpskrb), host principal (host/win2kps.example.com@EXAMPLE.COM) and service principal (smps/win2kps.example.com@EXAMPLE.COM) for the Policy Server host. The password used for creating host account must be same as the password specified when using the ksetup utility on the Policy Server host.

6. Create a keytab file for the web server service principal as follows:

   ```
   ktadd -k /tmp/win2k3iis6.keytab HTTP/win2k3iis6.example.com
   ```

7. Create keytab for Policy Server service principal as follows:

   ```
   ktadd -k /tmp/win2kps.keytab smps/win2kps.example.com
   ```

The Kerberos Realm is configured for SiteMInder on a UNIX host.

Kerberos Configuration at the Policy Server on Windows Example

The following procedure shows an example of how to configure a Policy Server on Windows to support SiteMinder Kerberos authentication.

Note: If the Policy Server is installed on Windows and the KDC is deployed on UNIX, be sure to perform additional required configuration on the Policy Server host using the Ksetup utility (see page 876).

Follow these steps:

1. Install and configure the SiteMinder Policy Server.
2. Install and configure policy store directory services.
3. Log in to the Policy Server host with the service account (for example, polsrvwin2kps) created in Active Directory on the Windows domain controller.
4. Add a Host Configuration Object referencing the Policy Server.
5. Create an Agent Configuration Object and add these three new parameters:

Parameter	Value
KCCExt	.kcc
HttpServicePrincipal	Specifies the web server principal name. **Example:** HTTP/win2k3iis6.test.com@TEST.COM
SmpsServicePrincipal	Specifies the Policy Server principal name. **Example:** smps@win2kps.test.com

6. Create a user directory.
7. Create a user, for example, testkrb, in the user directory.

8. Configure a new Authentication Scheme using the SiteMinder Admin UI:
    1. Create the scheme using the custom template.
    2. Specify the SiteMinder Kerberos Authentication Scheme library.
    3. Select the parameter field and specify the following three semicolon-delimited values in the specified order:
        - Server name and target fields.
        - Policy Server principal name from the Windows 2003 Kerberos realm.
        - Mapping between the user principal and an LDAP search filter.

        Sample parameter field:

        ```
        http://win2k3iis6.test.com/siteminderagent/Kerberos/creds.kcc;smps/win2kps.test.com@TES.COM;(uid=%{UID})
        ```
9. Configure a policy domain.
10. Add a realm to protect a resource using the Authentication Scheme.
11. Add Rules and Policies to allow access for the user, testkrb.
12. Configure a Kerberos configuration file (krb5.ini) and place krb5.ini in the Windows system root path:
    - Configure the KDC for the Windows 2003 Kerberos realm (domain) to use the Windows 2003 domain controller.
    - Configure krb5.ini to use the Windows 2003 KDC keytab file containing the Policy Server principal credentials.

      See the following sample krb5.ini:

    ```
    [libdefaults]
    default_realm = TEST.COM
    default_keytab_name = C:\WINDOWS\krb5.keytab
    default_tkt_enctypes = rc4-hmac des-cbc-md5
    default_tgs_enctypes = rc4-hmac des-cbc-md5
    [realms]
    TEST.COM = {
    kdc = winkdc.test.com:88
    default_domain = test.com
    }
    [domain_realm]
    .test.com = TEST.COM
    ```
13. Deploy the Windows KDC keytab file containing the Policy Server principal credentials to a secure location on the Policy Server.

The Policy Server on a Windows host is configured for Kerberos authentication.

Kerberos Configuration at the Policy Server on UNIX Example

The following procedure shows an example of how to configure a Policy Server on a UNIX host to support SiteMinder Kerberos authentication.

Follow these steps:

1. Create a user, for example, sol8psuser, with the same password used for creating a service account for the Policy Server host (sol8ps) in Active Directory.
2. Add the host to the test.com domain and login to host with user sol8psuser.
3. Install and configure SiteMinder Policy Server.
4. Install and configure policy store directory services.
5. Add a Host Configuration Object referencing the Solaris Policy Server.
6. Add an Agent Configuration Object and add the following three new parameters:

Parameter	Value
KCCExt	.kcc
HttpServicePrincipal	Specify the web server principal name. **Example:** HTTP/win2k3iis6.test.com@TEST.COM
SmpsServicePrincipal	Specify the Policy Server principal name. **Example:** smps@win2kps.test.com

7. Create a user directory.
8. Create a user, for example, testkrb, in the user directory.
9. Configure a new Authentication Scheme using the SiteMinder Admin UI:
   1. Create the scheme using the custom template.
   2. Specify the SiteMinder Kerberos Authentication Scheme library.
   3. Select the parameter field and specify the following three semicolon-delimited values in the specified order:
      - Server name and target fields.
      - Policy Server principal name from the Windows 2003 Kerberos realm.
      - Mapping between the user principal and an LDAP search filter.

      Sample parameter field:

      ```
      http://sol8sunone.test.com/siteminderagent/Kerberos/creds.kcc;smps/sol8ps
      .test.com@TEST.COM;(uid=%{UID})
      ```
10. Configure a policy domain.
11. Add a realm to protect a resource using the Authentication Scheme.
12. Add Rules and Policies to allow access for the user, testkrb.

13. Configure a Kerberos configuration file (krb5.ini) and place krb5.ini in the /etc/krb5 system path.

    - Configure the KDC for the Windows 2003 Kerberos realm (domain) to use the Windows 2003 domain controller.
    - Configure krb5.ini to use the Windows 2003 KDC keytab file containing the Policy Server principal credentials.

      See the following sample krb5.ini:

```
[libdefaults]
 ticket_lifetime = 24000
 default_realm=TEST.COM
 default_tgs_enctypes = des-cbc-md5
 default_tkt_enctypes = des-cbc-md5
 default_keytab_name = FILE:/etc/krb5.keytab
 dns_lookup_realm = false
 dns_lookup_kdc = false
 forwardable = true
 proxiable = true
[realms]
 TEST.COM = {
  kdc = winkdc.test.com:88
  admin_server = winkdc.test.com:749
  default_domain = test.com
 }
[domain_realm]
 .test.com=TEST.COM
  test.com=TEST.COM
```

14. Use the ktutil utility to merge the keytab files (sol8ps_smps.keytab & sol8ps_host.keytab) containing the host principal and service principal names for the Policy Server host in the /etc/krb5.keytab file:

```
ktutil: rkt sol8ps_host.keytab
ktutil: wkt /etc/krb5.keytab
ktutil: q
ktutil: rkt sol8ps_smps.keytab
ktutil: wkt /etc/krb5.keytab
ktutil: q
```

15. Verify the created krb5.keytab as follows:

```
klist -k
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
----
--------------------------------------------------------------------------
   3 host/sol8ps.test.com@TEST.COM
   3 smps/sol8ps.test.com@TEST.COM
```

16. Deploy the Windows 2003 KDC keytab file containing the host and Policy Server principal credentials to a secure location on the Policy Server.
17. Verify that the following environment variable is set before starting the Policy Server:

```
KRB5_CONFIG=/etc/krb5/krb5.conf
```

The Policy Server on a UNIX host is configured for Kerberos authentication.

Verify that a Resource is Protected

You can verify that a resource in your Kerberos domain is protected by creating a SiteMinder policy using the Kerberos Authentication scheme and attempting to access the resource with a user principle.

To verify that a resource is protected by Kerberos authentication

1. Log into the Administrative UI.

   Note: When you create or modify a Policy Server object in the FSS Administrative UI, use ASCII characters. Object creation or modification with non-ASCII characters is not supported.
2. Configure a policy domain.
3. Configure a realm using the Kerberos Authentication scheme.
4. Configure a rule to protect a specific resource within your Kerberos domain.
5. Configure a policy to protect the Kerberos resource, and add a test user to the policy.
6. Log in to the Kerberos domain as a user principle and attempt to access the protected resource.

SiteMinder authenticates the user using the KDC security token.

Troubleshooting SiteMinder Kerberos Authentication

Be aware of the following issues when working with SiteMinder Kerberos authentication:

- Be sure that the SiteMinder system clocks are synchronized (within 2 minutes) to the KDC system clock. Otherwise, Kerberos authentication fails because of clock skew errors.
- All the hosts must have suitable entries for IP Address, fully qualified domain name (FQDN) and hostname in the /etc/hosts file. The order of these entries also matters in some cases. Also, include at least single space between these entries:

  ```
  IP Address  FQDN hostname
  ```

- Use the Windows resource tool, kerbtray, to see the Kerberos tickets description. The GUI shows the various tickets received by KDC client, its flags, encryption types, and time stamp.
- Use klist (UNIX KDC clients) to list tickets present in the credential cache. The flag –f provides information about tickets flags. These tickets are forwarded to complete the Kerberos authentication process.

  For example, see sample klist output listed following.

  ```
  bash-2.05$ klist
  Ticket cache: /tmp/krb5cc_1002
  Default principal: HTTP/sol8sunone.test.com@TEST.COM

  Valid starting            Expires                 Service principal
  Wed Dec 26 15:00:03 2007  Wed Dec 26 21:40:03 2007 krbtgt/TEST.COM@TEST.COM
  ```

- Always use the kinit program for the user and service principal at the MIT KDC host and UNIX KDC client.

  The kinit syntax for user principal is:

  ```
  kinit principalname
  ```

  This command prompts for the password used while creating the user

  principal.

  The kinit syntax (it can vary from host to host) for service principal is:

  ```
  kinit –k [–t keytab location containing service principal] principalname
  ```

  The kinit command does not prompt for a password, because it uses a keytab file to authenticate the service principal.
- Install any network packet trace utility on the workstation to see the Kerberos tokens exchanged between the browser and web server. The token starting with TIR indicates NTLM tokens, and tokens starting with YII denote the Kerberos tokens.
- Log off the workstation host after any change in encryption type at the KDC. Also, restart the Policy Server and web server services after any changes are made to the Kerberos configuration or the keytab files.

- The KDC does not support post-dated tickets.
- Only DES-CBC-MD5 and DES-CBC-CRC encryption types are available for MIT interoperability.
- Always enable the Policy Server and the Web Agent logs to record the authentication error messages.
- Verify that the name and location of keytab file for any host matches the ones specified in krb5.conf file.
- The SPN is always case-sensitive. The krb5.keytab files for the Policy Server and the web server host must contain the host and service principals for the Policy Server and web server hosts.
- Always verify the version of ktpass utility. The ktpass command tool utility is included in the Windows support tools and can be installed from MSDN download or Windows 2003 SP1 CD.
- Always confirm the encryption type used while creating the keytab file. Windows by default supports RC4-HMAC encryption.
- Confirm the Kerberos version number (kvno). The kvno of the service principal must match the kvno of its keytab file. The kvno number of any keytab is displayed when it is created.
- To determine the kvno of a service account:

  For Windows Active Directory:

  The kvno number of any service principal can be confirmed using ADSI Edit as follows:

  1. Run adsiedit.msc from command prompt.
  2. Go to service account, under CN=Users, DC=domain, DC=com under drop-down Domain (fqdn_ADhost) at the left.
  3. Right-click the service account and click properties.
  4. Verify the value of the msDs-KeyVersionNumber attribute, which matches the one shown while creating the keytab file for the service account.

  The kvno for any user account changes every time its password is changed. If the version numbers do not match, create an account and keytab, or change the password to match the kvno to the kvno of the keytab.

  For UNIX MIT KDC:

  Use the kvno utility. The syntax of this command is:

  ```
  Kvno principalname
  ```

- On Windows:

  You can verify if your keytab file is valid by installing windows support tools on both Policy Server and Web Agent running the following command:

  ```
  Kinit –k –t <keytab file location> <respective spn>
  ```

  For example:

  ```
  kinit –k –t C:\Windows\webserver.keytab HTTP/websvriis6.test.com@KRISH.COM
  ```

  This command returns no error when the keytab file is valid.

- On Solaris:

  Verify both keytab files validity by running the following commands:

  ```
  Kinit –k –t <keytab file> <SPN>
  ```

  For example:

  ```
  kinit –k –t host.keytab host/<fqdn>@DOMAIN.COM
  kinit –k –t smps.keytab smps/<fqdn>@DOMAIN.COM
  ```

  If you get no errors, keytab files are fine, and the krb.conf file has valid values.

  If you get an error, check whether the SPN is valid in KDC using the following command:

  ```
  Kinit host/<fqdn>@DOMAIN.COM
  ```

  This command usually asks for password. If you provide a valid password, you do not get an error message. If this command does not ask for password, the SPN was not identified. Check the property of that object, and in Account tab your SPN can be visible, for example, *host/fqdn*. If it is visible, make sure no other object has the same entry set as SPN.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y