CA SiteMinder® Web Access Manager

Policy Server Configuration Guide

r12 SP1

Third Edition

Contact CA

Contact Technical Support

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

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

Provide Feedback

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

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

CA Product References

This document references the following CA products:

- CA Single Sign-on
- CA SiteMinder® Web Access Manager
- CA SiteMinder® Web Access Manager Federation Security Services

Contents

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

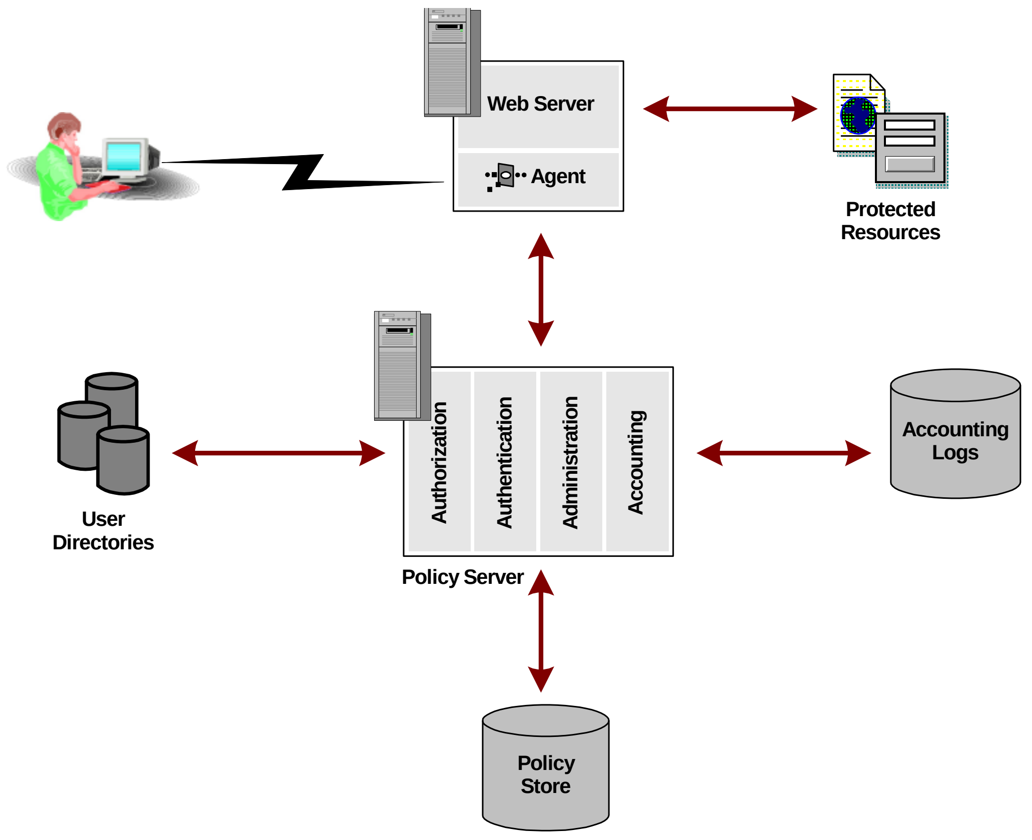
Accounting

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

Health Monitoring

The Policy Server provides features for monitoring activity throughout a 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 116)

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 (see page 29)
- Policy Domain Objects (see page 32)
- Global Objects (see page 33)

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, you must 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 popular set of resources.

Agent Configuration Objects

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

Host Configuration Objects

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

User Directories

A user directory in 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 groups of resources (realm), as well as the users who may access the resources, and the administrator who sets up entitlements.

Affiliate Domains

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

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

Administrators

An administrator is an object that contains profile information for a SiteMinder administrator account. Everyone who logs into 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 a user will need to access a protected resource. Authentication schemes are assigned to realms. When a user tries to access a resource in a realm, the authentication scheme of the realm determines the credentials that a user must supply in order to access the resource.

Registration Schemes

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

Agent Types

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

SQL Query Schemes

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

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

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

Trusted Hosts

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

Policy 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, etc. 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 will be allowed or denied for the resource. Rules can also include actions associated with specific events, such as what to do if a user fails to authenticate correctly when asked for their credentials.

Rule Groups

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

Responses

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

Response Groups

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

Policies

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

Variables

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

Affiliates

An affiliate object binds users, and optionally, time restrictions and IP address restrictions together. It also contains configuration data and a list of user entitlement attributes to be passed to an affiliate after a user is authenticated.

Note: More information on affiliates exists in the *Federation Security Services Guide*.

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

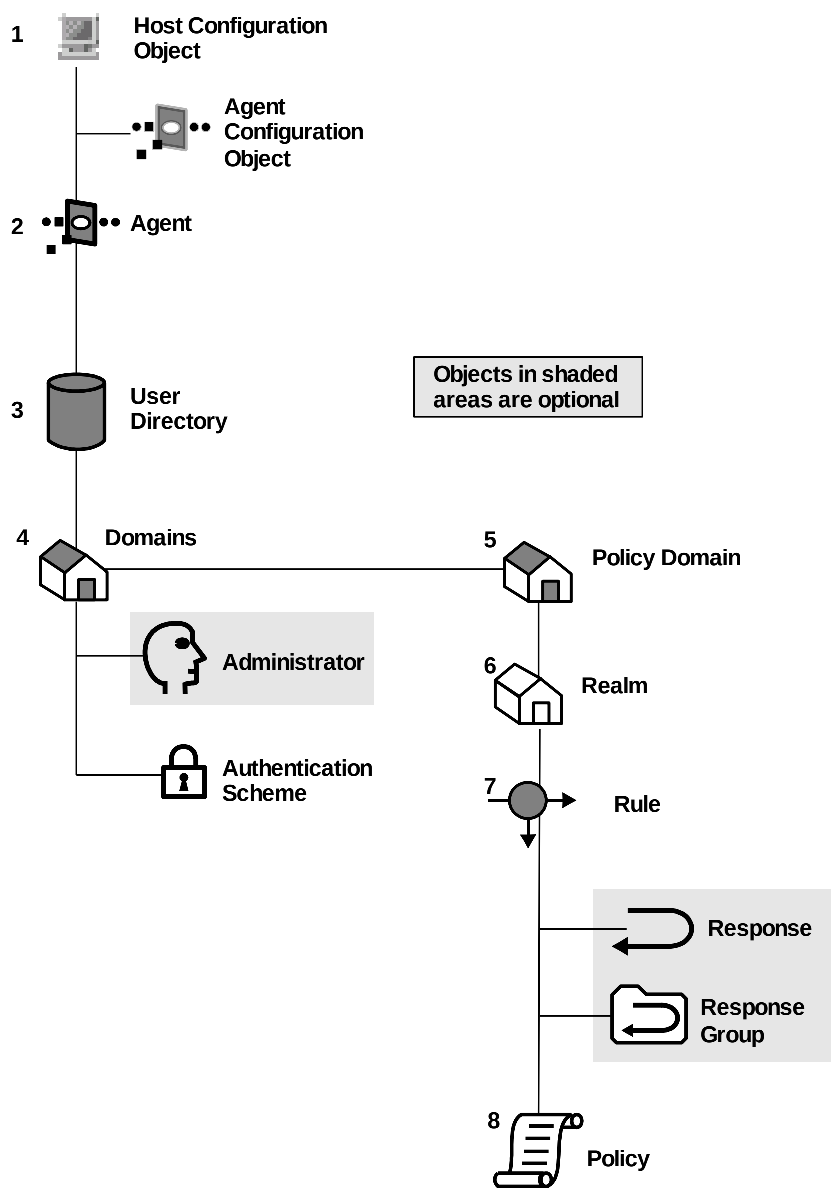
A global response is a Policy Server object that determines a reaction to a global rule. Global responses are included in global policies, and take place when a global rule is triggered.

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 should not have access.

Failing to manage users and secure resources effectively can have negative affects 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 can't 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.

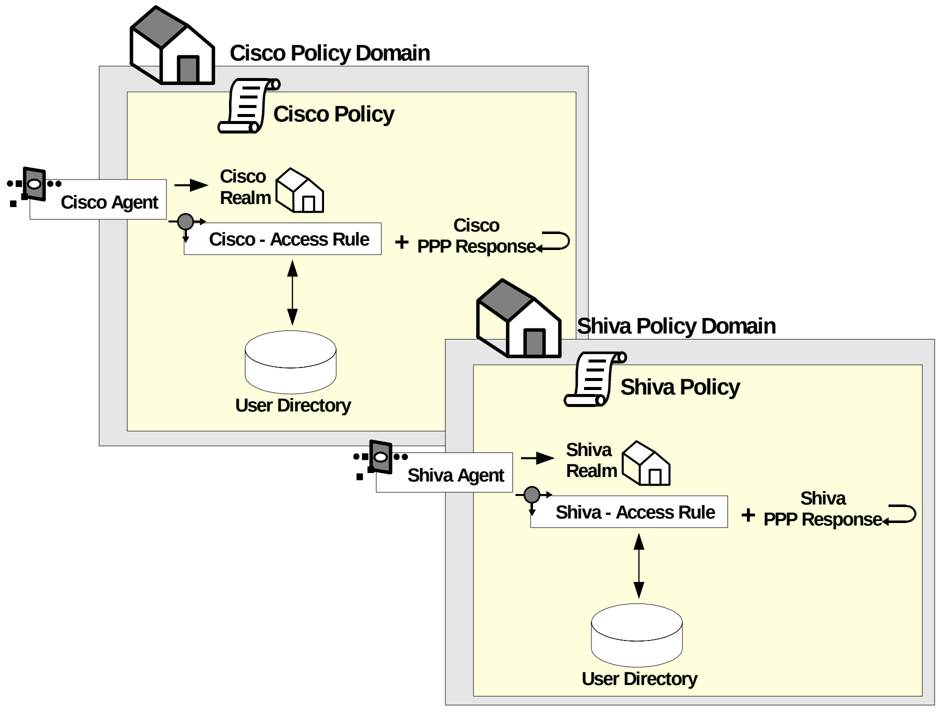
SiteMinder Security Policies

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:

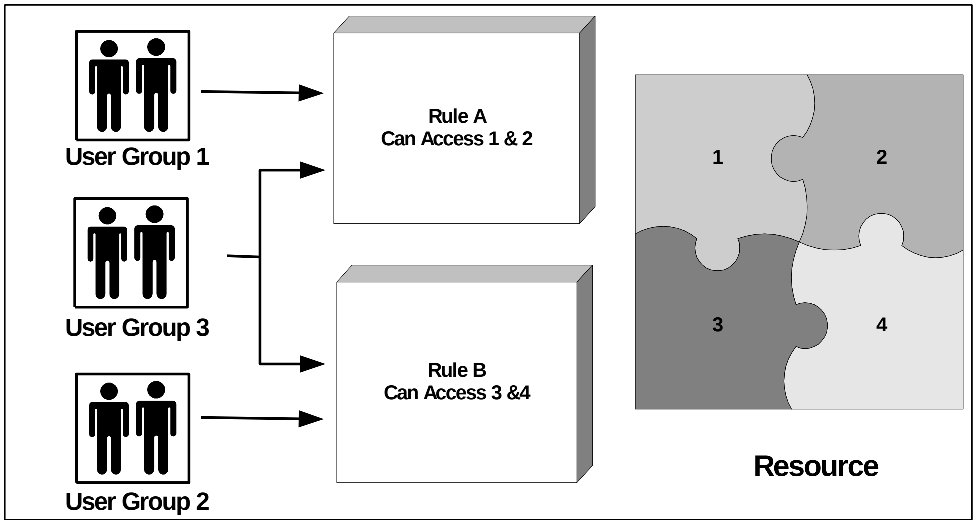
- How Privileges Are Established (see page 39)
- How Content Is Personalized (see page 39)
- How Sessions Are Managed (see page 40)

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:

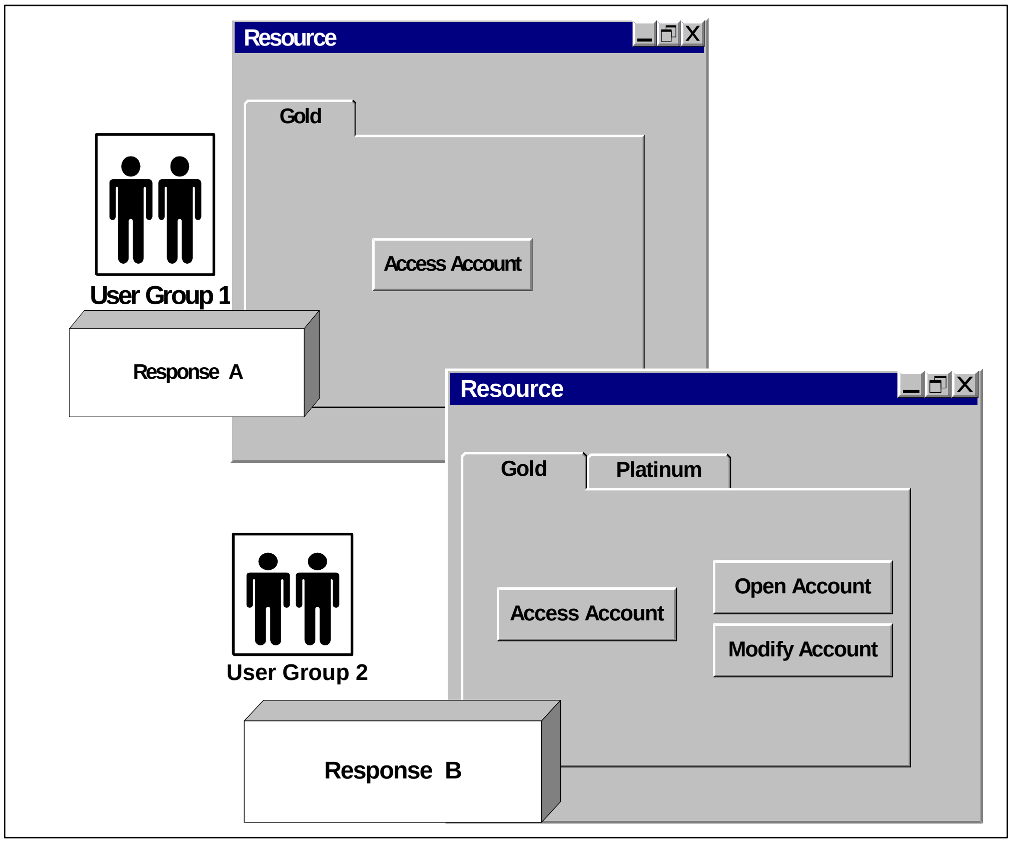
- Group 1 has been assigned Rule A and can access resources 1 & 2,
- Group 2 has been assigned Rule B and can access resources 3 & 4, and
- Group 3 has been assigned both rules and can access all resources.



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.

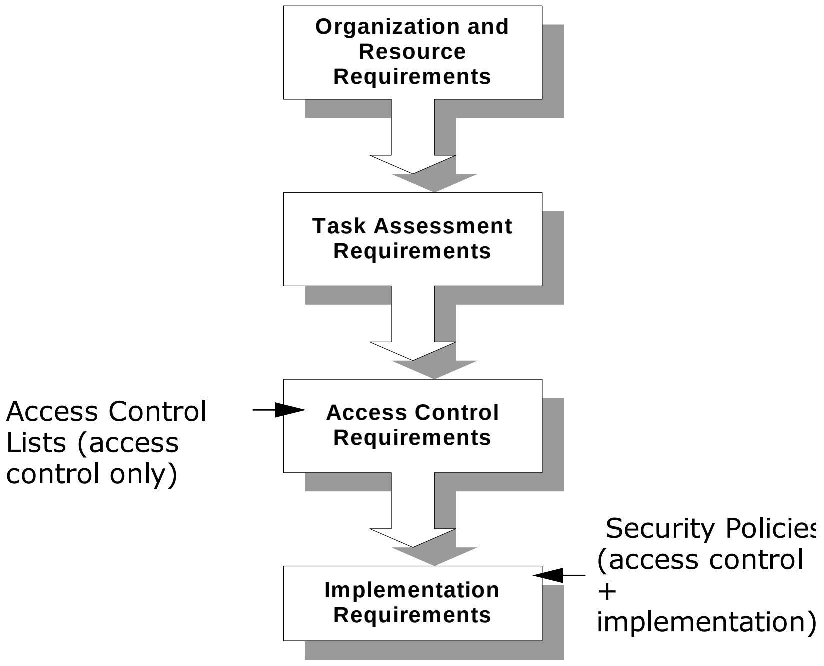
  You can also implement persistent sessions to provide Windows security context functionality and support for Federated Web Services.

More information:

How Privileges Are Established (see page 39)
How Content Is Personalized (see page 39)
How Persistent Sessions for User Security Contexts Are Maintained (see page 78)

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.


Organization and Resource Requirements
Task Assessment Requirements
Access Control Requirements
Implementation Requirements
Access Control Lists (access control only)
Security Policies (access control + implementation)

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 43)
Define Task-Assessment Requirements (see page 45)

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 from 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 46)
Organize Security Model Requirements (see page 43)
Organization and Resource Requirement Considerations (see page 43)
Define Implementation Requirements (see page 46)

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 A: 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 43)
Organization and Resource Requirement Considerations (see page 43)
Define Task-Assessment Requirements (see page 45)

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.

Chapter 4: Administrative User Interface Management

This section contains the following topics:

SiteMinder Administrative User Interfaces

Beginning with SiteMinder r12 SP1, there are two graphical user interfaces (UIs) that configure SiteMinder policy objects, as follows:

Administrative UI

The Administrative UI is a web-based administration console that is installed independent of the Policy Server. Use the Administrative UI to view, modify, and delete all Policy Server objects except those related to Federation Security Services. All federation-related configuration tasks should be handled using the FSS Administrative UI.

Federation Security Services Administrative UI (FSS Administrative UI)

The FSS Administrative UI is an applet-based application that is installed with the Policy Server. The federation-specific UI objects consist of affiliates (consumers, service providers, resource partners) and SAML authentication schemes that you configure to support federated communication between two partners.

The intent of the FSS Administrative UI is to let you manage SiteMinder Federation Security Services. If you are familiar with previous versions of the SiteMinder Policy Server User Interface, you will notice that all SiteMinder objects appear in the FSS Administrative UI, except the application objects for Enterprise Policy Management (EPM). You may use the FSS Administrative UI to manage these objects. If you need information while using the FSS Administrative UI, please consult the online help.

Important! You must register each UI with the Policy Server. Registering the FSS Administrative UI with the Policy Server ensures that the communication between both components is FIPS-encrypted (AES encryption). For more information about registering a UI, see the *Policy Server Installation Guide*.

Administrative UI Overview

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

The Administrative UI contains two panes:

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

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

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

Start the Administrative UI

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

To start the Administrative UI

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

   hostname

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

domain

Specifies the fully qualified domain name.

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

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

The log in screen appears.

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

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

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

More information:

Default Administrator Account for the Administrative UI (see page 56)

Manage Policy Server Objects

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

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

Duplicate Policy Server Objects

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

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

- Agent Type
- AuthAz Directory Mapping
- AuthValidate Directory Mapping

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

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

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

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

   Example: Click Infrastructure, Agent.

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

   The Create Object pane opens.

   Example:

   Click Agent, Create Agent.

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

   A list of objects that match the search criteria opens.

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

   The Create Object: *Name* pane opens.

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

   The Create Object task is submitted for processing.

View Policy Server Object Properties

You can view the properties of a Policy Server object.

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

To view the properties of an object

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

   Example: Click Policies, Domains.

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

   The View Object pane opens.

   Example:

   Click Domain, View Domain.

   The View Domain pane opens.

3. Specify search criteria, and click Search.

   A list of objects that match the search criteria opens.

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

   The View Object pane opens.

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

Modify an Existing Policy Server Object

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

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

To modify the properties of an existing object

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

   Example: Click Policies, Domains.

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

   The Modify Object pane opens.

   Example:

   Click Realm, Modify Realm.

   The Modify Realm pane opens.

3. Specify search criteria, and click Search.

   A list of objects that match the search criteria opens.

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

   The Modify Object: *Name* pane opens.

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

   The Modify Object task is submitted for processing.

More information:

Start the Administrative UI (see page 50)

Delete a Policy Server Object

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

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

To delete an object

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

   Example: Click Infrastructure, Authentication.

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

   The Delete Object pane opens.

   Example:

   Click Authentication Scheme, Delete Authentication Scheme.

   The Delete Authentication Scheme pane opens.

3. Specify search criteria, and click Search.

   A list of objects that match the search criteria opens.

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

   A confirmation pane opens.

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

5. Click Yes.

   The Delete Object task is submitted for processing.

More information:

Start the Administrative UI (see page 50)

SiteMinder Administrators

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

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

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

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

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

Administrator

An administrator can do the following:

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

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

Legacy Administrator

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

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

Default Administrator Account for the Administrative UI

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

Create an Administrator

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

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

Before you create an administrator, consider the following:

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

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

To create an administrator

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

   The Create Administrator dialog opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
2. Specify a Name and optionally, a description for the administrator you are creating in the General group box.
3. Enter a value for the User Path in the Details group box.

   You can either type a value for the user path or click Lookup to search for users in the current user store configured for the Administrative UI.
4. (Optional) Select Super User to grant all rights to the administrator and then click Submit. Otherwise, go to the next step.

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

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

6. Click on Create in the Rights group box

   The Select category dialog opens.

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

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

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

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

   d. Click Finish.

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

8. Click Submit.

You have defined an administrator.

Disable an Administrator Account

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

To disable an administrator account:

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

   The Search dialog opens.

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

   The Modify Administrator dialog opens.

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

The administrator is now disabled.

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

Administrator Use Case

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

This scenario involves three administrators

- SuperAdmin
- ManagerAdmin
- JuniorAdmin

Using the three administrators, the following scenarios are described:

- SuperAdmin creating the ManagerAdmin (see page 59)
- ManagerAdmin creating the JuniorAdmin (see page 60)

The following terms are used throughout the use case:

Access Method

Specifies how an administrator interacts with the Policy Server.

Security Category

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

Scope

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

Permissions

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

Rights

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

SuperAdmin Grants Privileges to ManagerAdmin

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

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

Initial Privileges for ManagerAdmin

The SuperAdmin initially assigns the following to ManagerAdmin:

Access Method

GUI Allowed

Rights

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

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

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

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

Additional Privileges for ManagerAdmin

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

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

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

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

ManagerAdmin's complete rights are now as follows:

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

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

ManagerAdmin Assigns Privileges to JuniorAdmin

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

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

- Application Administration
- Policy Administration

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

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

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

Access Method

GUI Allowed

Rights

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

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

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

JuniorAdmin's final rights are as follows:

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

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

Create a Legacy Administrator

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

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

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

To configure a legacy administrator

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

   The Create Legacy Administrator dialog opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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

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

   SiteMinder Database

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

   External Directory

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

4. Select the administrator privileges from the following options:

   System

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

   Domain

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

5. Choose the tasks the administrator can perform.

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

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

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

   - Manage Domain Objects
   - Manage Users
   - Manage Password Policies

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

You have defined a legacy administrator.

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

More information:

Legacy Administrator Privileges (see page 63)

Legacy Administrator Privileges

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

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

System Administrator Tasks

Tasks	**Administrative Privilege**
Manage System and Domain Objects	■ Create/edit/delete Agents, Agent Configuration Objects, Agent groups, Agent types, Host Configuration Objects, user directories, policy domains, authentication schemes, directory mappings, certificate mappings, registration schemes, and SQL query schemes. ■ All privileges for Manage Domain Objects.

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

Domain Administrator Tasks

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

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

Recreate a Deleted UI Administrator

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

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

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

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

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

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

To restore the Super User administrator:

1. Open a command prompt from the machine that is hosting the Policy Server, and enter the following command:

   ```
   xpsregclient client_name -adminui -su
   ```

   This command recreates the default Super User account.

   client name

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

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

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

   -adminui

   Specifies that an Administrative UI is being registered.

-su

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

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

2. Register the Admin UI with a Policy Server.

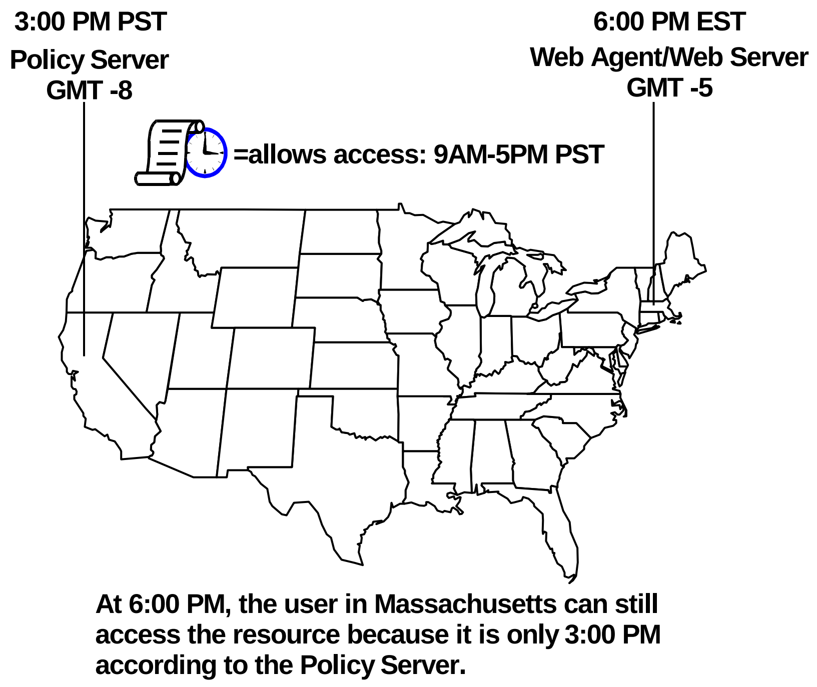
   Registering the Admin UI is an administration task in the Administrative UI. More information about this process can be found in the *Policy Server Installation Guide*.

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

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).


3:00 PM PST
Policy Server
GMT -8
6:00 PM EST
Web Agent/Web Server
GMT -5
=allows access: 9AM-5PM PST
At 6:00 PM, the user in Massachusetts can still access the resource because it is only 3:00 PM according to the Policy Server.

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.

Chapter 5: 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.

Persistent and Non-persistent Sessions

Standard SiteMinder sessions are non-persistent: they are maintained on the client side by writing a cookie containing session information to a user's browser. SiteMinder also provides the ability to configure persistent sessions to provide Windows security context functionality and support for Federated Web Services. Persistent sessions are configured on a per realm basis.

Persistent sessions should only be used where necessary since use of the session services to maintain sessions has an impact on system performance.

Note: More information about enabling and configuring session services exists in the Policy Server Administration Guide.

Session Tickets

SiteMinder implements session management using session tickets. A session ticket contains basic information about a user and that user's authentication information; it is used to identify the user's session across all sites in a single sign-on SiteMinder environment. Session tickets are encrypted and can only be read/validated by the Policy Server. 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.

Non-persistent session

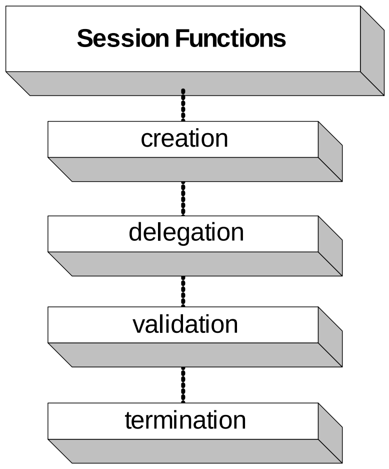
The Web Agent places the session ticket in a cookie. The cookie contains the user session data; no user-specific data is kept in the cookie itself. The Web Agent is responsible for validating the cookie and enforcing session timeouts.

Persistent Session

The Web Agent places the session ticket in a session server database and, if possible, in an optional cookie on the client. The session ticket data, whether retrieved from a cookie or the session server database, is used as an index into the Web Agent's cache, 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.

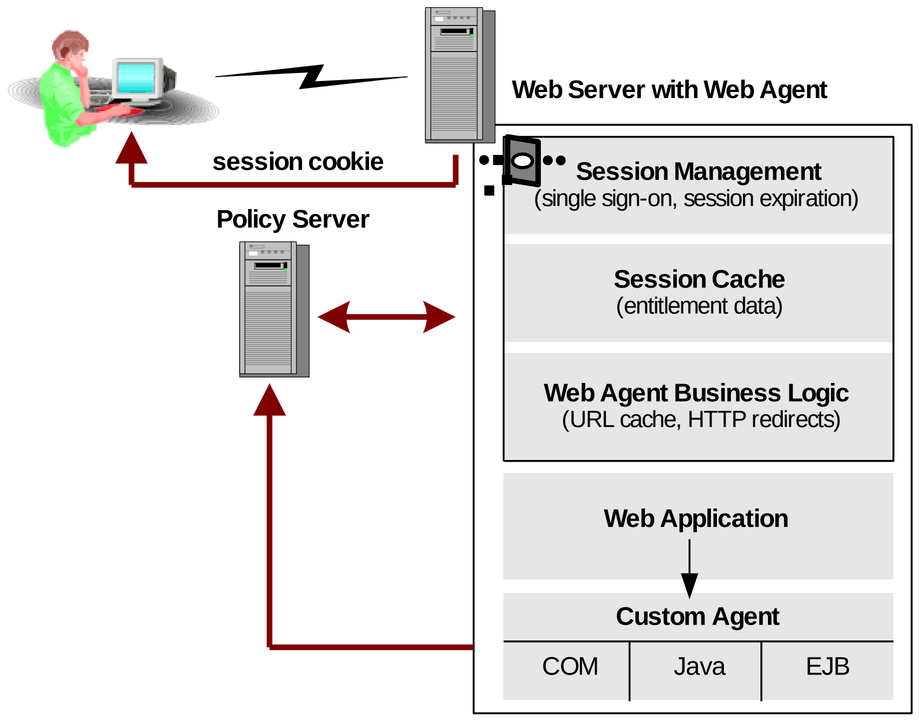
Session validation

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

Session termination

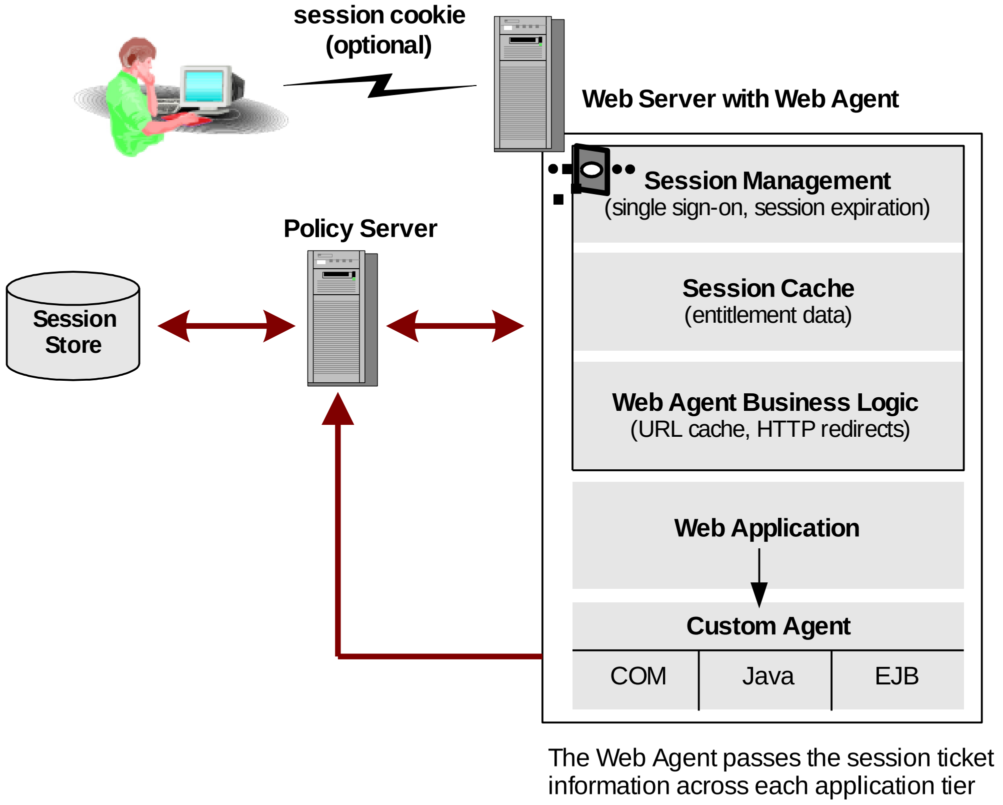
Ending a user session when a user logs out, when the configured session timeouts expire, or when a user is manually disabled by the SiteMinder System Manager. When a user logs out or the user session expires, they must log in again to create a new session. In the case of manual user disablement, the 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.


session cookie
(optional)
Web Server with Web Agent
Session Management
(single sign-on, session expiration)
Policy Server
Session Cache
(entitlement data)
Session
Store
Web Agent Business Logic
(URL cache, HTTP redirects)
Web Application
Custom Agent
COM
Java
EJB
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.

More information:

Protection Levels (see page 253)

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 247).

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.

More information:

Protection Levels (see page 253)

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's identity is passed from one application tier to another. After getting the session ticket, each application can make authorization calls to the Policy Servers.

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 that 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's 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. It delivers the user's identification automatically, regardless of the application. In addition, the ID is built into applications so that the applications have 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.

Name	Purpose
	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 377)
Advanced Policy Components for Applications (see page 507)

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's identity and authentication information. Web applications such as Microsoft Exchange Server or SQL Server need a user's security context to provide native security in the form of Microsoft's 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's 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's own security tools to protect applications.

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

  SiteMinder can work with Internet Explorer and Netscape Web browsers; it is not restricted to Internet Explorer.

- 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 pass on a user's 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, you need to enable persistent sessions for all realms that include those resources. Persistent sessions are maintained by the session server.

Note: For conceptual information about persistent sessions, see Persistent and Non-persistent Sessions (see page 69). For instructions on enabling persistent sessions for a realms protected by Web Agents, see Configure a Realm (see page 382).

The session server, which resides on the same system as the Policy Server, stores a user's encrypted credentials 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 Web Agent's user session cache becomes full and entries are purged, the Agent can retrieve the user's credentials from the session server and re-establish the session. The session server 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 server to revalidate a session. A session cookie stored in the user's browser contains the session ID, which the cookie uses to reacquire the user's credentials from the session server.

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

The frequency at which the Agent revalidates a session is determined by a configurable value called the *validation period*. The validation period defines how long the Agent can keep a session active using the information in its cache before contacting the session server for updates.

If the validation period is too small, the Agent goes back to the session server 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 Agent's maximum user session cache value, the Agent uses the cached information instead of contacting the session server.

Note: If the session server is not operating, the validation period is infinite and the Agent will not contact the session server.

For instructions on configuring the validation period, see Configure a Realm (see page 382).

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 119)
Associate one of the Supported Authentication Schemes (see page 80) with each realm	Resource group box of the Realm pane	Configure a Realm (see page 382)
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 382)
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 247)

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 6: 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. There are two operation modes: failover and round robin.

Failover

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

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

Round Robin

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

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

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

Implement an Operation Mode

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

To implement an operation mode

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

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

TCP/IP Connections

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

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

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

Maximum Sockets Per Port

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

Minimum Sockets Per Port

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

New Socket Step

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

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

Delete Trusted Host Objects

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

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

To delete a trusted host

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

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

   A list of trusted hosts that match the search criteria opens.

4. Select a trusted host from the list and click Select.

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

   You are prompted to confirm the deletion of the host.

5. Click Yes.

   The Trusted Host is deleted.

Host Configuration Objects for Trusted Hosts

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

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

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

Copy a Host Configuration Object

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

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

To copy a host configuration object

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

   The Create Host Configuration pane opens.

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

   The Create Host Configuration: *Name* pane opens.

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

   The Create Host Configuration task is submitted for processing.

Add Multiple Policy Servers to the Host Configuration Object

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

To add a Policy Server to a Host Configuration object

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

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

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

   The Modify Host Configuration: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
5. Click Add on the Clusters group box.

   The Add Cluster group box opens.

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

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

7. Click OK.

   The cluster is added to the Clusters group box.

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

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

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

9. Click Submit.

   The Modify Host Configuration task is submitted for processing.

More information:

Delete Trusted Host Objects (see page 86)
Operation Mode (see page 84)
Host Configuration Objects for Trusted Hosts (see page 87)

Configure Policy Server Clusters for a Host Configuration Object

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

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

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

To configure a cluster

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

   The Create Host Configuration pane opens.

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

   The Create Host Configuration: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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

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

6. Click Add on the Clusters group box.

   The Add Cluster group box opens.

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

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

8. Click OK.

   The cluster is added to the Clusters group box.

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

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

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

10. Click Submit.

    The Create Host Configuration task is submitted for processing.

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

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.

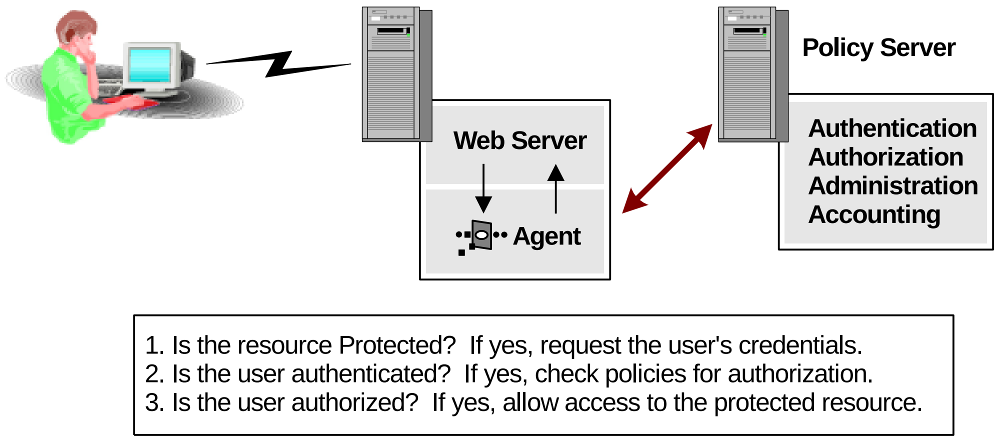
The Policy Server determines whether or not a user can be authenticated based on the credentials, and whether or not the user is authorized for the requested resource. The Policy Server then communicates with the Agent, which allows or denies access to the requested resource.

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 114)

SAML Affiliate Agents

The SAML Affiliate Agent is part of the Federation Security Services solution. Federation Security Services enables businesses to share security information across multiple domains.

Note: Federation Security Services and the SAML Affiliate Agent are packaged as separately-licensed items.

The SAML Affiliate Agent provides seamless single sign-on from a producer site, such as a portal, to a SAML consumer acting as an affiliate in a federated network. The affiliate provides resources and services related to the portal. For example, affiliateA.com and affiliateB.com have an agreement that visitors to affiliateA.com receive a 10% discount for purchases at affiliateB.com. These two sites are affiliates. Using the SAML Affiliate Agent eliminates the need for the a user to re-authenticate or provide additional information about themselves.

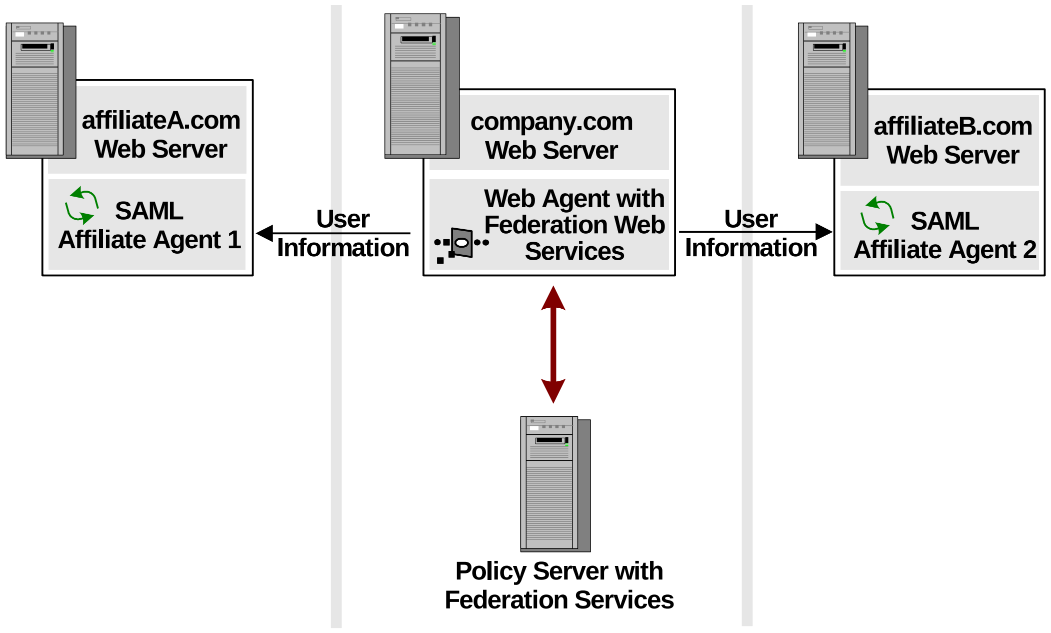
The communication exchange between a SAML Affiliate Agent and SiteMinder at the producer site results in the generation of a SAML assertion. A SAML assertion is an XML document containing authorization and authentication about a user. The producer sends this document to the SAML Affiliate Agent at the consumer site. The assertion confirms that a user has been authenticated at the main portal, and the information it contains enables the affiliate to provide user information to a Web server for use with its Web applications.

If you install a SAML Affiliate Agent at an affiliate site, it is the only SiteMinder component installed at that site. The affiliate site does not require a full installation, because a SAML Affiliate Agent does not protect resources; it only enables personalization.

The following components are required to support Federation Security Services at the producer site:

- Policy Server
- Web Agent
- Web Agent Option Pack-provides the Federation Web Services application

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


affiliateA.com Web Server
SAML Affiliate Agent 1
User Information
company.com Web Server
Web Agent with Federation Web Services
User Information
affiliateB.com Web Server
SAML Affiliate Agent 2
Policy Server with Federation Services

In the previous figure, the Policy Server is connected to a Web Agent on the company.com Web server. This Agent can pass user information about company.com users to the affiliateA.com and affiliateB.com Web servers using the SAML Affiliate Agents.

The producer site authenticates the user and passes information about the user to the affiliate site. This site has a full SiteMinder installation. The affiliate site has only a SAML Affiliate Agent and Web server; there is no Policy Server. The affiliate site does not require a full installation because a SAML Affiliate Agent does not protect resources in the same way as a Web Agent. The SAML Affiliate Agent simply provides user information to a Web server for use with its Web applications. The applications can use the information to personalize Web content and create a unique experience for each user.

SAML Affiliate Agent and Federation Security Services Configuration

Configuration of the SAML Affiliate Agent happens at the consumer site—you do not configure a SAML Affiliate Agent at the producer-site Policy Server. However, in the Policy Server User Interface, you set up Federation Security Services components, create affiliate domains, and configure affiliates for the affiliate domains.

Note: In the Policy Server User Interface, there is an option to create an Agent and choose Affiliate Agent as the Agent type. This option is not for the SAML Affiliate Agent—it is only applicable for 4.x Affiliate Agents. Do not select it. More information exists in the *Federation Security Services Guide*.

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 673)

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 96)
Combined Central and Local Configuration (see page 101)

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 95)

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 95)

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 87)
Legacy Administrator Privileges (see page 63)
Create an Agent Object to Establish a Web Agent Identity (see page 102)
Agent Configuration Object Overview (see page 106)
Enable a Web Agent (see page 111)

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, Create Host Configuration.

   The Create Host Configuration pane opens.
3. Do one of the following:

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

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

   The Create Host Configuration: *Name* pane opens.

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

   The Create Host Configuration task is submitted for processing.

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, you must create an Agent object in the Administrative UI. The object name must match the Agent name in the AgentName or DefaultAgentName parameter in the Agent configuration file or Agent Configuration Object. The Policy Server uses the Agent identity to map the Agent name to the IP address of the 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 you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To create a Web Agent object and identity

1. Click Infrastructure, Agent, Create Agent.

   The Create Agent pane opens.

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

   The Create Agent: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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

   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.

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

5. Click Submit.

   The Create Agent Task is submitted for processing.

More information:

Realms (see page 377)
Start the Administrative UI (see page 50)

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

To create a 4.x Web Agent identity, you must 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 regarding the end date for 4.x Web Agent support.

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

To create a 4.x Web Agent object and identity

1. Click Infrastructure, Agent, Create Agent.

   The Create Agent pane opens.

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

   The Create Agent: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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

   Limits:

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

4. Confirm the following:

   - That the SiteMinder radio button is selected.
   - That Web Agent appears in the Agent Type drop-down list.

5. Select the Supports 4.x agents check box

   The Trust Settings group box appears.

6. In the IP Address field, type 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.

7. Type and confirm a shared secret in the respective fields.

   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.

8. Click Submit.

   The Create Agent Task is submitted for processing.

More information:

Start the Administrative UI (see page 50)
Realms (see page 377)

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 106)

Agent Configuration Object Overview

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

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

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 108)

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. Once the new Agent configuration object is created, you can modify its list of parameters and their values. For more information about parameters and default settings, see the *Web Agent Configuration Guide*.

To copy an Agent Configuration Object

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

   The Create Agent Configuration: Search pane opens.

2. Select the Create a copy radio button, and do one of the following:

   ■ Select one of the default Agent Configuration Objects from those listed.

   ■ Click Search and choose from the list of Agent Configuration Objects that is displayed.

3. Click OK.

   The Create Agent Configuration: *Name* pane opens.

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

   The Create Agent Configuration Task is submitted for processing.

More information:

Start the Administrative UI (see page 50)

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.

To create an Agent Configuration object

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

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

- Create a new Agent Configuration object.

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

To create an Agent Configuration Object

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

   The Create Agent Configuration: Search pane opens.

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

   The Create Agent Configuration: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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

4. Click Add.

   The Create Parameter group box opens.

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

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

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

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

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

   The Create Agent Configuration Task is submitted for processing.

Required Agent Configuration Object Parameters

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

- All Agents require DefaultAgentName
- 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: More information on parameters and default settings exists in the *Web Agent Configuration Guide*.

To modify an Agent configuration object

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

   The Modify Agent Configuration: Search pane opens.

2. Specify search criteria, and click Search.

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

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

   The Modify Agent Configuration: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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

   The Edit Parameter group box opens.

5. Edit the parameter's value.

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

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

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

7. Click OK.

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

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

8. Click Submit.

   The Modify Agent Configuration Task is submitted for processing.

Enable a 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 Agent Identities for Non-Web Agents

All SiteMinder Agents require an Agent identity. The Policy Server uses the Agent identity to map the Agent name to the IP address of each Web, RADIUS, or application server instance hosting an Agent. The Policy Server uses the Agent identity to associate policies with the correct Agent.

You can configure an Agent identity by creating an Agent object in the Administrative UI. In the Agent object, you can assign a name to the Agent and define its agent type—Web Agent, EJB Agent, Servlet Agent, RADIUS Agent, Custom Agent.

Note: An Affiliate Agent is one of the Agent types for an Agent object—this Agent type applies only to 4.x Affiliate Agents. Do not use this option to configure SAML Affiliate Agents.

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

To configure an Agent object

1. Click Infrastructure, Agent, Create Agent.

   The Create Agent pane opens.

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

   The Create Agent: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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

   Limits:

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

4. Specify the Agent settings in the Agent Type group box.
5. Complete any remaining fields based on the appropriate instructions:
   - For Web Agents, read Create an Agent Object to Establish a Web Agent Identity (see page 102).
   - For RADIUS Agents, read Configure a RADIUS Agent (see page 112).
   - For Custom Agents, read Create Custom Agents (see page 116).
   - For EJB and Servlet Agents, see the appropriate SiteMinder Application Server Agent guide for WebLogic or WebSphere application servers.

More information:

Start the Administrative UI (see page 50)

Configure a RADIUS Agent

To create a RADIUS Agent identity, you must 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 only configure RADIUS Agents locally. More information exists in the *Web Agent Installation Guide*.

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

To configure a RADIUS Agent

1. Click Infrastructure, Agent, Create Agent.

   The Create Agent pane opens.

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

   The Create Agent: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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

   Limits:

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

4. Click the Radius radio button, and then select a RADIUS vendor from the Agent Type drop-down list.

   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.

   The Trust Settings and Radius Settings group boxes open.

5. Add the following in the Trust Settings group box:

   - Type an IP Address of the RADIUS client (NAS device) in the field.
   - Type and confirm an alphanumeric shared secret in the respective fields .

6. Type the number of the RADIUS realm hint in the Realm Hint field on the Radius Settings group box.

   Note: More information about realm hints exists in the *Policy Server Administration Guide*.

7. Click Submit.

   The Create Agent Task is submitted for processing.

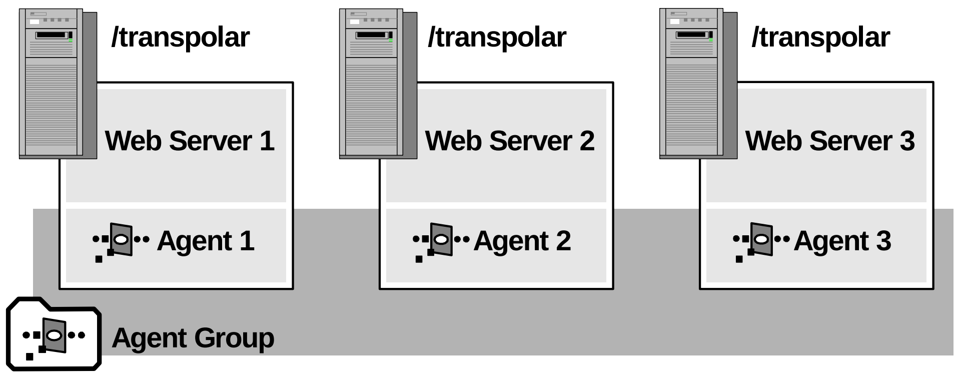
More information:

Start the Administrative UI (see page 50)
Configure a Realm Protected by a RADIUS Agent (see page 384)

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 439)

Configure an Agent Group

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

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

To configure an Agent group

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

   The Create Agent Group pane opens.

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

   The Create Agent Group: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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

4. Select the Agent Style and Agent Type on the Agent Type group box.

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

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

   The Choose agents pane opens.

   Note: Only Agents of the specified Agent type are listed in the Available Members column. 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.

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

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

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

7. Click OK.

   The selected Agents are added to the Agent group.

8. Click Submit.

   The Create Agent Group Task is submitted for processing.

More information:

Start the Administrative UI (see page 50)

Add Agents to an Agent Group

You can add existing Agents to an Agent group.

To add Agents to an Agent group

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

   The Modify Agent Group pane opens.

2. Specify search criteria, and click Search.

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

3. Select an Agent group, and click Select.

   The Modify Agent Group: *Name* pane opens.

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

   The Choose agents pane opens.

   Note: Only Agents of the specified Agent type are listed in the Available Members column. 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.

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

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

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

6. Click OK.

   The selected Agents are added to the Agent group.

7. Click Submit.

   The Modify Agent Group Task is submitted for processing.

Custom Agents

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

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

To create an Agent object

1. Click Infrastructure, Agent, Create Agent.

   The Create Agent pane opens.

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

   The Create Agent: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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

   Limits:

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

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

   The IP Address and Shared Secret group boxes open.

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

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

   Limits:

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

7. Click Submit.

   The Create Agent Task is submitted for processing.

More information:

Configure Agent Identities for Non-Web Agents (see page 111)
Configure a Custom Agent Type (see page 117)
Start the Administrative UI (see page 50)

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.

Chapter 7: 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
- Windows NT
- Custom

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

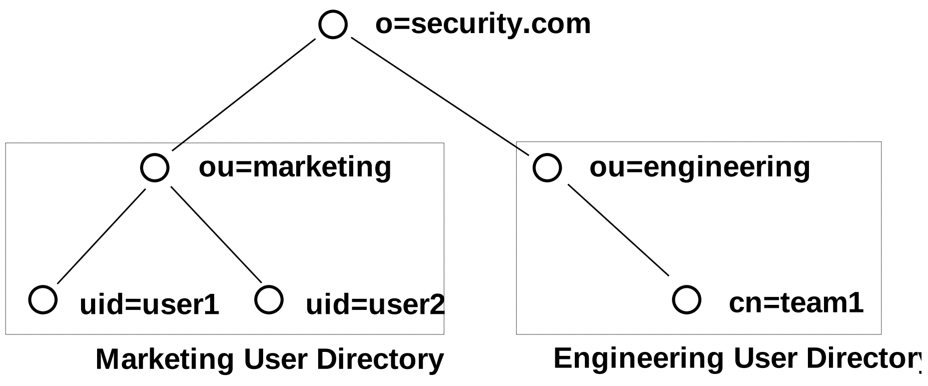
Note: If you are trying to configure/upgrade a store using a directory server or database listed in the Support Matrix and cannot find the procedures in this guide, refer to the *Directory Configuration Guide*.

LDAP Overview

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

General Information About LDAP

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



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

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

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

User Disambiguation in an LDAP Directory

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

- DN
- Search expression

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

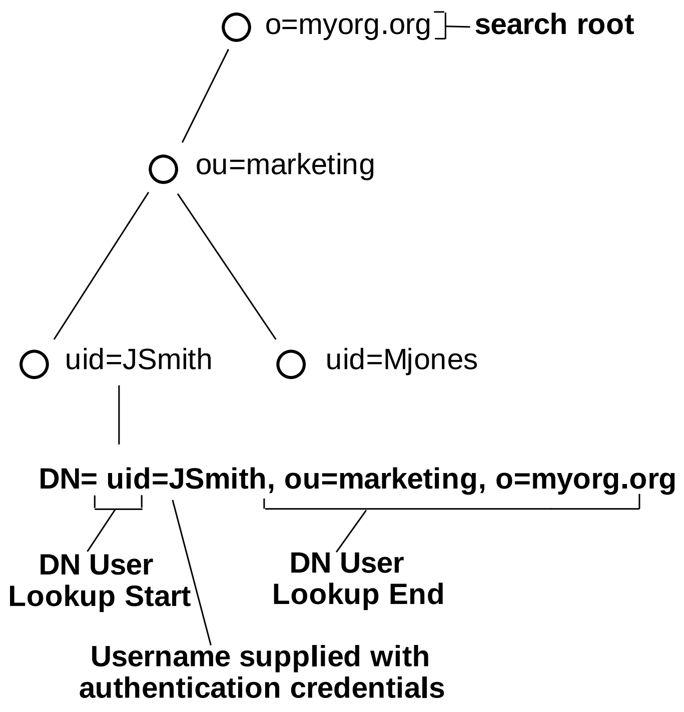
User Lookup by DN

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

The resulting DN has the following format:

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

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



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

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

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

User Lookup via a Search Expression

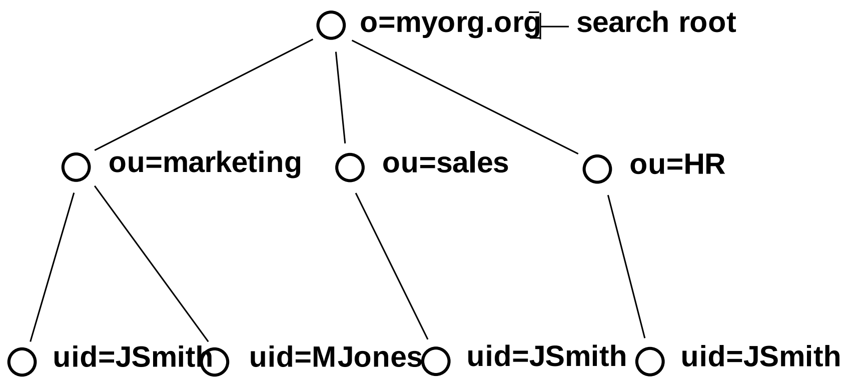
An LDAP directory server may contain numerous users in complicated DITs, and it may not be practical to create a large number of user directory connections. Your organization may have hundreds of organizational units and you may want to avoid having end users log in with detailed string representations.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

Example: Search expressions for user DN lookups

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

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

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



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

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

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

LDAP Search Filters

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

Search Filter	Format	Description
Equality	attribute=value For example, to find a user whose user ID is jsmith, the search filter is uid=jsmith.	This filter finds a specific value for an attribute in an LDAP directory.

String Matching	attribute=*value, OR
	attribute=value*, OR
	attribute=val*ue, OR
	attribute=*value*
	For example, uid=*smith matches all values that end in smith, such as jsmith, msmith, etc. A value of uid=*smith* matches jsmith, msmith, and bsmithe.
		LDAP search filters support wild cards, which allow you to search for an attribute value based on a partial string. To find all of the values that match a partial string, use the wildcard character (*).

Greater than or equal to	attribute>=value For example, to find all of the users in a directory who are age 21 or over, part of the search filter would be age>=21.	This filter finds values that are greater than or equal to the specified value.
Less than or equal to	attribute<=value For example, to find all of the users in a directory who are age 21 or younger, part of the search filter would be age<=21.	This filter finds values that are less than or equal to the specified value.

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

Objectclass Searches

Each entry in an LDAP table has at least one objectclass attribute. You can use a presence filter in conjunction with the objectclass attribute to build filters for searching your LDAP user directories. In 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\LDAPProvide
r\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 129)

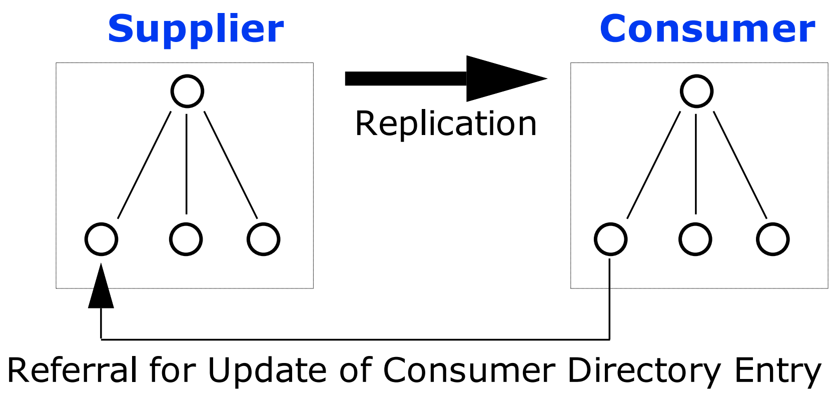
Directory Topology and LDAP Referrals

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:

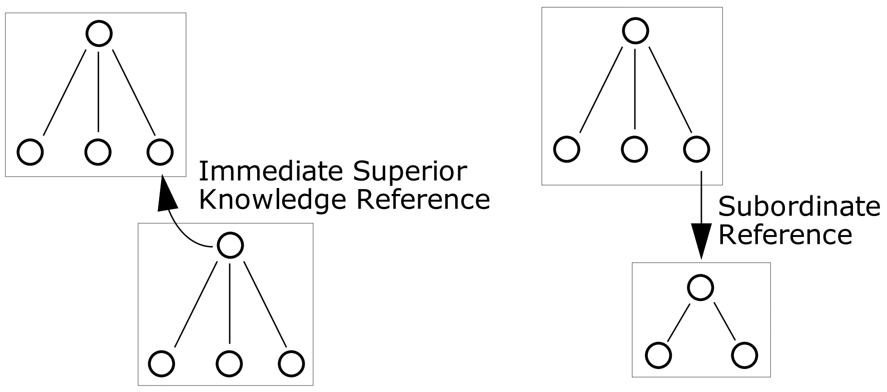
Replication Agreements

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.



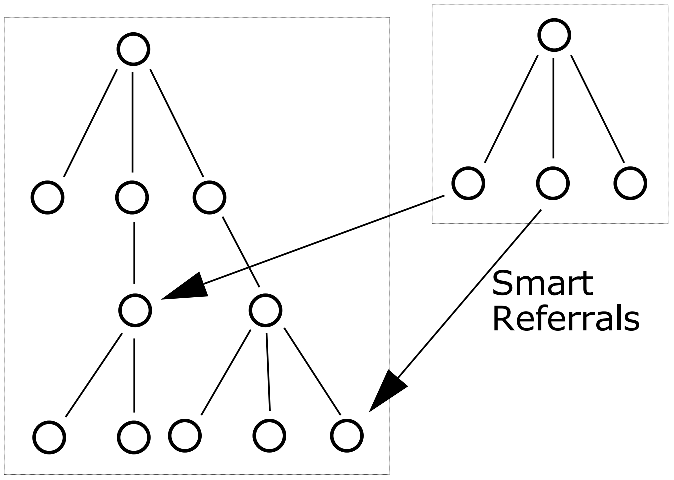
Knowledge References

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.

More information:

Configure a SQL Query Scheme (see page 186)

Windows Directory Overview

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

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

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

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

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

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

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

WinNT user names and passwords can be used as credentials.

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

Active Directory Overview

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.

Note: For information about using the LDAP namespace for an Active Directory user store, see Configure Active Directory Connections (see page 157).

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.
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.

Attribute Name	Data Type	Directory Types	Description
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 197)
Password Policies (see page 589)
Anonymous Authentication Schemes (see page 252)
User Attribute Mapping (see page 212)

How to Configure a CA Directory User Directory Connection

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

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

Ping the User Store System

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

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

Configure CA Directory User Directory Connections

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

To configure the user directory connection

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

   The Create User Directory pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Select LDAP from the Namespace list.

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

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

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

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 181)
Define an Attribute Mapping (see page 216)

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. You must configure CA Directory to handle these requests or CA Directory runs out of connections and authentication fails.

To enable user store DSA parameters

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

   ```
   #SiteMinder

   mimic-netscape-for-siteminder = true;

   concurrent-bind-user = true;

   hold-ldap-connections = true;
   ```
3. Save and close the .dxc file.

   The user store DSA parameters are enabled.

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 Sun Java System User Directory Connection

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

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

Ping the User Store System

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

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

Configure Sun Java System User Directory Connections

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

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

To configure the user directory connection

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

   The Create User Directory pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Select LDAP from the Namespace list.

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

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

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

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 181)
Define an Attribute Mapping (see page 216)
How to Configure an LDAP User Directory Connection over SSL (see page 173)

How to Configure a IBM Directory Server User Directory Connection

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

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

Ping the User Store System

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

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

Configure IBM Directory Server User Directory Connections

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

To configure the user directory connection

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

   The Create User Directory pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

3. Select LDAP from the Namespace list.

   LDAP settings open.

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

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

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

   The Create Attribute Mapping pane opens.

9. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 181)
Define an Attribute Mapping (see page 216)
How to Configure an LDAP User Directory Connection over SSL (see page 173)

How to Configure a Domino User Directory as a User Store

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

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

Verify that a Domino User Directory Meets Policy Server Requirements

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

- The Domino user directory must be version 5.02 or greater.

- The Domino user groups must share the root DN that you specify when configuring the connection from the Policy Server to the Domino user store.

  Example: When adding the group *marketing/myorg.org* to the address book (names.nsf) in Lotus Notes, you must type *o=myorg.org* in the Root field on the User Directory pane.

- Each new user must have both a user name entry and an Internet password entry in the Domino user directory.

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

Ping the User Store System

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

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

Configure Domino Directory Connections

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

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

To configure the user directory connection

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

   The Create User Directory pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Select LDAP from the Namespace list.

   LDAP settings open.

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

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

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

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

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

   Note: The search strings that you specify in the User DN Lookup Start and End fields must adhere to proper LDAP notation, not the Lotus Notes shorthand notation. More information about search strings exists in LDAP Search Filters.

6. (Optional) Click Configure to configure load balancing and failover.

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

7. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.

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

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

   The Create Attribute Mapping pane opens.

10. Click Submit.

    The Create User Directory task is submitted for processing.

More information:

User Disambiguation in an LDAP Directory (see page 121)
LDAP Load Balancing and Failover (see page 181)
Directory Attributes Overview (see page 133)
Define an Attribute Mapping (see page 216)
How to Configure an LDAP User Directory Connection over SSL (see page 173)

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 133)

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 (see page 147).

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.

To configure the user directory connection

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

   The Create User Directory pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

3. Select LDAP from the Namespace list.

   LDAP settings open.

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

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

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

   Note: If the user directory contains multiple organizations, you can leave the Root field blank. This lets the Policy Server search for users in multiple organizations.

6. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.

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

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

   The Create Attribute Mapping pane opens.

9. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 181)
Define an Attribute Mapping (see page 216)
How to Configure an LDAP User Directory Connection over SSL (see page 173)

How to Configure an ADAM User Directory Connection

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

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

Create an ADAM Instance and User Store Connection

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

To create an ADAM instance and user store connection

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

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

Create a SiteMinder User For Connecting to the ADAM User Store

You can create a SiteMinder user in the Connection Settings dialog shown in the previous topic.

To create a SiteMinder user for connecting to the ADAM user store

1. Under o=userstore, right-click CN=Roles and select New, Object.
2. In the Create Object dialog:
   a. Select user and click Next.
   b. In the Value field, enter a user name that SiteMinder uses to connect to the ADAM user store and click Next. This example uses admin.
   c. Click Finish.

3. In the right column of ADAM ADSI Edit, right-click CN=admin and select Reset Password.
4. In the Reset Password dialog, enter and confirm the new password and click OK.

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

Assign Administrator Privileges to the SiteMinder User

You use ADAM ADSI Edit to assign administrator privileges to SiteMinder users.

To assign administrator privileges to the SiteMinder user

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

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

Load Users into the ADAM User Store

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

To load users into the ADAM user store

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

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

   user store

   Specifies the name of your user store.

   IP address

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

   port number

   Specifies the port number on which ADAM is listening.

Configure ADAM User Store Directory Connections

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

To configure the user directory connection

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

   The Create User Directory pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Select LDAP from the Namespace list.

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

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

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

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

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 181)
Define an Attribute Mapping (see page 216)
How to Configure an LDAP User Directory Connection over SSL (see page 173)

How to Configure an Active Directory Directory Connection

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

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

Active Directory Considerations

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

Windows deployments

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

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

the resulting Windows ID is <server>\<username>. IIS requires that <username> be the same as the Windows user ID, and that <server> be the logon domain name.

Multi-byte Character Support

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

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

Authentication against an AD namespace

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

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

HKLM\Software\Netegrity\SiteMinder\CurrentVersion\Ds\LDAPProvider\EnableSaslBind

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

Administrator Credentials

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

LDAP Search Root Configuration

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

For example, if you have created five users through the AD console at DN ou=People,dc=clearcase,dc=com and locked two of those users. The SiteMinder User Management dialog will show locked users disabled only if you configure the LDAP search root as the DN of the AD domain (that is, dc=clearcase,dc=com). If you configure the LDAP search root as ou=People,dc=clearcase,dc=com, the locked users will incorrectly be shown as enabled.

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/LDAPProvider
>
> **Values:** 0 (disabled) or 1 (enabled)
>
> **Default:** 0. If the registry key is disabled, the default behavior is in effect.

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

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 126)

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.
2. Click User Directory, Create User Directory.

   The Create User Directory pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Select either AD or LDAP from the Namespace list on the Directory Setup group box.

   LDAP settings open.

   Note: Because Microsoft Active Directory is an LDAP-compliant user directory, you can configure an Active Directory connection using the AD namespace or the LDAP namespace.
4. Complete the remaining required connection information on the General and Directory Setup group boxes.

   Note: Consider the following:

   – For more information about an authenticated user's security context, see How a Windows User Security Context is Obtained.

   – If you plan to use an SSL connection from the Policy Server to an Active Directory namespace, you must specify the FQDN and port number in the Server field on the Directory Setup group box. When the FQDN is not specified, an error is logged that states the user directory cannot be contacted. A Windows Event is also logged that reports the certificate does not match the server name.

   Note: If the Policy Server is operating in FIPS mode and the directory connection is to use a secure SSL connection when communicating with the Policy Server, the certificates used by the Policy Server and the directory store must be FIPS compliant.
5. (Optional) Click Configure on the Directory Setup group box to configure load balancing and failover.

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

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

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

7. Type the LDAP Search and LDAP User DN Lookup settings in the fields on the LDAP Settings group box.
8. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder's use in the fields on the User Attributes group box.
9. (Optional) Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.

10. Click Submit.

    The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 181)
Directory Attributes Overview (see page 133)
Define an Attribute Mapping (see page 216)
How to Configure an LDAP User Directory Connection over SSL (see page 173)

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.

To configure the user directory connection

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

   The Create User Directory pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Select LDAP from the Namespace list.

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

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

   Note: More information on load balancing and failover exists in LDAP Load Balancing and Failover.
7. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.
8. (Optional) Specify the user directory profile attributes that are reserved for SiteMinder's use in the fields on the User Attributes group box.
9. (Optional) Click Create on the Attribute Mapping List group box.

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

    The Create User Directory task is submitted for processing.

More information:

Define an Attribute Mapping (see page 216)
How to Configure an LDAP User Directory Connection over SSL (see page 173)

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: You can click Help for a description of fields, controls, and their respective requirements.
3. Select LDAP from the Namespace list.

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

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

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

   The Create Attribute Mapping pane opens.

9. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

LDAP Load Balancing and Failover (see page 181)
Define an Attribute Mapping (see page 216)
How to Configure an LDAP User Directory Connection over SSL (see page 173)

How to Configure an ODBC User Directory Connection

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

1. Ping the User Store System
2. Configure the ODBC Directory Connection

Ping the User Store System

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

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

Configure ODBC Directory Connections

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

If you are using a SQL database for audit logs and caching is turned on, under heavy load, 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.
2. Click User Directory, Create User Directory

   The Create User Directory pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

3. Select ODBC from the Namespace list.

   ODBC settings open.

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

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

5. Select a SQL query scheme.
6. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator who has an account on the user directory in the fields on the group box.

   Note: The user name must match the user who owns the tables containing user directory data. For example, if you are using the SmSampleUsers schema, this user must be the owner of the SmUser, SmUserGroup, and SmGroup tables. The administrator's account must have read or read/write privileges for the user directory.

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

   The Create Attribute Mapping pane opens.

9. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

SQL Query Schemes (see page 186)
Configure ODBC Data Source Failover (see page 186)
Define an Attribute Mapping (see page 216)

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 Windows Directory Connection

You can use a Windows Directory as a user store. User directory connections can be set up with any of the following WinNT implementations:

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

The following process lists the steps for creating the user directory connection in the Policy Server.

1. Meet the WinNT Domain Connection Requirements
2. Ping the User Store System
3. Configure the Windows Directory Connection

WinNT Domain Connection Requirements

For the Policy Server to connect to your WinNT domain, it must meet the following requirements:

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

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

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

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

The resulting Windows ID is <server>\<username>. IIS requires that <username> be the same as the Windows user ID, and that <server> be the logon domain name.

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

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

WinNT user names and passwords can be used as credentials.

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

Ping the User Store System

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

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

Configure a Windows Directory Connection

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

To configure the directory connection

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

   The Create User Directory pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Select WinNT from the Namespace list.

   WinNT settings open.
4. Complete the remaining required connection information on the General and Directory Setup group boxes for the WinNT domain, a computer within the WinNT domain, or a stand-alone WinNT computer.
   - WinNT domains

     To define a namespace that represents global users and groups in a WinNT domain, specify the name of the domain in the NT Domain Name field. SiteMinder supports domain authentication and trusted domain authentication for user accounts in domains for user accounts in domains.

     Note: The system on which the Policy Server is installed must have a computer account in the appropriate domain for domain authentication to work. If the system does not have a computer account in all domains in which users need to be authenticated, the appropriate trust relationships must be established between domains.

     You can change the NT account under which the Policy Server is running by opening the Services dialog from the Control Panel and changing the account under which the Policy Server is running to an account that has the necessary privileges to access the specified domain. This account must have the "Act As Part of Operating System" system privilege.

- Individual WinNT computer in a domain

  To define a Namespace that represents local users and groups in a computer that is a member of a domain, specify the name of the domain followed by the name of the computer in the NT Domain Name field. The domain name and computer name must be separated by a forward slash (/). If you do not specify the name of the computer, performance during searches may suffer.

  Example: SampleDomain/Comp1

- Stand-alone WinNT computer

  To define a namespace that represents local users and groups in a stand-alone computer, specify the name of the computer in the NT Domain Name field. The stand-alone computer must have a Policy Server installed on it in order for this namespace to be accessible. There may be an initial delay when the Policy Server accesses this namespace for the first time.

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

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

   The Create User Directory task is submitted for processing.

More information:

Directory Attributes Overview (see page 133)
Windows User Security Context Requirements (see page 79)
Define an Attribute Mapping (see page 216)

LanMan User Directories

A LanMan user directory connection is an alternative method for distributing directory requests in a WinNT environment. The LanMan user directory connection option allows you to specify a failover list of Backup Domain Controllers (BDCs) used for each user directory lookup in the WinNT Registry. This list of BDCs takes precedence over the Primary Domain Controller (PDC), which handles directory requests in most WinNT deployments. Using a LanMan directory connection, the Policy Server sends WinNT directory requests to the first active BDC in the Registry list, rather than forcing requests to pass through the PDC.

More information:

LanMan User Directories (see page 651)

Failover for WinNT User Directories

Windows NT LanManager automatically provides failover for WinNT user directories by defining a list of Primary Domain Controllers (PDCs) and Backup Domain Controllers (BDCs). SiteMinder does not contain additional features for WinNT directory failover. To configure failover, use Windows NT LanManager.

Note: More information about LanManager exists in the Windows NT documentation.

WinNT Attributes in Responses

For the purpose of delivering information back to a Web application via responses, SiteMinder has access to the following default user attributes:

- Description
- FullName
- AccountExpirationDate
- UserFlags
- LoginWorkstations
- BadPasswordAttempts
- MaxLogins
- MaxStorage
- PasswordExpired
- LastLogin
- LastLogoff
- HomeDirectory
- Profile
- LoginScriptMinPasswordLength
- MaxPasswordAge
- MinPasswordAge
- PasswordHistoryLength

These user attributes can be accessed through the User Attribute feature on a Response pane. A response can return the value of any of these attributes to a SiteMinder Agent. For Web implementations, SiteMinder returns these attributes as name/value pairs in HTTP header variables.

More information:

Responses and Response Groups (see page 413)

How to Configure a Custom User Directory Connection

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

1. Ping the User Store System
2. Configure the Custom Directory Connection

Ping the User Store System

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

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

Configure Custom Directory Connections

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

To configure the directory connection

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

   The Create User Directory pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create User Directory: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select Custom from the Namespace list.

   Custom settings open.

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

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

6. (Optional) Select Require Credentials on the Administrator Credentials group box, and type the user name and password of an administrator's account on the user directory in the fields on the group box.

   Note: The Policy Server uses the shared library to determine the user attributes that are available to the custom directory. Before you enter user attributes, you must create the user directory connection.

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

   The Create Attribute Mapping pane opens.

8. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

Directory Attributes Overview (see page 133)
Define an Attribute Mapping (see page 216)

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. Review the prerequisites to configuring a connection over SSL.
2. Install the NSS utility.
3. Create the certificate database files.
4. Add the root Certificate Authority (CA) to the certificate database.
5. Add the server certificate to the certificate database.
6. List the certifications in the certificate database.
7. Configure the user directory connection for SSL.
8. Point the Policy Server to the certificate database.
9. Verify the SSL connection.

Before You Configure a Connection over SSL

Review the following before configuring an LDAP user directory connection over SSL:

- Ensure your directory server is SSL-enabled.

  Note: For more information on configuring your directory server to communicate over SSL, refer to the vendor-specific documentation.

- SiteMinder uses a Netscape LDAP SDK to communicate with LDAP directories. As a result, SiteMinder requires that the database files be in a Netscape version file format (cert7.db).

  Important! Do not use Microsoft Internet Explorer to install certificates into your cert7.db database file.

- A third-party certificate utility, which is compatible with Netscape, is required to manage your SSL certificates. We recommend the Mozilla® Network Security Services (NSS) utility, version 3.2.2.

  Note: Version 3.2.2 is required to support the cert7.db format. Do not use later versions.

- (Active Directory) Considering the following:

  - If the SiteMinder user directory connection was configured with the AD namespace, the following process does not apply. The AD namespace uses the native Windows certificate repository when establishing an SSL connection. When configuring the AD namespace to communicate over SSL:

    - Ensure that the SiteMinder user directory connection is configured for a secure connection. For more information, refer to Configure the User Directory Connection for SSL (see page 180).

    - On the machine hosting the Active Directory instance, ensure that the root CA certificate and the server certificate are added to the services' certificate store.

    Note: For more information on configuring Active Directory to communicate over SSL, refer to the Microsoft documentation.

  - If the SiteMinder user directory connection was configured with the LDAP namespace, complete the following process to configure the connection over SSL.

Install the NSS Utility

You install the NSS utility to manage your certificate database files.

Note: Install the utility on a system to which the Netscape Portable Runtime (NSPR) or the Policy Server is installed. Installing the utility to a system with either component ensures that the necessary DLLs or shared objects are available.

To install the NSS utility

1. Access the Mozilla NSS 3.2.2 FTP site.
2. Download the respective zip or tar for your operating system.

   Note: A zip is not available for Windows Server 2000 or 2003. Download the zip for Windows NT.
3. Extract the NSS utility to a temporary location on the system to which you are managing your certificate database files.

Create the Certificate Database Files

The Policy Server requires that the certificate database files be in the Netscape version file format (cert7.db). You may use the NSS utility to create the certificate database files.

Note: The following procedure details the specific options and arguments to complete the task. For a complete list of the NSS utility options and arguments, refer to the Mozilla documentation on the NSS project page.

To create the certificate database files

1. From a command prompt, navigate to the bin directory in the location to which you extracted the NSS utility.

   Example: C:\nss\bin

   Note: Windows has a native certutil utility. Ensure you are working from the bin directory of the NSS utility or you may inadvertently run the Windows certutil utility.
2. Enter the following command:

   certutil -N -d *certificate_database_directory*

   -N

   Creates the cert7.db, key3.db, and secmod.db certificate database files.

-d *certificate_database_directory*

Specifies the directory to which the NSS utility is to create the certificate database files.

Note: If the file path contains spaces, bracket the path in quotes.

The utility prompts for a password to encrypt the database key.

3. Enter and confirm the password.

   NSS creates the required certificate database files:

   - cert7.db
   - key3.db
   - secmod.db

Example: Create the Certificate Database Files

```
certutil -N -d C:\certdatabase
```

Add the Root Certificate Authority to the Certificate Database

You add the root Certificate Authority (CA) to make it available for communication over SSL.

Note: The following procedure details the specific options and arguments to complete the task. For a complete list of the NSS utility options and arguments, refer to the Mozilla documentation on the NSS project page.

To add the root CA certificate to the certificate database

1. From a command prompt, navigate to the bin directory in the location to which you extracted the NSS utility.

   Example: C:\nss\bin

   Note: Windows has a native certutil utility. Ensure you are working from the bin directory of the NSS utility or you may inadvertently run the Windows certutil utility.

2. Run the following command to add the root CA to the database file:

   ```
   certutil -A -n alias -t trust_arguments -i root_CA_path -d certificate_database_directory
   ```

   -A

   Adds a certificate to the certificate database.

   -n *alias*

   Specifies an alias for the certificate.

   Note: If the alias contains spaces, bracket the alias with quotes.

-t *trust_arguments*

Specify the trust attributes to apply to the certificate when adding it to the certificate database. There are three available trust categories for each certificate, which are expressed in this order: "SSL, email, object signing". Specify the appropriate trust arguments so that the root CA is trusted to issue SSL certificates. In each category position, you may use zero or more of the following attribute arguments.

p

Valid peer.

P

Trusted peer. This argument implies p.

c

Valid CA.

T

Trusted CA to issue client certificates. This argument implies c.

C

Trusted CA to issue server certificates (SSL only). This argument implies c.

Important! This is a required argument for the SSL trust category.

u

Certificate can be used for authentication or signing.

-i *root_CA_path*

Specifies the path to the root CA file. Consider the following:

- The path must include the certificate name.
- Valid extensions for a certificate include .cert, .cer, and .pem.

Note: If the file path contains spaces, bracket the path in quotes.

-d *certificate_database_directory*

Specifies the path to the directory that contains the certificate database.

Note: If the file path contains spaces, bracket the path in quotes.

NSS adds the root CA to the certificate database.

Example: Adding a Root CA to the Certificate Database

```
certutil -A -n "My Root CA"   -t "C,," -i C:\certificates\cacert.cer -d C:\certdatabase
```

Add the Server Certificate to the Certificate Database

You add the server certificate to the certificate database to make it available for communication over SSL.

Note: The following procedure details the specific options and arguments to complete the task. For a complete list of the NSS utility options and arguments, refer to the Mozilla documentation on the NSS project page.

To add the server certificate to the certificate database

1. From a command prompt, navigate to the bin directory in the location to which you extracted the NSS utility.

   Example: C:\nss\bin

   Note: Windows has a native certutil utility. Ensure you are working from the bin directory of the NSS utility or you may inadvertently run the Windows certutil utility.

2. Run the following command to add the root certificate to the database file:

   certutil -A -n *alias* -t *trust_arguments* -i server*_certificate_path* -d *certificate_database_directory*

   -A

   Adds a certificate to the certificate database.

   -n *alias*

   Specifies an alias for the certificate.

   Note: If the alias contains spaces, bracket the alias with quotes.

   -t *trust_arguments*

   Specify the trust attributes to apply to the certificate when adding it to the certificate database. There are three available trust categories for each certificate, which are expressed in this order: "SSL, email, object signing". Specify the appropriate trust arguments so that the certificate is trusted. In each category position, you may use zero or more of the following attribute arguments:

   p

   Valid peer.

   P

   Trusted peer. This argument implies p.

   Important! This is a required argument for the SSL trust category.

-i *server_certificate_path*

Specifies the path to the server certificate. Consider the following:

- The path must include the certificate name.
- Valid extensions for a certificate include .cert, .cer, and .pem.

Note: If the file path contains spaces, bracket the path in quotes.

-d *certificate_database_directory*

Specifies the path to the directory that contains the certificate database.

Note: If the file path contains spaces, bracket the path in quotes.

NSS adds the server certificate to the certificate database.

Example: Adding a Server Certificate to the Certificate Database

```
certutil -A -n "My Server Certificate" -t "P,," -i C:\certificates\servercert.cer -d C:\certdatabase
```

List the Certificates in the Certificate Database

You list the certifications to verify that they were added to the certificate database.

Note: The following procedure details the specific options and arguments to complete the task. For a complete list of the NSS utility options and arguments, refer to the Mozilla documentation on the NSS project page.

To list the certifications in the certificate database

1. From a command prompt, navigate to the bin directory in the location to which you extracted the NSS utility.

   Example: C:\nss\bin

   Note: Windows has a native certutil utility. Ensure you are working from the bin directory of the NSS utility or you may inadvertently run the Windows certutil utility.

2. Run the following command:

   ```
   certutil -L -d certificate_database_directory
   ```

   -L

   Lists all of the certificates in the certificate database.

   -d *certificate_database_directory*

   Specifies the path to the directory that contains the certificate database.

   Note: If the file path contains spaces, bracket the path in quotes.

NSS displays the root CA alias, the server certificate alias, and the trust attributes you specified when adding the certificates to the certificate database.

Example: List the Certificates in the Certificate Database

```
certutil -L -d C:\certdatabase
```

Configure the User Directory Connection for SSL

You configure the user store connection to ensure that an SSL connection is used when the Policy Server and user store communicate.

To configure the user store connection for SSL

1. Log in to the Administrative UI.
2. Click Infrastructure, Directory.
3. Click User Directory, Modify User Directory.

   The Modify User Directory pane appears with a list of existing user directory connections.
4. Select the user directory connection you want, and click Select.

   User directory settings appear.
5. Select the Secure Connection check-box, and click Submit.

The user directory connection is configured to communicate over SSL.

Point the Policy Server to the Certificate Database

You point the Policy Server to the certificate database to configure the Policy Server to communicate with the user directory over SSL.

To point the Policy Server to the certificate database

1. Start the Policy Server Management Console.
2. Click the Data tab.

3. Enter the path to the Netscape certificate database file in the Netscape Certificate Database File field.

   Example: C:\certdatabase\cert7.db

   Note: The key3.db file must also be in the same directory as the cert7.db file.

4. Restart the Policy Server.

The Policy Server is configured to communicate with the user directory over SSL.

Verify the SSL Connection

You verify the SSL connection to ensure the user directory and the Policy Server are communicating over SSL.

To verify the SSL connection

1. Log in to the Administrative UI.
2. Click Infrastructure, Directory.
3. Click User Directory, View User Directory.

   The View User Directory pane appears with a list of existing user directory connections.

4. Select the connection you want, and click Select.

   User directory settings appear.

5. Click View contents.

   If SSL is properly configured, the Directory Content pane appears and lists the contents of the user directory.

LDAP Load Balancing and Failover

The Policy Server can spread LDAP queries over multiple LDAP servers to enable failover and load balancing. If configured for failover, the Policy Server uses one LDAP server to fulfill requests until that server fails to respond. When the default server does not respond, the Policy Server routes the request to the next server specified for failover. This process can be repeated over multiple servers. Once the default server is able to fulfill requests again, the Policy Server routes requests to the original server.

If configured for load balancing, the Policy Server spreads requests over the specified LDAP servers. This distributes requests evenly across LDAP servers. Coupled with failover, load balancing provides faster, more efficient access to LDAP user directory information, with the added benefit of redundancy in the event of a server failure.

Port Number Considerations

You can assign ports to individual LDAP servers and failover groups, or let the Policy Server use the default port numbers for LDAP servers.

The following guidelines apply when specifying port numbers:

If	Then
any server in a failover group other than the last server contains a port number	The Policy Server assumes that servers in the group that do not have a specific port are using a default port. The default for SSL is 636. The default for non-SSL is 389. For example, a failover group of servers includes the following: 123.123.12.12:350 123.123.34.34 The first server in the failover group includes port 350. Communication with that server takes place on port 350. If the first server fails, the Policy Server communicates with the second server using the default port 389 because no port was specified for the second server in the failover group.

Configure Failover

You configure failover to provide for redundancy if the primary LDAP directory connection becomes unavailable.

Note: If you are adding a server for failover, the failover directory must use the same type of communication (SSL or non-SSL) as the primary directory, since both directories share the same port number.

To configure failover

1. Click Configure on the Directory Setup group box on the User Directory pane.

   The Directory Failover and Load Balancing Setup pane opens. The primary user directory opens in the Failover Group.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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: You can click Help for a description of fields, controls, and their respective requirements.

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: You can click Help for a description of fields, controls, and their respective requirements.
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 B.

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: You can click Help for a description of fields, controls, and their respective requirements.

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

To configure a SQL Query Scheme

1. Click Infrastructure, Directory.
2. Click SQL Query Scheme, Create SQL Query Scheme.

   The Create SQL Query Scheme pane opens.
3. Verify that Create a new object is selected, and click OK.

   The SQL Query Scheme: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Type a name and description in the fields on the General group box.
5. Update the contents of the query fields to correspond to your database schema.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

   You must configure each of the queries to work with your relational database. You must replace the following database table and column names with the table and column names from your relational database:

   - Name

     When you configure the Policy Server to use a user store residing in a relational database, the Name parameter for a user must be unique so that 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

6. Select a query type, and click Submit.

   The query is saved. You can associate query scheme with a user directory connection

7. Restart the Policy Server using the Policy Server Management Console.

More information:

Configure ODBC Directory Connections (see page 162)

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'.

Configure SQL Query Schemes for Authentication via Stored Procedures

When stored procedures are required for authentication with ODBC user directories, configure the SQL query scheme to call the stored procedure as follows:

SQLServer

Syntax: Call *Procedure_Name* %s , %s

Example: Call EncryptPW %s , %s

Stored procedures in SQLServer must meet the following requirements:

- The first parameter must be the username, and the second parameter must be the password.
- All parameters must be defined using the keyword OUT.
- The stored procedure must return an integer value.

The following example shows how to create a stored procedure for a SQLServer user directory:

```
CREATE PROCEDURE EncryptPW
@UserName varchar(20) OUT ,
@PW   varchar(20) OUT
AS
SELECT Smuser.name from Smuser where Smuser.name= @UserName and   password = @PW
SELECT Smuser.password from Smuser where name= @UserName and   password = @PW
return 0
```

MySQL

Syntax: Call *Procedure_Name* %s, %s

Example: Call EncryptPW %s, %s

Stored procedures in MySQL must meet the following requirements:

- The first parameter must be the username, and the second parameter must be the password.
- The stored procedure does not return a value.

The following example shows how to create a stored procedure for a MySQL user directory:

```
CREATE PROCEDURE EncryptPW(INOUT p_UserName varchar(20), INOUT p_PW varchar(20))
BEGIN
SELECT SmUser.Name into p_UserName from test.SmUser where SmUser.Name =
p_UserName and SmUser.Password = p_PW;
SELECT SmUser.Password into p_PW from test.SmUser where SmUser.Name =
p_UserName and SmUser.Password = p_PW;
END;
```

Oracle Functions

For Oracle user directories, you can create the following functions using the templates below:

- EncryptPW
- ChangePW

EncryptPW Function

The EncryptPW function must return an integer value, as follows:

- value = 0

  Specifies success.

- value = 1

  Specifies failure.

You can use the following template to create the EncryptPW function:

```
CREATE OR REPLACE FUNCTION EncryptPW(p_UserName IN OUT SmUser.Name%type, p_PW IN OUT
SmUser.Password%type)
RETURN INTEGER IS
nRet INTEGER :=1;
nCount NUMBER := 0;
BEGIN
select count(*) into nCount
from SmUser
where SmUser.Name = p_UserName and SmUser.Password = p_PW;
IF (nCount = 1) THEN
SELECT SmUser.Name   into p_UserName
from SmUser
where SmUser.Name = p_UserName and SmUser.Password = p_PW;
SELECT SmUser.Password into p_PW
from SmUser
where SmUser.Name = p_UserName and SmUser.Password = p_PW;
```

```
RETURN 0;
END IF;
RETURN nRet;
END EncryptPW;
```

ChangePW Function

The ChangePW function must return an integer value, as follows:

- value = 1

  Specifies success.

- value = 0

  Specifies failure.

You can use the following template to create the ChangePW function:

```
CREATE OR REPLACE FUNCTION ChangePW(p_PW IN SmUser.Password%type, p_UserName IN
SmUser.Name%type)
RETURN INTEGER IS
nRet INTEGER :=1;
nCount NUMBER := 0;
BEGIN
select count(*) into nCount
from SmUser
where SmUser.Name = p_UserName;
IF (nCount = 1) THEN
UPDATE SmUser
SET SmUser.Password = p_PW
where SmUser.Name = p_UserName;
COMMIT;
RETURN 0;
END IF;
```

Asynchronous Call Support During Failover and Connection Pooling

Synchronous calls are reliable, returning only after the request is complete. Asynchronous calls return immediately. A caller can choose to abandon an asynchronous call and avoid delays associated with network failures.

SiteMinder supports asynchronous calls to the following databases:

- SQLServer
- Oracle 8 on Windows NT and Solaris

Asynchronous Call Support Configuration

The following registry options are stored under the registry sub-key *Netegrity\SiteMinder\CurrentVersion\Database.*

AsynchronousCalls

Determines whether database calls are made asynchronously.

Values: 0 (no); 1 (yes)

Default: 0

AsynchronousSleepTime

Specifies the amount of time between calls to wait before checking the status of an outstanding SQL call.

Values: 0 to n milliseconds

Default: 15 milliseconds

LoginTimeout

The amount of time to allow for a connection to log in to the database.

Values: minimum of 1 second

Default: 15 seconds

QueryTimeout

The amount of time to allow for a query to complete before canceling it.

Values: minimum of 1 second

Default: 15 seconds

Note: When SQL Server is running on Windows NT, asynchronous call support causes a very small memory leak per abandoned connection. You may choose to extend the timeouts to reduce the number of failovers in an unreliable network by adjusting the settings discussed in the table above.

Configure Oracle 8 on Solaris for Asynchronous Calls

The Merant ODBC driver for Oracle on Solaris 2.6 and 2.7 may cause a core dump when asynchronous calls are supported. This is due to an Oracle bug which is fixed as follows:

Oracle 8.0.5

To each data source in system_odbc.ini in the <install_directory>/db directory, add the entry:

```
ArraySize=1
```

Note: This change turns off multi-row fetches and will affect performance when loading large policy stores.

Oracle 8.1.5

1. Remove or rename libclntsh.so in *siteminder*/bin and/or *siteminder*/odbc/lib.
2. Verify that the Oracle client has libclntsh.so installed in $ORACLE_HOME/lib. Refer to the Oracle documentation for installation and rebuilding instructions.
3. Make sure that LD_LIBRARY_PATH references the Oracle client library directory $ORACLE_HOME/lib.

If you get the following log message:

```
[MERANT][ODBC Oracle 8 driver][Oracle 8] ORA-03106: fatal two-task
communication protocol error
```

add an entry to the affected data source in system_odbc.ini in the <install directory>/db directory:

ArraySize=<*value_greater_than 60000*>

This increases the size of the multi-row fetch buffer to eliminate the error. The default value of this variable is 60000 bytes. The maximum allowed value is 4 Gigabytes.

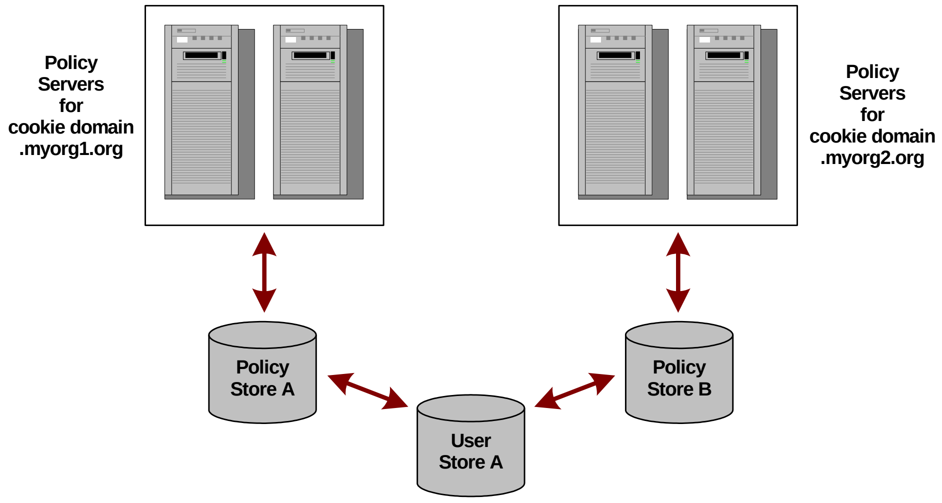
ODBC Connection Pooling

The following applies to ODBC connection pooling:

- Connections are pooled by ODBC data source. If a data source is used by one or more user directories AND (the policy store or another store on the policy server), the number of connections available may reach the total of the individual values specified for user directories and stores.
- SQL Server is more limited than Oracle in how connections may be shared. Two callers may not share an open result set while the results are being fetched to the client. Although the Policy Server uses server side cursors for SQL Server, there are limitations to concurrent activity, especially on multi-processor machines. Increasing the number of connections will generally improve concurrency.
- Oracle allows for multiple callers to share a connection, but may serialize calls internally. Again, you may see improved concurrency by increasing the number of connections allowed.
- If connections are shareable, the number of active requests will be balanced across the connections in a pool.
- Once a connection is opened, it is not closed until the Policy Server is shut down.

Define the Same User Directory Connection in Multiple Policy Stores

Every Policy Server is connected to a policy store. Multiple Policy Servers may be configured to point to a single policy store. When you open an instance of the Administrative UI, the objects that you add and modify are stored in the policy store associated with the Policy Server. As shown in the following figure, your SiteMinder environment may contain multiple independent policy stores for maintaining Policy Server data.


Policy Servers for cookie domain .myorg1.org
Policy Servers for cookie domain .myorg2.org
Policy Store A
Policy Store B
User Store A

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 135)

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 135)

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.

Unnamed expressions are stored in policy store objects like responses and rules. *Named expressions* are policy store objects that can be referenced by name and reused. 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. SiteMinder evaluates named expressions to determine the values of calculated user attributes.

There are two types of named expressions:

- Virtual user attributes (see page 200)
- User classes (see page 202)

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

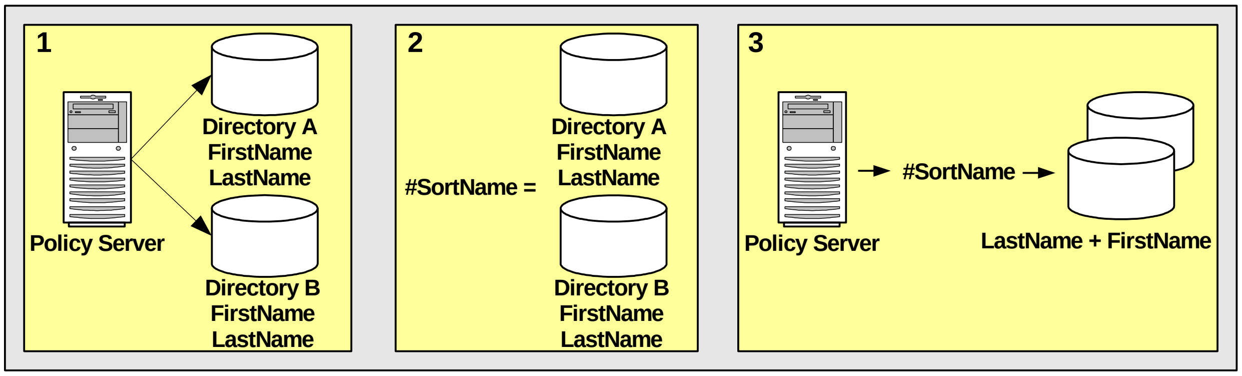
More information:

Expression Syntax Overview (see page 712)

Virtual User Attribute Use Case

This use case represents a basic scenario in which two LDAP user directories identify the last and first names of users with different underlying schema.

The following illustration shows how the virtual user attribute *#SortName* (LastName,FirstName) can be calculated for users in different user directories through user attribute mapping. User attribute mapping lets you map one common name to different user attribute names in different user directories.



1. Two user directories identify the last and first names of users differently. To create a common view of this information, you can create user attribute mappings:

   ■ FirstName maps to the underlying directory schema that identify the first names of users in Directory A and Directory B.

   ■ LastName maps to the underlying directory schema that identify the last names of users in Directory A and Directory B.

2. *#SortName* is a virtual user attribute that can calculate the sort name of users in both directories with the following expression:

   (LastName + "," + FirstName)

3. Instead of entering the expression (LastName + "," + FirstName) repeatedly, you can create a virtual user attribute named *#SortName* that is defined as: (FirstName + "," + LastName). Then, you can enter *#SortName* each time that the expression is needed.

Define a Virtual User Attribute

You define a virtual user attribute to calculate user information that is not uniquely referenced by one or more user directories.

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

To define a virtual user attribute

1. Click Policies, Expressions.
2. Click Named Expression, Create Named Expression.

   The Create Named Expression pane opens.
3. Verify that a new object of type Expression is selected, and click OK.

   The Create Named Expression: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select Virtual User Attribute, and type the name and a description in the fields on the General group box.
5. Type the expression in the Expression field on the Add Named Expression group box.
6. (Optional) Select the Disabled check box on the Add Named Expression group box.

   The named expression is marked as disabled, is not listed in the expression editor, and cannot be called by another expression, named or unnamed.
7. (Optional) Select the Private check box on the Add Named Expression group box.

   The named expression is marked as private and can only be called by other named expressions; it cannot be called by unnamed expressions.
8. (Optional) Click Edit on the Add Named Expression group box.

   The Expression Editor pane opens.
9. Click Submit.

   The Create Named Expression task is submitted for processing.

User Classes

A user class lets you define a re-usable expression to calculate user information. You use this type of expression when the user attribute is not uniquely referenced by the user directory. Rather, the user attribute must be calculated using attributes and other criteria that is established by business logic.

A user class names an expression that returns a TRUE value if a user is a member of a specified class or a FALSE value if not.

User classes are prefixed by the "at" symbol (@). The "at" symbol prevents name clashes with user attribute names and mappings and is a visual reminder that the user attribute value is calculated.

As an expression, a user class can include:

- User attributes, either directory-specific or mapped
- References to other named expressions
- SiteMinder built-in functions and expression syntax

A user class is not a role. A role is a feature of Enterprise Policy Management. While roles can use user classes, they have additional information associated with them. For more information about roles, see the Enterprise Policy Management.

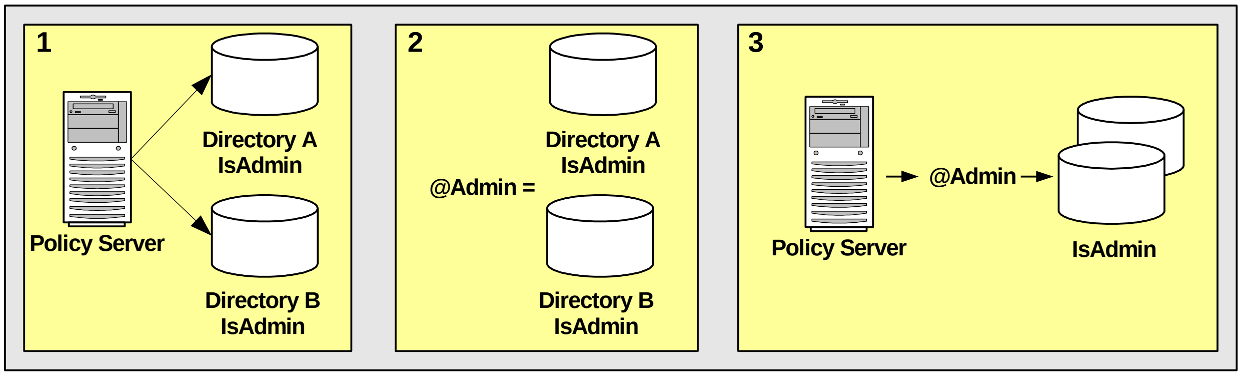
More information:

Expression Syntax Overview (see page 712)

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

To create a user class

1. Click Policies, Expressions.
2. Click Named Expression, Create Named Expression.

   The Create Named Expression pane opens.
3. Verify that a new object of type Expression is selected, and click OK.

   The Create Named Expression: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select User Class, and type the name and a description in the fields on the General group box.
5. Type the expression in the Expression field on the Add Named Expression group box.

   Note: The expression must be a Boolean expression.
6. (Optional) Select the Disabled check box on the Add Named Expression group box.

   The named expression is marked as disabled, is not listed on the Expression Editor pane, and cannot be called by another expression, named or unnamed.
7. (Optional) Select the Private check box on the Add Named Expression group box.

   The named expression is marked as private and can be called by other named expressions, but not by unnamed expressions.

8. (Optional) Click Edit on the Add Named Expression group box.

   The Expression Editor pane opens.

9. Click Submit.

   The Create Named Expression task is submitted for processing.

How to Use the Expression Editor

You can use the expression editor to:

- Look up 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 208)
   - Delete a Condition from an Expression (see page 208)
   - Group the Conditions in an Expression (see page 208)
   - Change a Boolean Operator in an Expression (see page 209)
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 709)

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

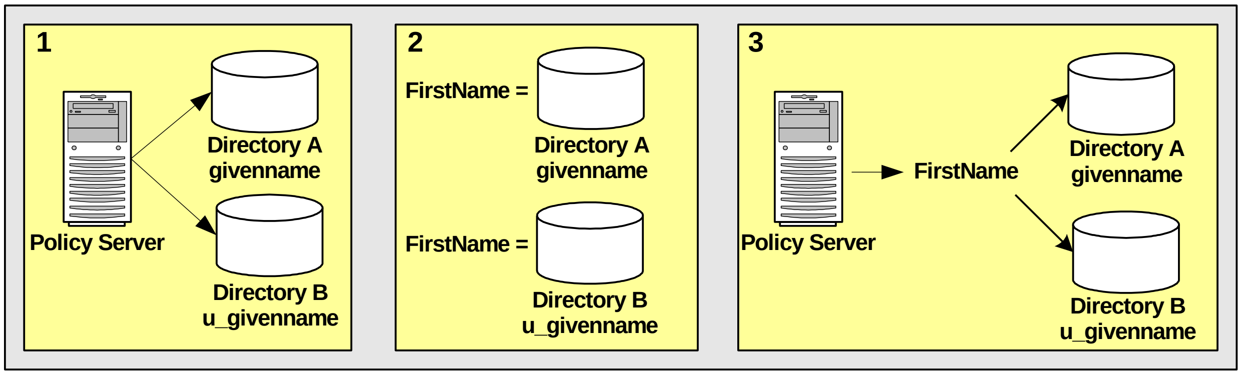
User directories store user information such as organizational information, user and group attributes, and individual credentials. Multiple user directories in a SiteMinder environment often store the same user information, but use different underlying schema and user attribute names to identify them. 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 216)
- Group Name (see page 218)
- Mask (see page 221)
- Constant (see page 226)
- Expression (see page 228)

Alias

Alias attribute mapping lets you map a common name to the name used by the underlying directory schema to identify a user attribute. *Alias* attribute mappings map common names to user attribute values that can be read or changed. This type of access is called read/write (RW).

Alias attribute mappings can map to user attributes that have any of the following data types:

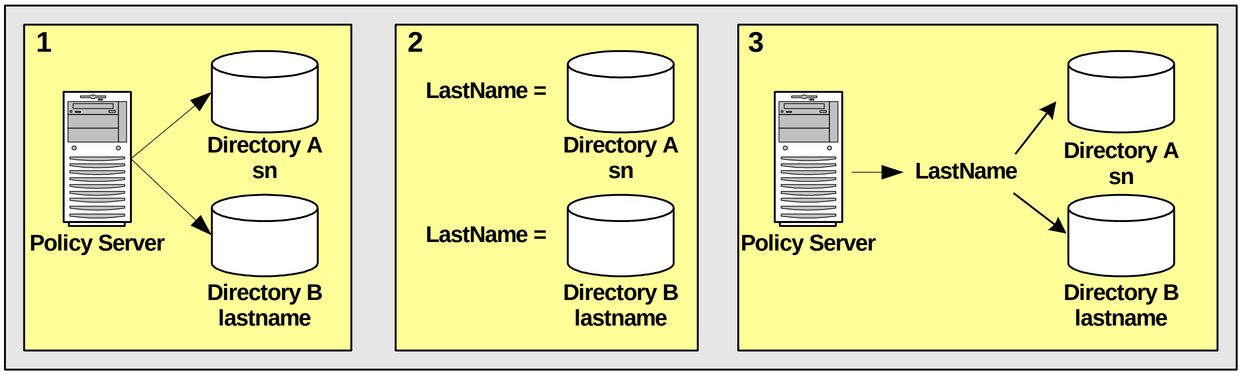
- string
- number
- Boolean

Alias Attribute Use Case

This use case represents a basic scenario in which two LDAP user directories identify the last name of users with different underlying schema.

Note: Advanced use cases are shown in the Apply Attribute Mapping section. The advanced use cases detail how you use different attribute mapping types to identify the same user attribute across different directory types.

The following illustration details how two *alias* attribute mappings can define a universal schema and create a common view of the same user information.



1. Two user directories identify the last name of users differently:
   - Directory A identifies the last name of users with sn.
   - Directory B identifies the last name of users with lastname.

   This results in two different representations and views of the same user information.
2. LastName is the common name or *alias* that is mapped to the underlying directory schema:
   - LastName is mapped to sn in Directory A.
   - LastName is mapped to lastname in Directory B.
3. LastName results in a common view of the same user information. You can reference LastName when defining policies, expressions, or other objects that require the last name of users, without concern for the directory-specific schema, because the directories are operationally identical. SiteMinder determines that LastName is sn in Directory A and lastname in Directory B.

More information:

Named Expressions (see page 198)

Create an Alias Attribute Mapping

You can map a common name or *alias* to the user attribute name specified by the user directory's underlying schema. The user attribute's data type can be string, number, or Boolean. You can use an alias attribute mapping to read or change a user attribute value in a user directory. User attribute mappings are directory-specific.

To create an alias attribute mapping

1. Navigate to the User Directory: *Name* pane.
2. Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create Attribute Mapping: *Name* pane opens.
4. Type the common name and a description of the attribute mapping in the fields on the General group box.

   Note: Common names must conform to the same rules as user attribute names.
5. Select Alias on the Properties group box.
6. Type the definition in the Definition box on the Properties group box.

   Example:

   Name: FirstName

   Definition: givenname

   Description: The common name FirstName is mapped to the user attribute name givenname.
7. (Optional) Select Disabled to disable this attribute mapping.
8. Click OK.

   The Create Attribute Mapping task is submitted for processing, and the new attribute mapping is added to the list on the Attribute Mapping List group box.

Group Name

A *group name* attribute lets you map a common name to the underlying directory schema that identifies whether a user belongs to a specific group. Group name attribute mappings map common names to user attributes that can be read or changed. This type of access is called read/write (RW).

Group name attribute mappings result in a Boolean value. If the user is a member of the specified group, the mapping results in a TRUE value. Otherwise, the result is FALSE.

Group name attribute mapping and user classes, one type of named expression, are similar in the following ways:

- Both are used to determine group membership.
- Both result in a Boolean value.

Group name attribute mapping differs from user classes as follows:

- *A* group name attribute mapping is defined for particular user directories. User classes are global and can be applied to any user in any user directory.
- A group name attribute mapping can change a user's membership in a group in a user directory. User classes are Boolean expressions and cannot be changed.
- User classes must begin with the "at" sign (@). A group name mapping does not.

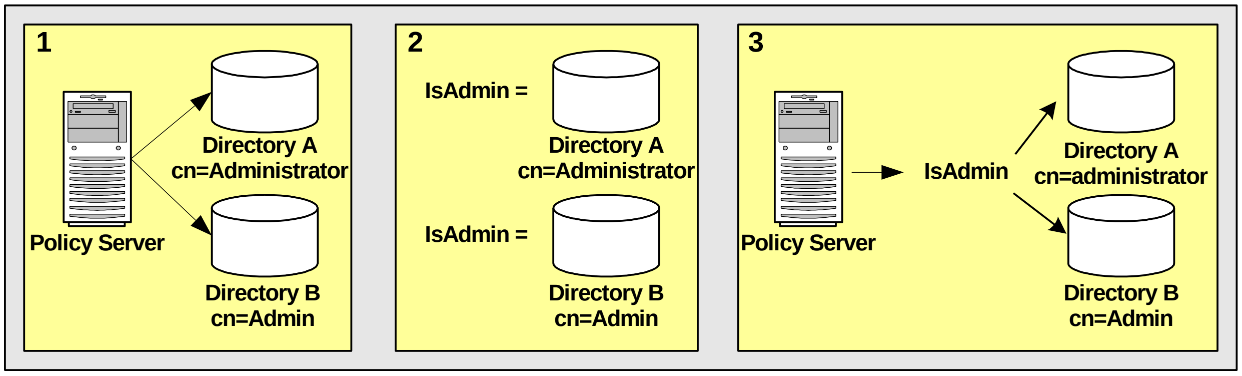
Note: For more information about user classes, see the Named Expressions.

Group Name Use Case

This use case represents a basic scenario in which two LDAP user directories use different underlying schema to identify users that belong to an Administrator group.

Note: Advanced use cases are shown in the Apply Attribute Mapping section. The advanced use cases detail how you use different attribute mapping types to identify the same user attribute across different directory types.

The following illustration details how two *group name* attribute mappings can define a universal schema and create a common view of the same user information.



1. Two user directories identify membership to the administrator group differently:

   - Directory A identifies membership in the administrator group as cn=Administrators,ou=groups,o=acme.com.

   - Directory B identifies membership in the administrator group as cn=Admin,ou=groups,o=acme.com.

   This results in two different representations and views of the same user information.

2. IsAdmin is the common name that is mapped to the underlying directory schema:

   - IsAdmin is mapped to cn=Administrators,ou=groups,o=acme.com in Directory A.

   - IsAdmin is mapped to cn=Admin,ou=group,o=acme.com in Directory B.

3. IsAdmin results in a common view of the administrator group. You can reference IsAdmin when defining policies, expressions, or other objects that apply to the Administrator group, without concern for the directory-specific schema, because the directories are operationally identical. SiteMinder determines that IsAdmin is cn=Administrators,ou=groups,o=acme.com in Directory A and cn=Admin,ou=group,o=acme.com in Directory B.

More information:

Named Expressions (see page 198)

Create a Group Name Attribute Mapping

You define a group name attribute to map a common name to the underlying directory schema that identifies whether a user belongs to a specific group. The result of a group name attribute mapping is a Boolean value. You can use a group name attribute mapping to read or change a user's group name in a user directory. User attribute mappings are directory-specific.

Note: For information about creating a global object that returns group membership, see user classes in the Named Expression chapter in this guide.

To Create a User Attribute Mapping of Type Group

1. Navigate to the User Directory: *Name* pane.
2. Click Create on the Attribute Mapping List group box.

   The Create Attribute Mapping pane opens.

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

   The Create Attribute Mapping: *Name* pane opens.

4. Type the common name and a description of the attribute mapping in the fields on the General group box.

   Note: Common names must conform to the same rules as user attribute names.

5. Select Group on the Properties group box.

6. Type the definition in the Definition box on the Properties group box.

   Example:

   Name: IsAdmin

   Definition: cn=administrators,ou=groups,o=acme.com

   Description: The common name IsAdmin is mapped to the group name cn=administrators,ou=groups,o=acme.com. If the user is a member of cn=administrators,ou=groups,o=acme.com, IsAdmin is TRUE.

7. (Optional) Select Disabled to disable this attribute mapping.

8. Click OK.

   The Create Attribute Mapping task is submitted for processing, and the new attribute mapping is added to the list on the Attribute Mapping List group box.

Mask

Mask attribute mapping lets you map a common name to the name used by the underlying directory schema to identify a user attribute that stores a bit pattern. *Mask* attribute mappings map common names to user attributes that can be read or changed. This type of access is called read/write (RW).

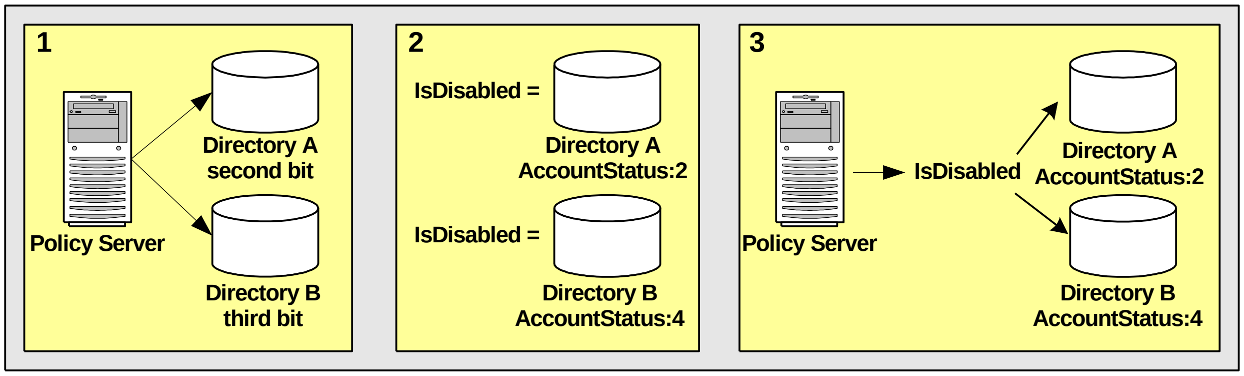
Mask attribute mappings result in a Boolean value. If the bit pattern in the user directory matches the specified mask, the mapping results in a TRUE value. Otherwise, the result is FALSE.

Mask Use Case

This use case represents a basic scenario in which two Active Directory user directories identify disabled user accounts with different underlying schema.

Note: Advanced use cases are shown in the Apply Attribute Mapping section. The advanced use cases detail how you use different attribute mapping types to identify the same user attribute across different directory types.

The following illustration details how two *mask* attribute mappings can define a universal schema and create a common view of the same user information.



1. Two user directories contain a user attribute named AccountStatus. AccountStatus stores user information in a bit pattern, where each bit is a flag.

   - In Directory A, the second bit flags a disabled account. When the second bit equals 1, the account is disabled.
   - In Directory B, the third bit flags a disabled account. When the third bit equals 1, the account is disabled.

   This results in two different representations and views of the same user information.

2. IsDisabled is the common name that is mapped to the underlying directory schema. In both directories, IsDisabled is mapped to AccountStatus.

   - In Directory A, SiteMinder uses the bit mask 2 (decimal) to determine whether the second bit of AccountStatus is set and the account is disabled.
   - In Directory B, SiteMinder uses the bit mask 4 (decimal) to determine whether the third bit of AccountStatus is set and the account is disabled.

3. IsDisabled results in a common view of disabled user accounts. You can reference IsDisabled when defining policies, expressions, or other objects that require the account status of users, without concern for the directory-specific schema, because the directories are operationally identical. SiteMinder determines that IsDisabled is AccountStatus:2 for Directory A and AccountStatus:4 for Directory B.

More information:

Named Expressions (see page 198)

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

The purpose of a bit mask in mask attribute mapping is to test the value of one or more bits by masking the values of the other bits in a user attribute.

A mask attribute mapping is defined as follows:

```
user_attribute_name:bit_mask
```

For example, assume that the user attribute is named AccountStatus and that AccountStatus stores the states of the following three flags in a bit pattern, as follows:

Bit Pattern	Flag
00?	account disabled?
0?0	password expired?
?00	gold member?

When a bit equals one, the flag is TRUE, as follows:

Bit Pattern	Account Status
000 (0)	no flags are TRUE
001 (1)	account disabled
010 (2)	password expired
100 (4)	gold member
011 (3)	password expired, account disabled
101 (5)	gold member, account disabled
110 (6)	gold member, password expired
111 (7)	gold member, password expired, account disabled

Note: Equivalent decimal values are shown in parentheses.

Assume that you only want to test whether a user is a gold member. To test this bit, select the bit pattern that corresponds to a gold member as the bit mask or 100 (binary) and specify it as 4 (decimal). The resulting mask attribute mapping is defined as follows:

```
AccountStatus:4
```

SiteMinder performs a bitwise AND operation on AccountStatus and the bit mask and tests whether the result is equal to the bit mask. If they are equal, the value of the tested bit is one, and the flag is TRUE, as shown in the following table:

Account Status	Bit Mask	Result of Bitwise AND	Gold Member?
000 (0)	100 (4)	000 (0)	FALSE
001 (1)	100 (4)	000 (0)	FALSE
010 (2)	100 (4)	000 (0)	FALSE
011 (3)	100 (4)	000 (0)	FALSE
100 (4)	100 (4)	100 (4)	TRUE
101 (5)	100 (4)	100 (4)	TRUE
110 (6)	100 (4)	100 (4)	TRUE
111 (7)	100 (4)	100 (4)	TRUE

Note: Equivalent decimal values are shown in parentheses.

You can also use a bit mask to test the value of a bit set or more than one bit at a time. Assume that you want to know whether the account is disabled and the password has expired. To test these bits, specify a bit mask of 011 (binary) or 3 (decimal). The resulting mask attribute mapping is defined as follows:

```
AccountStatus:3
```

SiteMinder performs a bitwise AND operation on AccountStatus and the bit mask and tests whether the result is equal to the bit mask. If they are equal, the value of both tested bits is one, and both flags are TRUE, as shown in the following table:

Account Status	Bit Mask	Result of Bitwise AND	Both Flags Set?
000 (0)	011 (3)	000 (0)	FALSE
001 (1)	011 (3)	001 (1)	FALSE
010 (2)	011 (3)	010 (2)	FALSE

Account Status	Bit Mask	Result of Bitwise AND	Both Flags Set?
011 (3)	011 (3)	011 (3)	TRUE
100 (4)	011 (3)	000 (0)	FALSE
101 (5)	011 (3)	001 (1)	FALSE
110 (6)	011 (3)	010 (2)	FALSE
111 (7)	011 (3)	011 (3)	TRUE

Note: Equivalent decimal values are shown in parentheses.

Constant

Constant attribute mapping lets you map a common name to a value that is the same or *constant* for every user in a directory. Since *constant* attribute mappings map common names to constants, which are read only (R), they cannot be changed (except by a system administrator).

Constant attribute mappings can map to constants that have any of the following data types:

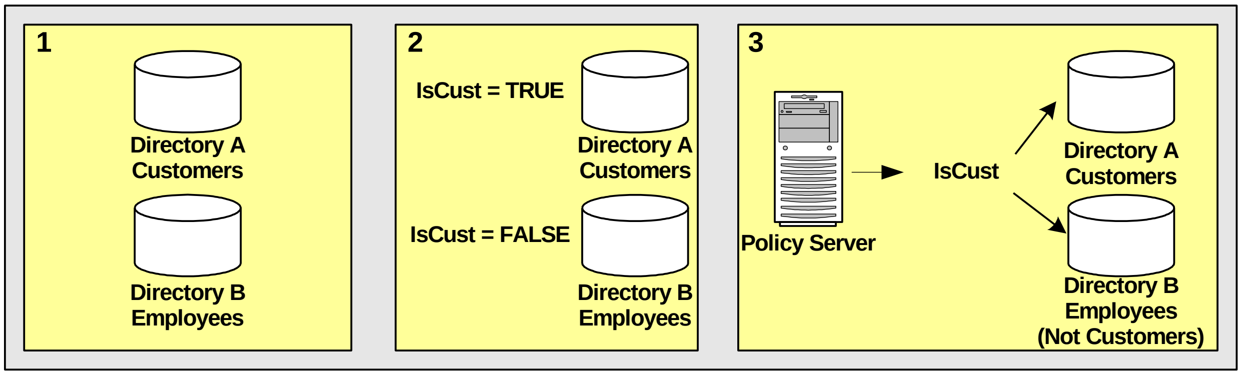
- string
- number
- Boolean

Constant Use Case

This use case represents a basic scenario in which one user directory only stores customers, while another user directory only stores employees.

Note: Advanced use cases are shown in the Apply Attribute Mapping section. The advanced use cases detail how you use different attribute mapping types to identify the same user attribute across different directory types.

The following illustration details how two *constant* attribute mappings can represent different values for different user directories.



1. Directory A only stores customers. Directory B only stores employees.
2. IsCust is the common name that is mapped to different values in different directories:
   - IsCust is mapped to TRUE in Directory A.
   - IsCust is mapped to FALSE in Directory B.
3. You can reference IsCust when defining policies, expressions, or other objects that must determine if a user is a customer, without attending to the particular directory in which the user is stored. SiteMinder determines that every user in Directory A is a customer, while every user in Directory B is not a customer.

More information:

Named Expressions (see page 198)

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 198)

Expression Use Case

This use case shows how you can use an expression attribute mapping to simplify references to multiple user attributes in one directory.

Given:

This use case represents a basic scenario in which a protected resource requires the sort name of users (last name,first name). The user directory does not uniquely reference this user attribute, but does store the last name of users as surname and the first name of users as givenname.

Solution:

Map a common name to an expression that creates the sort name using the user attribute names specified by the user directory's underlying schema.

- Name the mapping **SortName**.
- Define SortName as:

```
{surname + "," + givenname}
```

Note: The expression must conform to the syntax rules of a SiteMinder expression.

Result:

You can reference SortName when defining policies, expressions, or other objects that require the sort name of users, without concern for the directory-specific schema. SiteMinder calculates the sort name using the expression.

Note: Advanced use cases are shown in the Apply Attribute Mapping section. The advanced use cases detail how you use different attribute mapping types to identify the same user attribute across different directory types.

More information:

Named Expressions (see page 198)
Expression Syntax Overview (see page 712)

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.

Apply User Attribute Mapping

Multiple user directories in a SiteMinder environment often store the same user attributes, but use different underlying schema to identify them. In this example, a retail clothing company uses two user directories of different types. Directory A is an internal LDAP user directory for employees only. Directory B is an ODBC user directory for customers only. Each user attribute mapping is specific to the user directory in which it is defined.

The following table details how Directory A and Directory B use different underlying schema to identify the same user information. The accompanying use cases explain how you can use different attribute mappings to define a universal schema that creates a common view of the same user information, thereby making the directories operationally identical from a SiteMinder perspective.

Attribute Description	**Directory A (LDAP)**	**Directory B (ODBC)**
The first name of users	givenname	u_first_name
The last name of users	surname	u_last_name
The sort name of users (last name, first name)	The user directory does not uniquely store the user attribute.	sort_name
Is the user a customer?	group:cn=customer,ou=groups,o=acme.com	Users are always customers
Is the user account disabled?	The user directory has an AccountStatus attribute, which is a set of flags. The second bit indicates a disabled account.	u_disabled

First Name Use Case

This use case shows how you can use two *alias* attribute mappings to represent the first name user attribute in Directory A and Directory B.

Given:

User Directory A identifies the first name of users with givenname. Directory B identifies the first name of users with u_first_name. This results in two different representations and views of the same user information.

Solution:

1. Create an *alias* attribute mapping for Directory A.
   - Name the mapping **FirstName**.
   - Define FirstName as **givenname**.
2. Create an *alias* attribute mapping for Directory B.
   - Name the mapping **FirstName**.
   - Define FirstName as **u_first_name**.

Result:

FirstName results in a common view of the same user information. You can reference FirstName when defining policies, expressions, or other objects that require the first name of users, without concern for the directory-specific schema, because the directories are operationally identical. SiteMinder determines that the first name of users is identified by givenname when referencing users in Directory A and is identified by u_first_name when referencing users in Directory B.

Last Name Use Case

This use case shows how you can use two *alias* attribute mappings to represent the last name user attribute in Directory A and Directory B.

Given:

User Directory A identifies the last name of users with surname. Directory B identifies the last name of users with u_last_name. This results in two different representations and views of the same user information.

Solution:

1. Create an *alias* attribute mapping for Directory A.
   - Name the mapping **LastName**.
   - Define LastName as **surname**.
2. Create an a*lias* attribute mapping for Directory B.
   - Name the mapping **LastName**.
   - Define LastName as **u_last_name**.

Result:

You can reference LastName when defining policies, expressions, or other objects that require the last name of users, without concern for the directory-specific schema, because the directories are operationally identical. SiteMinder determines that the last name of users is identified by surname when referencing users in Directory A and is identified by u_last_name when referencing users in Directory B.

Sort Name Use Case

This use case shows how you can use a *calculated expression* attribute mapping and an *alias* attribute mapping to represent the sort name of a user in Directory A and Directory B.

Given:

1. Directory A does not uniquely identify a user's sort name. Directory A does store a user's first name as givenname and a user's last name as surname.
2. Directory B identifies a user's sort name with sort_name.

   This results in two different representations and views of the same user information.

Solution:

1. Create a *calculated expression* attribute mapping for Directory A:
   - Name the mapping **SortName**.
   - Define SortName as:

     (surname + "," + givenname)

     Note: The expression must conform to the syntax rules of a SiteMinder expression.

2. Create an *alias* attribute mapping for Directory B:
   - Name the mapping **SortName**.
   - Define SortName as **sort_name**.

Result:

You can reference SortName when defining policies, expressions, or other objects that require the sort name of users, without concern for the directory-specific schema, because the directories are operationally identical. SiteMinder determines that the sort name must be calculated when referencing users in Directory A and is sort_name when referencing users in Directory B.

Customer Use Case

This use case shows how you can use a g*roup membership* attribute mapping and a *constant* attribute mapping to identify customers in Directory A and Directory B.

Given:

1. Directory A stores employees, and because an employee of the company can also be a customer, Directory A identifies customers as those employees that belong to:

   cn=Customers,ou=Groups,o=acme.com

2. Directory B only stores customers. Directory B does not have a user attribute that identifies customers, because to store a user in Directory B implies that the user is a customer.

Solution:

1. Create a *group membership* attribute mapping for Directory A:
   - Name the mapping **IsCustomer**.
   - Define IsCustomer as:

     cn=Customers,ou=Groups,o=acme.com
2. Create a *constant* attribute mapping for Directory B:
   - Name the mapping **IsCustomer**.
   - Define IsCustomer as **TRUE**.

Result:

You can reference IsCustomer when defining policies, expressions, or other objects that must determine if the user is a customer, without concern for the directory-specific schema, because the directories are operationally identical. SiteMinder determines that a user is a customer if they belong to cn=Customers,ou=Groups,o=acme.com when referencing Directory A and that every user is a customer when referencing Directory B.

Account Status Use Case

This use case shows how you can use a *mask* attribute mapping and a *calculated expression* attribute mapping to identify user accounts that are disabled in Directory A and Directory B.

Given:

1. Directory A identifies disabled accounts with a user attribute named AccountStatus, which is a set of flags. The second bit indicates a disabled account.
2. Directory B identifies disabled accounts with a user attribute named u_disabled. When u_disabled is equal to "y", the account is disabled. When u_disabled is equal to "n", the account is active.

Solution:

1. Create a *mask* attribute mapping for Directory A:
   - Name the mapping **IsDisabled**.
   - Define the mapping as **AccountStatus:2**, where:
     - The bit pattern is stored in AccountStatus.
     - The bit mask is 2 (decimal).

2. Create a c*alculated expression* attribute mapping for Directory B:
   - Name the mapping **IsDisabled**.
   - Define the mapping as:

     ```
     (u_disabled = "y")
     ```

     where u_disabled is a Boolean expression.

Result:

You can reference IsDisabled when defining policies, expressions, or other objects that must determine the account status of users, without concern for the directory-specific schema, because the directories are operationally identical. SiteMinder checks the bit pattern to determine if a user is disabled when referencing Directory A and performs the calculation to determine if a user is disabled when referencing Directory B.

Chapter 8: 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.

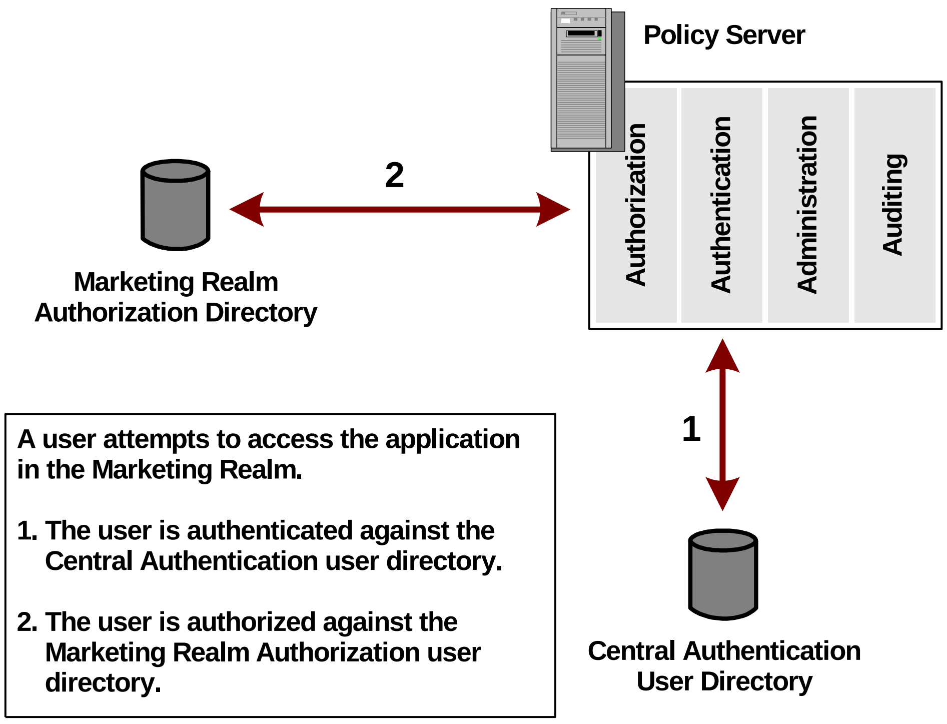
2. Configure a Directory Mapping

   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.


Policy Server
Authorization
Authentication
Administration
Auditing
2
Marketing Realm Authorization Directory
1
Central Authentication User Directory

A user attempts to access the application in the Marketing Realm.

1. The user is authenticated against the Central Authentication user directory.

2. The user is authorized against the Marketing Realm Authorization user directory.

More information:

How to Configure a CA Directory User Directory Connection (see page 135)
Configure a Directory Mapping (see page 240)
Realms (see page 377)
Advanced Policy Components for Applications (see page 507)

Directory Mapping Requirements

Directory mapping requires that the user directory connections to the Policy Server must already exist for the authentication directory, as well as the authorization or validation directory.

Note: More information on creating user directory connections exists in User Directory Connections Overview.

Supported Directory Mappings

The following table describes supported types of directory mapping, and the method that can be used to map the authentication directory to the authorization or validation directory.

	Authorization Directory/Validation Directory		
Authentication Directory	**LDAP**	**Relational Database**	**WinNT**
LDAP	Identical DN Universal ID	Universal ID	N/A
AD	Identical DN Universal ID	Universal ID	N/A
Relational Database	Universal ID	Identical DN Universal ID	N/A
WinNT	Universal ID	Universal ID	Identical DN

How to Configure an Authentication and Authorization Directory Mapping

Configuring an authentication and authorization directory mapping is a two-step process:

1. Configure the Directory Mapping
2. Assign an Authorization Directory to a Realm

Configure a Directory Mapping

You can configure a directory mapping to authenticate users against one directory and authorize users against another directory.

To configure a directory mapping

1. Click Infrastructure, Directory.
2. Click Auth/Az Mapping, Create Directory Mapping.

   The Create Directory Mapping pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Select the authentication and authorization directories from the respective lists.
4. Select the Identical DN or Universal ID radio button.
5. Click Submit.

   The Create Directory Mapping task is submitted for processing.

More information:

Universal IDs (see page 197)

Assign an Authorization Directory to a Realm

You assign a directory mapping to a realm so the Policy Server may authenticate a user in one directory and authorize a user in another directory. The Policy Server uses the authorization directory specified in the realm to authorize users.

To assign a directory mapping to a realm

1. Open the realm to which you want to assign a directory mapping.
2. Select the user directory for which the realm should use to authorize an authenticated user from the Directory Mapping list.

   Note: The Default value indicates that there is no directory mapping; the authentication directory will be used as the authorization directory when a user attempts to access a resource in the realm. The list only contains user directories that have been configured as authorization directories in an existing directory mapping.
3. Click Submit.

   The Policy Server saves the directory mapping. Users that access the realm authenticate normally and authorize against the directory specified in the realm.

More information:

Configure a Realm (see page 382)

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.

To configure an AuthValidate Directory Mapping

1. Click Infrastructure, Directory.
2. Click AuthValidate Mapping, Create AuthValidate Directory Mapping.

   The Create AuthValidate Directory Mapping pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Type the name of the directory that will be used to authenticate users in the Authentication Directory field.
4. Select the directory that will be used to validate users from the Validation Directory list.

5. Select the Identical DN or Universal ID radio button.
6. Click Submit.

   The Create AuthValidate Directory Mapping task is submitted for processing.

Assign an Authvalidate Directory Mapping to an Existing Realm

You can assign an AuthValidate directory mapping to an existing realm. This enables the Policy Server to authenticate the user against one user directory and validate the user against another user directory. When the Policy Server validates an authenticated user, the user is given access to a resource in the realm.

To assign an AuthValidate directory mapping to an existing Realm

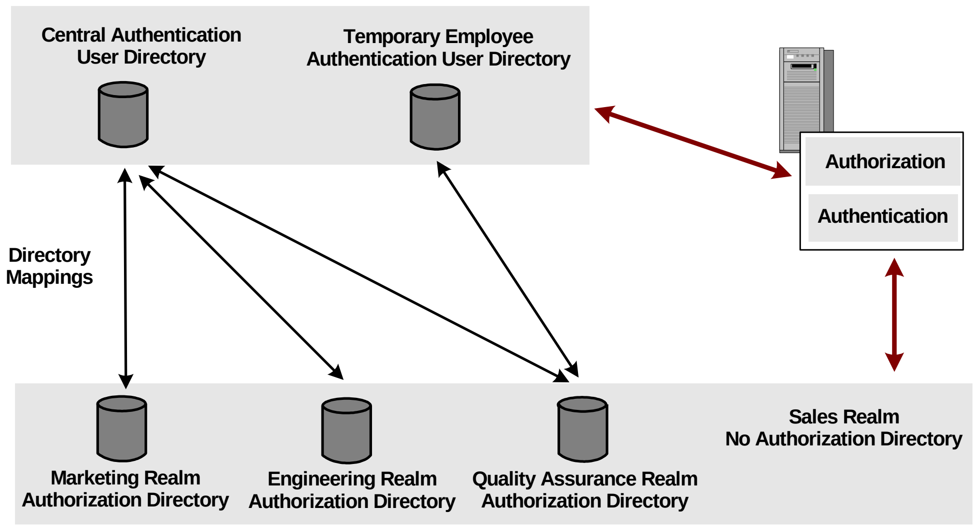
1. Click Policies, Domains, Realm, Modify Realm.
2. Select the realm that you want to modify.
3. Select the user directory to use as the validation directory from the Directory Mapping drop-down list.

   Note: The Directory Mapping drop-down list only contains user directories that are configured as authentication or validation directories in existing directory mappings. If the selected Directory Mapping is Default, the authentication directory is used as the validation directory when the user tries to access a resource in the realm.
4. Click Submit.

   The Modify Realm task is submitted for processing.

Directory Mapping Examples

Multiple directory mappings may be needed in order to correctly authenticate and authorize users that have access to network resources. The following figure illustrates a simple sample case where multiple directory mappings may be required.


Central Authentication User Directory
Temporary Employee Authentication User Directory
Authorization
Authentication
Directory Mappings
Sales Realm No Authorization Directory
Marketing Realm Authorization Directory
Engineering Realm Authorization Directory
Quality Assurance Realm Authorization Directory

In the example, there are three realms that have separate authorization user directories and one realm that does not have its own authorization user directory. User authentication information is maintained in two distinct authentication directories. The Policy Server uses one of the authentication directories based on where the user's information is stored and one of the authorization directories if the requested resource is in the Marketing, Engineering, or Quality Assurance realms.

More information:

Realms (see page 377)

Employee Accesses an Engineering Realm Resource

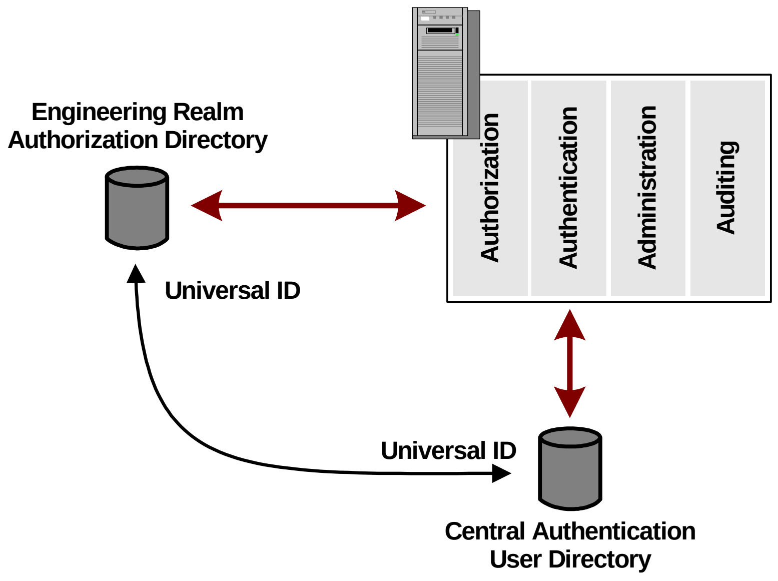
In the SiteMinder protected network described in the previous figure, a regular employee's authentication data resides in the Central Authentication user directory, and the employee attempts to access a resource in the Engineering Realm. When the employee is properly authenticated, the Policy Server 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).


Engineering Realm Authorization Directory
Authorization
Authentication
Administration
Auditing
Universal ID
Universal ID
Central Authentication User Directory

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.

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 413).

Chapter 9: 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.

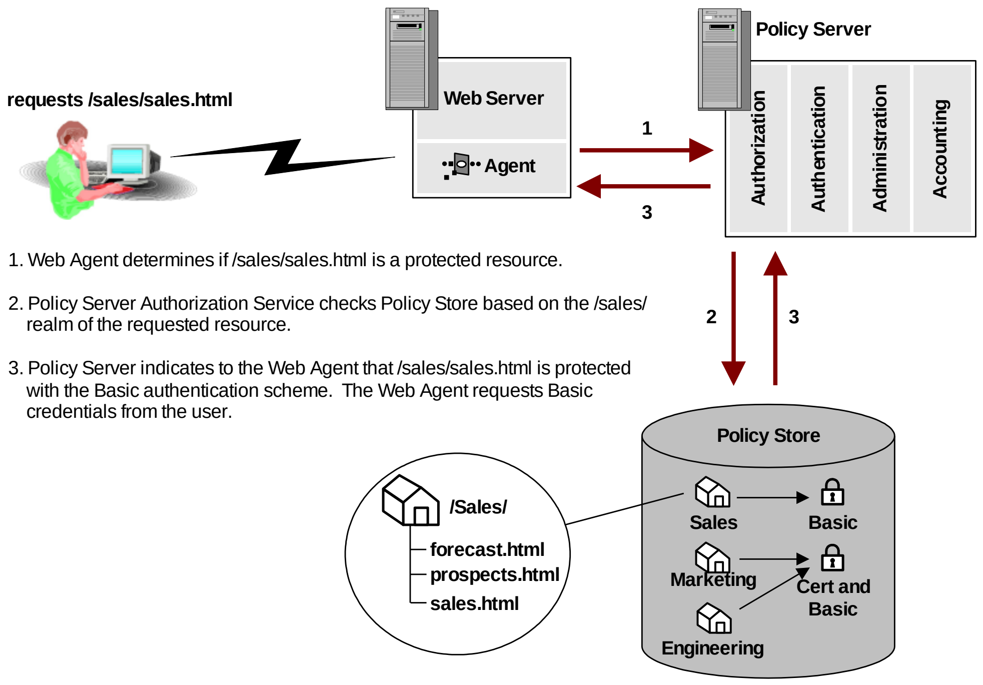
Authentication schemes must be configured using the Administrative UI. During authentication, SiteMinder Web Agents communicate with the Policy Server to determine the proper credentials that must be retrieved from a user who is requesting resources.

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.

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.


requests /sales/sales.html
Web Server
Agent
Policy Server
Authorization
Authentication
Administration
Accounting
1
3
2
3
1. Web Agent determines if /sales/sales.html is a protected resource.
2. Policy Server Authorization Service checks Policy Store based on the /sales/ realm of the requested resource.
3. Policy Server indicates to the Web Agent that /sales/sales.html is protected with the Basic authentication scheme. The Web Agent requests Basic credentials from the user.
Policy Store
/Sales/
forecast.html
prospects.html
sales.html
Sales
Basic
Marketing
Cert and Basic
Engineering

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 382)

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.

Token Authentication Schemes

Token authentication schemes generally rely on *two-factor* authentication. The first part of two-factor authentication requires a small device which may be connected to a PC, or may be an independent device that provides unique passwords. The hardware device provides *proof of possession*, since a user must have access to the device to authenticate. The second part of two-factor authentication provides *proof of knowledge*, generally in the form of a password.

In most cases, the passwords generated by the hardware components change at regular intervals. A user logs in and provides the authentication information specified by the hardware token. The Policy Server compares the information provided by the hardware token to the information provided by the token server component to verify a user's credentials.

The Administrative UI provides authentication scheme templates for the following hardware-based security tokens:

- CRYPTOCard RB-1
- Encotone TeleID

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.

Protection Levels

Authentication schemes require a protection level. This level ranges from 0 to 1000. A higher number indicates that the scheme provides higher level of protection. Protection levels allow single sign-on for Authentication Schemes of equal or lower protection levels within the same policy domain, while requiring additional authentication to access resources with higher protection level schemes.

Note: Anonymous authentication schemes always have a protection level of zero. A custom authentication scheme may have a protection level of 0-1000. All other authentication schemes may have a protection level of 1-1000.

For example, if you have a set resources that is available to all network users, you can assign a Basic (user name and password) authentication scheme with a low protection level such as 5. For revenue information that is available only to corporate executives, you can assign an X.509 client certificate scheme with a high protection level such as 15.

When users authenticate successfully against a scheme, they can access any resource with a protection level equal to or below the current authentication scheme without being challenged to authenticate a second time. However, users must still be authorized for a resource to gain access.

In the example above, a user who authenticated with a user name and password (protection level 5) would have to authenticate a second time with a digital certificate if he or she attempted to access the revenue information that was assigned the X.509 client certificate authentication scheme with the protection level of 15. However, if the user attempted to access resources for another department, which were also protected by a scheme of level 5, the user would not be challenged to authenticate a second time.

More information:

Domains (see page 369)
Policies (see page 439)

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			
CRYPTOCard RB-1	yes		yes		
Custom	optional	optional	optional	optional	optional
HTML Forms (over SSL optional)	custom credentials	custom credentials			optional
Impersonation	yes				optional
MS Passport	yes	yes			yes
NTLM or Windows	yes*	yes*			
RADIUS CHAP/PAP	yes	yes			
RADIUS Server	yes	yes			

	Credential Requirements				
Authentication Schemes	**Directory User Name**	**Directory Password**	**Code from Token**	**X.509 Certificate**	**User Profile Attributes**
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 (see page 329)

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 247)

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. For Windows NT directories, this is done via the LogonUser() function call. 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 258).

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 369)
Realms (see page 377)
Password Policies (see page 589)

Basic Scheme Prerequisites

Ensure 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 119)

Configure a Basic Authentication Scheme

You use a Basic authentication scheme to verify user identities against the user names and passwords that exist in the user directory.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select Basic Template from the Authentication Type Style list.
5. Enter a name and protection level in the General group box.
6. Click Submit.

   The authentication scheme is saved and may 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 250)
Password Policies (see page 589)

Basic over SSL Scheme Prerequisites

Ensure the following prerequisites are met before configuring a Basic over SSL authentication scheme:

- Client user name and password information 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 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 119)

Configure a Basic Over SSL Authentication Scheme

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select Basic over SSL Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and protection level in the General group box.

6. Enter a server name and target information in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may be assigned to a realm.

CRYPTOCard RB-1 Authentication Schemes

This scheme authenticates users who login via CRYPTOCard RB-1 hardware or software tokens. The scheme accepts user name and password credentials. The password is the dynamic code generated by the token in the Quick Log mode. The Administrative UI allows you to define multiple instances of this scheme.

CRYPTOCard RB-1 Scheme Prerequisites

Ensure the following prerequisites are met before configuring a Basic over SSL authentication scheme:

- The file containing CRYPTOCard token data is loaded into the Policy Server via the SiteMinder Token Tool. This data typically resides in the "Cryptocards" files used by the CRYPTOCard administrator utility.
- Users authenticating via this scheme must have the last eight digits of their card's serial number stored in one of the user attributes available in the user's native directory. The name of the attribute must be defined in the Windows Registry under:

  HKEY_LOCAL_MACHINE\Software\Netegrity\SiteMinder\CurrentVersion\Tokens\CryptoCard\SerialNumberUserAttribute\<*namespace*>\

  namespace

  LDAP:,ODBC:.

Configure a CRYPTOCard RB-1 Authentication Scheme

You use a CRYPTOCard RB-1 authentication scheme to authenticate users who login via CRYPTOCard RB-1 hardware or software tokens.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select CRYPTOCard RB-1 Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and protection level in the General group box.
6. Enter and confirm the secret CRYPTOCard is to use to synchronize tokens in the Scheme Setup group box.
7. 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*.

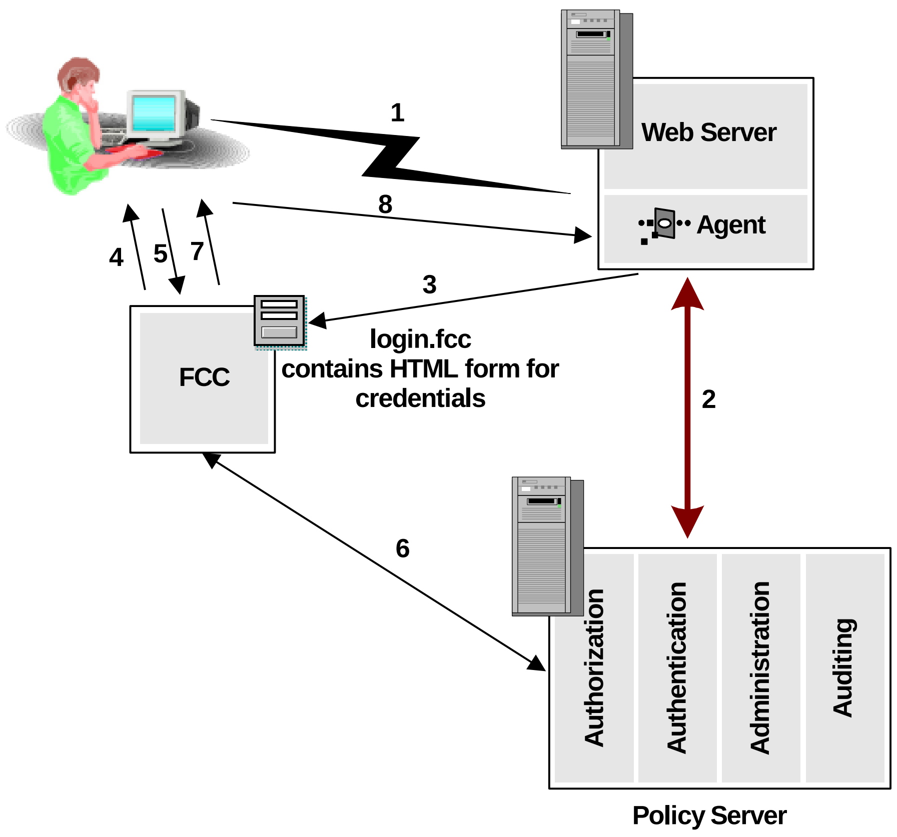
.unauth file

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 266)

HTML Forms Scheme Prerequisites

Ensure 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 SiteMinder makes an authentication request using forms.

More information:

SiteMinder FCC Files (see page 266)
User Directories (see page 119)

Custom Authentication Scheme Library Writing and Installation

The user name and password data collected by the FCC 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; it is distributed with the Policy Server and already installed on the Policy Server machine.

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 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 user name and password.

The FCC can be installed on the same machine as the Policy Server.

HTML Forms Authentication Templates

The templates used 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 you want in the FCCExtensions parameter of your Web Agent configuration file or object.

  Note: More information on Web Agent configuration parameters exists in the *Web Agent Guide*.

The SiteMinder 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

Language	Directory
Japanese	Windows: Web Agent\Samples\Formsja UNIX: webagent/samples/formsja

By default, SiteMinder uses the English .fcc files.

To use these sample .fcc files, create a Forms Authentication Scheme that specifies the appropriate target directory.

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

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 261)

How Name/Value Pairs are Generated in FCC Files

The LoginFCC generates a name space 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 name space 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. SiteMinder responses named in the “smheaders” name value pair (on Post only).

Special Name/Value Pairs

SiteMinder’s FCC can interpret a number of special name/value pairs (@directives) that invoke non-standard processing. The following are the special @directives and their meanings:

username

Name to use to as the login user name.

password

Password to use to perform the login.

target

Resource to access after login.

smheaders

Colon separated list of SiteMinder 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 SiteMinder transaction. For example, if you want to pass the value of header1 and header2 as part of a SiteMinder transaction, the following would be included in your FCC:

```
@smheaders=header1:header2
```

smerrorpage

If there is an error on a Post to the custom form, the user’s browser will be redirected to this page. If this special value is not specified in a .fcc file, SiteMinder uses the .unauth file 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 specified by smretries.

smpasswordfcc

Determines whether data is posted from the Password Services FCC file or from a different FCC file.

Default: 1

Important! It is recommended that the default value remain unchanged. 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 associated with a login failure.

smsavecreds

Set to Yes to save user credentials in a persistent cookie on the user’s 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 will be supplied to the Policy Server when a user enters credentials and submits the form for authentication. If the Agent parameter, FCCCompatMode=NO, you must specify a value using this directive.

smlogout

Logs a user out of SiteMinder, similar to logoffuri. By placing @smlogout=true in your .fcc template, the FCC will log a user out of SiteMinder and redirect the user to the target. As such, the @smlogout directive is typically used in conjunction with the @target directive *(@target=<yoururlhere>*).

urlencode(*name*)

Replaced by the URL encoded value of the named variable.

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 required by SiteMinder. When creating names for your login page do not use the "sm" prefix.

Localization Name/Value Pairs

.fcc files include two localization parameters:

SMLOCALE

Used by DMS and Self-Registration 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. DMS and Self-Registration use these localization properties files to generate language-specific HTML pages.

SMLOCALE values have the following format:

COUNTRY-LANGUAGE

For example, the value for **SMLOCALE** for United States English would be represented as follows:

SMLOCALE=US-EN

The value above corresponds to MSR_en_US.properties for DMS and Self-Registration.

SMENC

Contains information that tells the browser what language encoding to use. Changing the default value for this variable will override 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 username and password, you can collect additional attributes from users, such as a users email address or job title.

To collect additional attributes

1. Specify the attribute(s) by configuring the HTML Forms authentication scheme and the associated .fcc 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 (" . & = + ? ; / : @ = , $ %), you need to URL-encode each namespace in the .fcc file.

3. Modify the .fcc file to collect additional attributes.

   To modify the .fcc file, 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:

   ```
   @password=PASSWORD=%urlencode(PASSWORD)%&newattr1%=%urlencode(newattr1)%&newattr2=
   %urlencode(newattr2)%
   ```

   where newattr1=%newattr1%and newattr2=%newattr2% represent the additional attributes. The value before the equals 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 FCC, keep the following in mind:

- The attribute name, which is identified by the %*attribute_name*% format, must match the input name entry that you also add to the FCC. 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(a ddress)%

  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 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 261)

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. Although you can configure the smretries directive (@smretries) to provide users with additional login attempts, the default behavior does not let you display a message that informs users why the login failed.

The SiteMinder Web Agent is shipped with the DynamicRetry.fcc and DynamicRetry.unauth files. This sample 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 HTML page than the login page. 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 SiteMinder 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 on password policies exists in the Password Policies.

Configure an HTML Form Authentication Scheme

You use an HTML Forms authentication scheme to authenticate users with a custom HTML form.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select HTML Form Template from the Authentication Type Style list.

   Scheme-specific fields and controls open.
5. Enter a name and protection level in the General group box.
6. Enter a server name, target, and attribute list information in the Scheme Setup group box.

   Note: Ensure that the .fcc file you specify in the Target field complies with the guidelines listed in the prerequisites.
7. Click Submit.

   The authentication scheme is saved and may 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 non-browser 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 in with the initial request via an HTTP Authorization header or SiteMinder does not authenticate the users. If the credentials are not sent via an HTTP Authorization header, SiteMinder redirects to the HTML Form scheme without non-browser client support.

To enable non-browser client support

1. Open the HTML Form authentication scheme.
2. Select the Support non-browser clients check box.
3. Click Submit.

   Non-browser 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 79)

Kerberos Support

Kerberos is used for domain authentication in Windows 2000 with user and computer principals stored in Active Directory. Kerberos provides a platform independent architecture for authentication and single sign-on. However, this support is limited to Microsoft Web servers and browsers running Windows 2000, Windows XP and .NET.

SiteMinder supports Kerberos authentication indirectly using Windows authentication scheme. With SiteMinder 4.61, you can use Kerberos authentication only when the Web server is Microsoft IIS and the browser is Microsoft IE. SiteMinder Release 5.x supports using any SiteMinder Web Agent by allowing redirected NTLM authentication to an IIS server.

Users who login to their desktop using NT authentication, and use IE to access e-Business applications deployed on any Web server (including non-IIS Web servers), can login to SiteMinder without being re-challenged as long as there is one IIS web-server configured to use SiteMinder. This powerful capability allows the user to remember only their desktop password and still gives the enterprise the flexibility to choose the platform that is right for the application.

Windows Authentication Scheme Prerequisites

Ensure 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 you create in the Administrative UI specifies the WinNT namespace.
  - The requested resources are located on a Microsoft IIS Web server (4.0 or later) protected by a SiteMinder Web Agent.
- 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 are located on a Microsoft IIS Web server (4.0 or later) protected by a SiteMinder Web Agent.
  - Client and server accounts are enabled for delegation.
- Users must log in using Internet Explorer Web browsers (4.0 or later).
- To work on IIS6 in Windows 2003, the "Verified that file exists" option in the Wildcard Application Maps must not be set.
- Internet Explorer browser options are setup to allow automatic logon with a user's current username and password.

To configure automatic logon in Internet Explorer 5.x and 6.x Browsers

1. From the menu bar in Internet Explorer, select Tools, Internet Options.
2. The Internet Options dialog opens.
3. Click the Security tab to bring it to the front.

4. Select your Internet zone and click Custom Level.
5. The Security Settings dialog opens.
6. Scroll down to User Authentication, Logon.
7. Select the Automatic logon with current username and password radio button.
8. Click OK.

To configure automatic logon in Internet Explorer 4.x Browsers

1. From the menu bar in Internet Explorer, select View, Internet Options.
2. The Internet Options dialog opens.
3. Click the Security tab to bring it to the front.
4. Select your Internet zone from the drop down list.
5. In the Internet zone group box, select the and click Custom radio button and click Settings.
6. The Security Settings dialog opens.
7. Scroll down to User Authentication, Logon.
8. Select the Automatic logon with current username and password radio button.
9. Click OK.

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 use a Windows authentication scheme to authenticate users in a pure Windows environment.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select Windows Authentication Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and protection level in the General group box.
6. Enter a server name, target, and user DN information in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may 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 and 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).

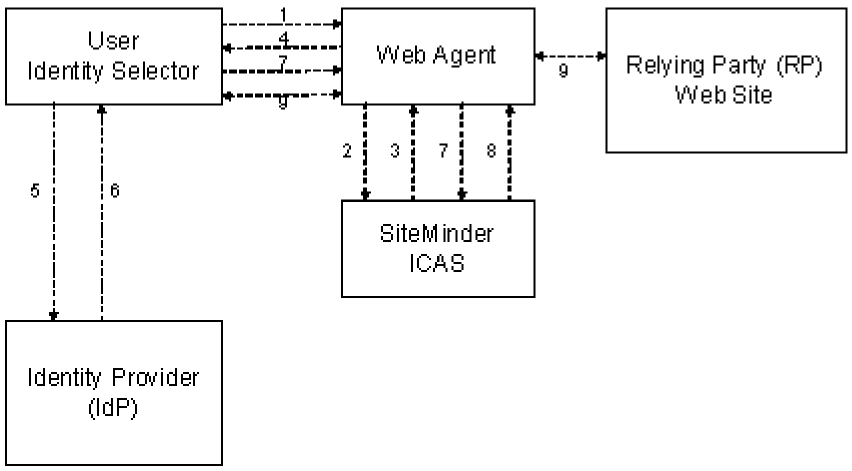
SiteMinder Information Card Authentication Scheme (ICAS)

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 are required by SiteMinder 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 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:

- Configure an ICAS Properties File
- How to Configure the SiteMinder Key Database
- Configure a User Directory for ICAS
- Create an Instance of ICAS
- Configure 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.

To configure the Java Runtime Environment (JRE) for ICAS

1. Download the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction policy files from http://java.sun.com/javase/downloads/index_jdk5.jsp and install them in the $NETE_JRE_ROOT\lib\security directory.
2. 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:

- Configure an ICAS Properties File
- Store Claims for Later Use in Active Responses
- How to Configure the SiteMinder Key Database for ICAS
- Configure a User Directory for ICAS
- Create an Instance of ICAS
- Configure an Active Response that Retrieves a Claim Value

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 fcc file.

Example:
fcc=https://*web_server_home*/siteminderagent/forms/infocard.fcc

Note: To activate the Identity Selector, you must specify "https".

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/emailaddress

alias

Specifies the key in the SiteMinder key store that is used to retrieve the Relying Party's SSL certificate.

Example: alias=rpssl

tokenPrim

Specifies the provider of the tokenPrim interface.

Example:
tokenPrim=com.ca.sm.authscheme.infocard.higgins.TokenAdapter

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

How to Configure the SiteMinder Key Database for ICAS

The Relying Party must use SSL to protect the fcc file. The Relying Party must export the SSL certificate associated with the web site to a pfx file. A SiteMinder administrator can then import the SSL certificate from the pfx file into smkeydatabase using smkeytool.

When the certificate is imported into smkeydatabase, it is associated with an alias, which is stored in the fcc file. The certificate's private key is used to decrypt the security token and verify the digital signature.

Configuring the SiteMinder key database is a two-step process:

1. To export an SSL certificate from an IIS web server to a pfx file on your local machine, you can use the Web Server Certificate Wizard. For more information, see Microsoft's documentation.

2. To import an SSL certificate from a pfx file into smkeydatabase using smkeytool, execute smkeytool.bat, specifying the options in the following example:

   smkeytool.bat -addPrivKey -alias example -keycertfile c:\Temp\www-example-com.pfx

   -password *CAdemo123*

   addPrivKey

   Specifies the action that you want smkeytool to take

   alias

   Specifies a name for the SSL certificate in smkeydatabase

   Note: This is the alias that is specified in the properties file.

   keycertfile

   Specifies the location of the pfx file on your local machine

password

Specifies the password that you provided when exporting the SSL certificate to the pfx file

Note: 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.

Note: If smkeydatabase does not exist, you can create it using the Policy Server Configuration Wizard. For more information, see the *Policy Server Installation Guide*.

Note: For more information about smkeydatabase and smkeytool, see the *Federation Security Services Guide*.

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

Create an instance of ICAS by specifying a custom authentication scheme in the Administrative UI.

Limit: Each policy store can support up to 10 instances of ICAS.

To create an instance of ICAS

1. Click Infrastructure, Authentication, Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme search pane opens.

2. Select Create a new object of type Authentication Scheme, and click OK.

   The Create Authentication Scheme: *Name* pane opens.

3. Type the authentication scheme's name and description in the fields on the General group box.

4. Select Custom Template from the drop-down list of Authentication Scheme Types on the Scheme Common Setup group box.

   The Advanced group box is replaced by the Scheme Setup group box.

5. Type a value in the Protection Level field on the Scheme Common Setup group box.

6. Clear the Password Policies check box on the Scheme Common Setup group box.

   Note: Password Policies are not supported by ICAS.

7. Type the following values in the fields on the Scheme Setup group box:

   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

8. 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 server. For more information about session servers, see the *Policy Server Administration Guide*.

To configure an active response that retrieves a claim value

1. Click Policies, Domains, Response, Create Response.

   The Create Response: Select Domain pane opens.

2. Select a domain, and click Next.

   The Create Response: Define Response pane opens.

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

4. Specify Web Agent as the Agent Type on the Attributes group box.

5. Click Create Response Attribute on the Attribute List group box.

   The Create Response Attribute: *Name* pane opens.

6. Select WebAgent-HTTP-Header-Variable or WebAgent-HTTP-Cookie-Variable from the drop-down list of attributes on the Attribute Type group box.

7. Select Active Response as the Attribute Kind on the Attribute Setup group box.

8. Type the following values in the Attribute Fields on the Attribute Setup group box.

   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

9. Click OK.

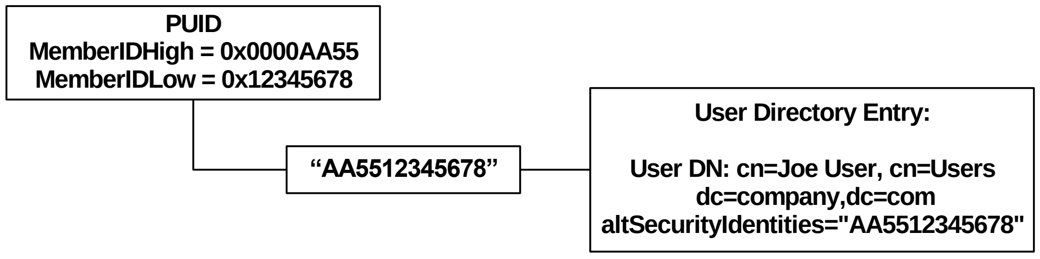
   The Create Response task is submitted for processing.

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".


PUID
MemberIDHigh = 0x0000AA55
MemberIDLow = 0x12345678
“AA5512345678”
User Directory Entry:
User DN: cn=Joe User, cn=Users dc=company,dc=com
altSecurityIdentities="AA5512345678"

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 SP1 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.

More information:

Protection Levels (see page 253)

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 an MS Passport Authentication Scheme

You use an MS Passport authentication scheme to authenticate users across participating .NET Passport Web sites.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select MS Passport Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and protection level in the General group box.
6. Enter configuration information in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may 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

Ensure 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.

Configure a RADIUS CHAP/PAP Authentication Scheme

You use a RADIUS CHAP/PAP authentication scheme when you are using the RADIUS protocol.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select RADIUS CHAP/PAP Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and protection level in the General group box.
6. Specify the clear text password in the Scheme Setup group box.
7. 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

Ensure the following prerequisites are met before configuring a RADIUS Server authentication scheme:

- The RADIUS server is on a network accessible by the Policy Server.

Configure a RADIUS Server Authentication Scheme

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select RADIUS Server Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and a protection level in the General group box.
6. Enter the RADIUS server IP address, port number, and shared secret in the Scheme Setup group box.
7. 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 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 you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select SafeWord Template from the Authentication Type Style list.

   Scheme-specific fields and controls open.
5. Enter a name and protection level in the General group box.
6. Enter the location of the SafeWord server configuration file in the Scheme Setup group box
7. Click Submit.

   The authentication scheme is saved and may 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 266)
User Directories (see page 119)

Configure a SafeWord Server and HTML Forms Authentication Scheme

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select SafeWord HTML Form from the Authentication Type Style list.

   Scheme-specific fields and controls open.
5. Enter a name and protection level in the General group box.
6. Enter the server name, the location of the SafeWord configuration file, and server and target information in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may 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.

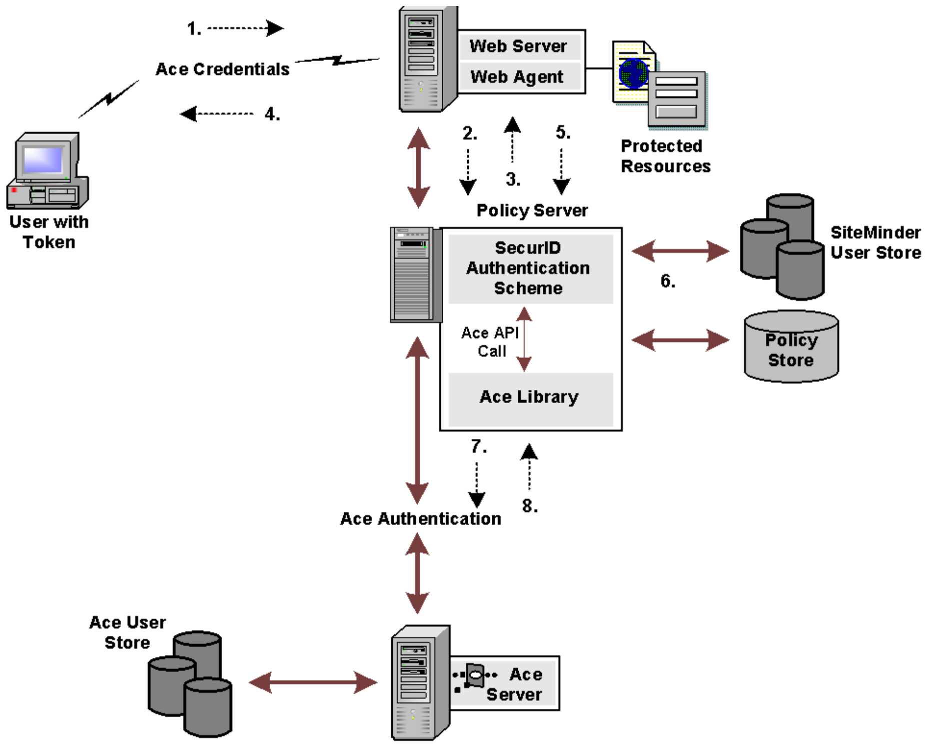
The RSA ACE/Server works with RSA SecurID tokens to authenticate users’ identities through valid RSA Ace/Agents. Further, RSA supports Web Agents and custom Agents. SiteMinder SecurID authentication schemes act as custom ACE/Agents and use ACE/SDK and libraries.

The ACE/Server, however, has it’s 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

Ensure 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 machine as the Policy Server. For information about supported RSA ACE/Client versions, see the Platform Support Matrix on the Support site.

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 use a SecurID authentication scheme to authenticate users logging in with ACE credentials.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select SecurID Template from the Authentication Type Style list.

   Scheme-specific fields and controls open.
5. Enter a name and a protection level in the General group box.

6. Enter the ACE User ID attribute in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may 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>*/bin directory. In addition, the VAR_ACE and USR_ACE variables are pointing to the *<policy server installation dir>*/bin, 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 266)
User Directories (see page 119)

Configure a SecurID and HTML Forms Authentication Scheme

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select SecurID HTML Form Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and a protection level in the General group box.
6. Enter server, target, and ACE attribute information in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may 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.

TeleID Authentication Schemes

The TeleID authentication scheme allows SiteMinder to authenticate users who login using passwords generated by Encotone's TeleID hardware tokens. The scheme accepts a user name and password. The password is the dynamic code generated by the token in the *Code* or *Signature* mode. The user name can be omitted if the password field contains the Signature. Multiple instances of this scheme may be defined.

TeleID Scheme Prerequisites

Ensure the following prerequisites are met before configuring a TeleID authentication scheme:

- The file containing token data is loaded into the Policy Server via the SiteMinder Token Tool.
- Users authenticating via this scheme have their card's serial number stored in one of the user attributes available in the user's native directory.
- For WinNT, the name of the directory attribute must be defined in the WinNT registry under:

  "HKEY_LOCAL_MACHINE\Software\Netegrity\SiteMinder\CurrentVersion\Tokens\Encotone\SerialNumberUserAttribute\<*namespace*>"

  namespace

  LDAP:, ODBC:.

Configure a TeleID Authentication Scheme

You use a TeleID authentication scheme to authenticate users who login using passwords generated by Encotone's TeleID hardware tokens.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select TeleID from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and a protection level in the General group box.

6. Enter and confirm encryption seed information in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may be assigned to a realm.

X.509 Client Certificate Authentication Schemes

X.509 client certificates provide cryptographic proof of a user's identity. The certificate, supplied by a certificate vendor, is unique, and can be used to identify the user who attempts to access a protected resource.

The certificate contains the following information:

- Name of the subscriber (user)—This is a unique name called the distinguished name.
- Public key of the subscriber
- Operational period for the certificate
- Name of the Certificate Authority that issued the certificate
- Certificate serial number

The X.509 Client Certificate authentication scheme is the method SiteMinder uses to implement certificate authentication. SiteMinder's X.509 Client Certificate authentication consists of two parts:

- Establishing an SSL connection and collecting the client certificate information.
- Identifying a user in a directory service based on the client certificate and optionally verifying this certificate with the directory.

How SiteMinder Establishes an SSL Connection for Certificates

The SiteMinder Web Agent establishes an SSL connection by redirecting the user's browser to an HTTPS:// URL pre-configured by SiteMinder. The SSL connection to this URL is set up to require a client certificate to be submitted by the browser. The SiteMinder component invoked at this URL extracts information from the certificate delivered as part of the SSL connection and conveys this information to the SiteMinder Web Agent.

More information:

Authentication over SSL (see page 253)

How SiteMinder Identifies Users Based on Certificates

Once SiteMinder establishes an SSL connection between the SSL certificate server and the user, the SiteMinder Web Agent collects the user's certificate information and passes it to the Policy Server for verification. The certificate is then mapped to a user DN in a directory. For LDAP directories, you can configure the Policy Server to verify that the same certificate is associated with the user DN in the directory and to check the appropriate Certificate Revocation List (CRL) to ensure the certificate has not been revoked. For WinNT directories you can configure the Policy Server to check CRLs.

X.509 Client Certificate Scheme Prerequisites

Ensure the following prerequisites are met before configuring a X.509 Client Certificate 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.

Note: For Apache Web servers where Certificates are required or optional, uncomment the SSL Verify Depth 10 line in the httpd.conf file.

Configure an X.509 Certificate Authentication Scheme

You use an X.509 Certificate authentication scheme to implement certificate authentication.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select X509 Client Cert Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and a protection level in the General group box.
6. Enter the server name and target information for the SSL Credentials Collector in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may be assigned to a realm.

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 258)
X.509 Client Certificate Authentication Schemes (see page 251)

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 119)

Configure an X.509 Certificate and Basic Authentication Scheme

You use an X.509 Certificate and Basic authentication scheme to combine certificate and basic authentication.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select X509 Client Cert and Basic Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and a protection level in the General group box.
6. Enter the server name and target information for the SSL Credentials Collector in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may 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's identity. In order for a user to authenticate successfully, one of the following two events must occur:

- The user's 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 WebAgent challenges the user's 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 to obtain a user name and password.

This scheme is useful if you need to 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 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 chooses not to use the certificate-based authentication for accessing a resource in a realm protected by a Certificate or Basic authentication scheme, the browser (both IE and Netscape) automatically caches this decision not to send the certificate. 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 will receive a " Forbidden " error message.

Since 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.

Users who have valid certificates should be encouraged 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 or not to send a certificate for authentication.

More information:

Basic Authentication Schemes (see page 250)
X.509 Client Certificate Authentication Schemes (see page 251)

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 119)

Configure an X.509 Certificate or Basic Authentication Scheme

You use an X.509 Certificate or Basic authentication scheme to implement certificate and/or basic authentication.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select X509 Client Cert or Basic Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and a protection level in the General group box.
6. Enter server and target information for the SSL Credentials Collector in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may 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 261)
X.509 Client Certificate Authentication Schemes (see page 251)

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 119)

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 use an X.509 Certificate and HTML authentication scheme to combine certificate and HTML forms-based authentication.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select X509 Client Cert and Form Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and a protection level in the General group box.
6. Enter the server name and target information for the SSL Credentials Collector in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may be assigned to a realm.

X.509 Client Certificate or HTML Forms Authentication Schemes

The X.509 Client Certificate or HTML Forms authentication scheme allows either HTML Forms authentication or X.509 Client Certificate authentication to establish a user's identity. In order for a user to authenticate successfully, one of the following two events must occur:

- The user's 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's browser to present a certificate, after which the scheme has the following effect:

If...	then...
There is a certificate presented	SiteMinder processes the certificate
The certificate is not accepted	SiteMinder issues error 500
There is no certificate presented	SiteMinder presents a form
The form is rejected	SiteMinder prompts again for a form

This scheme is useful if you need to 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 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 chooses not to use the certificate-based authentication for accessing a resource in a realm protected by a Certificate or Forms authentication scheme, the browser (both IE and Netscape) automatically caches this decision not to send the certificate. 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 will receive a " Forbidden " error message.

Since 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.

Users who have valid certificates should be encouraged 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 or not to send a certificate for authentication.

More information:

HTML Forms Authentication Schemes (see page 261)
X.509 Client Certificate Authentication Schemes (see page 251)

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 119)

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 use an X.509 Certificate or HTML Forms authentication scheme to implement certificate and/or HTML forms-based authentication.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select X509 Client Cert or Form Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and a protection level in the General group box.
6. Enter server and target information in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may 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 377)

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 119)

Configure an Anonymous Authentication Scheme

You use an Anonymous authentication scheme to give non-registered users access to specific Web content.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select Anonymous Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and a protection level in the General group box.
6. Enter the DN of a user in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may 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 655).

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 use an Custom authentication scheme when you need to use a scheme that SiteMinder does not provide.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select Custom Template from the Authentication Type Style list.

   Scheme-specific settings open.
5. Enter a name and a protection level in the General group box.
6. Enter the library that is to process the credentials for the authentication scheme and the parameters that are passed to the library in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may be assigned to a realm.

Federation Security Services Authentication Schemes

If you have purchased and installed Federation Security Services, the following authentication schemes are added to the existing SiteMinder schemes:

- SAML Artifact Template
- SAML POST Template
- SAML 2.0 Template
- WS-Federation Template

Note: More information about these authentication schemes exists in the *Federation Security Services Guide*.

SOA Security Manager Authentication Schemes

If you have purchased and installed SOA Security Manager, the following authentication schemes are added to the existing SiteMinder schemes:

- XML Document Credential Collector (XML DCC)
- XML Digital Signature (XML-DSIG)
- SAML Session Ticket
- WS-Security

For more information about these authentication schemes, see the *CA SOA Security Manager Policy Configuration Guide*.

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 261)
Impersonation (see page 567)

Impersonation Scheme Prerequisites

Ensure 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 266). For information about specific .fcc file requirements for impersonation, see Enable Impersonation through an .fcc File (see page 577).
- Directory connections exist between the Policy Server and the user directories containing impersonators and impersonatees.

- The default HTML forms library is installed. This library handles authentication processing:
  - smauthimpersonate.dll on Windows
  - smauthimpersonate.so on Solaris

  These files are installed automatically when you configure a Web Agent.

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.

Note: If a password policy applies to an impersonatee and is in effect, the Impersonation authentication scheme fails.

More information:

User Directories (see page 119)

Configure an Impersonation Authentication Scheme

You use an Impersonation authentication scheme to let privileged users impersonate other users.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select Impersonation Template from the Authentication Type Style list.

   Scheme-specific fields and controls open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
5. Enter a name and a protection level in the General group box.

6. Enter the server name and target information in the Scheme Setup group box.
7. Click Submit.

   The authentication scheme is saved and may be assigned to a realm.

Certificate Mapping

X.509 client certificates provides strong user authentication. However, in order for SiteMinder to use a certificate to identify a user, the certificate must be compared to a user's information in a directory. SiteMinder uses a certificate mapping to determine how to compare a user's certificate with the information stored in the user directory.

SiteMinder supports certificate mapping for users whose authentication information is stored in a WinNT, Microsoft SQL Server, Oracle, or LDAP user directory. A certificate mapping defines how data in the certificate is mapped to form a user Distinguished Name (DN). The Policy Server uses this user DN to authenticate the user.

If certificates are stored in an LDAP directory, a certificate mapping can direct the Policy Server to verify that the certificate presented by the user matches the certificate associated with the user DN in the LDAP directory.

More information:

User Directories (see page 119)

Configure a Certificate Mapping

You can configure a certificate mapping that lets SiteMinder determine how to compare a user's certificate with the information stored in the user directory.

To configure an Impersonation authentication scheme

1. Click Infrastructure, Directory.
2. Click Certification Mapping, Create Certificate Mapping.

   The Create Certificate Mapping pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

3. Type the certificate issuer DN in the Issuer DN field.

   Note: You must escape reserved special characters with a backslash (\). More information on reserved special characters for LDAP distinguished names exists at http://www.faqs.org/rfcs/rfc2253.html. Special characters include:

   - commas (,)
   - semicolons (;)
   - quotes (")
   - backslashes (\)
   - plus character (+)
   - greater than character (>)
   - less than character (<)

   Note: Issuer DNs cannot exceed 255 characters if a relational database is used as a policy store; Issuer DNs cannot exceed 1000 characters if an LDAP directory server is used as a policy store.

4. Specify how the X.509 client certificate is to map user authentication information in the authentication directory on the Mapping group box.
5. (Optional) Select Perform CRL Checks on the Certificate Revocation List (CRL) Checking group box, and specify the CRL settings on the group box.
6. Click Submit.

   The Create Certificate Mapping task is submitted for processing.

Test a Certificate Mapping

Testing a certificate mapping displays the search string the Policy Server is to use to map client certificates to user directory attributes.

To test a certificate mapping

1. Open the certificate mapping.
2. Click Test in Mapping group box.

   The Certificate Map Test group box opens.

3. Select a user directory connection from the Directory list.

   Note: The Directory list includes all of the existing directory connections of the type you selected when creating the certificate mapping.

   The contents of the Directory Information group box change based on the type of user directory connection. For WinNT, ODBC and OCI user directory connections, the group box displays the Directory Type you are testing. For LDAP directory connections, the group box displays the Directory Type, as well as the Lookup Start and Lookup End values used to locate a user's DN within the LDAP directory.

   The Policy Server tests the certificate mapping and the Certificate Map Test group box provides the results.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

4. Click Close.

   The Certificate Map Test group box closes.

Confirm the Validity of Certificates

SiteMinder can confirm the validity of certificates using either of the following two methods:

- Certificate Revocation List (CRL) checking
- Online Certificate Status Protocol (OCSP) checking

Certificate Revocation List Checking,

A certificate revocation list (CRL) is a list of revoked X.509 client certificates published by the Certificate Authority (CA) to an LDAP user directory. Comparing certificates against CRLs is one method of ensuring that certificates are valid.

Note: The Policy Server can support CRLs greater than 1.7 MB in size, but cannot verify the status of a certificate using a Certificate Revocation List (CRL) that is larger than 64 KB. This limit is due to the third party libraries that are used to parse CRLs.

SiteMinder compares certificates against CRLs stored in an LDAP directory. SiteMinder verifies the signature of the CRL by retrieving the CA public certificate from the LDAP directory. SiteMinder supports the following RSA algorithms for signature verification:

- MD5
- MD2
- SHA1

CA administrators must keep CRLs up to date. If SiteMinder retrieves an expired CRL, all certificates from the CA with the expired CRL will be denied access. If multiple CRLs exist, SiteMinder will search for and use the most recent CRL. If a CA's public certificate is not available or your CRL is signed with an unsupported algorithm, you can turn off signature checking during the CRL verification process.

Note: If signature checking is turned off, make sure that the LDAP directory is protected appropriately.

Configure Certificate Revocation List Checking

You configure CRL checking to ensure that a user with an invalid client certificate cannot access a protected resource.

To configure CRL checking

1. Open the certificate mapping.
2. Select Perform CRL Checks from the Certificate Revocation List Checking group box.

   CRL-specific fields and controls open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Select the LDAP user directory that contains the CRL from the CRL Directory list.

   Note: If the LDAP user directory is not in the list, click Create to add the new directory connection.
4. Complete the remaining settings, as necessary, and click Submit

   Certificate revocation list checking is enabled.

Online Certificate Status Protocol Checking

The Online Certificate Status Protocol (OCSP) lets OCSP-enabled applications determine the revocation state of an X.509 client certificate in a more timely manner than is possible with Certificate Revocation Lists. Certificate Revocation Lists can get large and their propagation can become slow. OCSP checking provides real-time status information and lessens network traffic significantly.

During certificate checking, the Policy Server looks for the existence of an Issuer DN in a configuration file: smocsp.conf. If the Issuer DN is found, a certificate status check is made using a certified OCSP 1.0 Responder, which is specified in the smocsp.conf file. If the Issuer DN is not found in the configuration file, the certificate is considered to have passed OCSP checking.

OCSP checking requires:

- An established CA environment
- An OCSP Responder
- One or more LDAP directories containing an OCSP Responder and Certificate Authority public certificates

Note: The Policy Server does not support the OCSP on a Linux platform.

Configure Online Certificate Status Protocol Checking

You configure OCSP checking to ensure that a user with an invalid client certificate cannot access a protected resource.

To configure OCSP certificate status checking

1. Ensure that CRL checking is not enabled for the certificate mapping.
2. Create an smocsp.conf file in the Policy Server config directory:

   siteminder_installation_dir/config

The smocsp.conf file must be an ASCII file containing one or more OCSPResponder records, each with the following format:

```
[
OCSPResponder
IssuerDN <IssuerDN>
[AlternateIssuerDN <IssuerDN>]
CACertDir <Name of User Dir containing CA cert>
CACertEP <Entry point in CACertDir containing CA cert>
ResponderCertDir <Name of User Dir containing Responder cert>
ResponderCertEP <Entry point in ResponderCertDir containing Responder cert>
ResponderCertAttr <Directory attribute of Responder cert>
ResponderLocation <Server-name of Responder:port #>
AIAExtension<YES|NO>

]
```

Consider the following when creating the smocsp.conf file:

- Each OCSPResponder record must start with the tag name OCSPResponder.
- Attribute names are not case sensitive.
- The user directory is the name of the user directory connection listed in the Administrative UI, not the IP address of the physical directory.
- The existence of a value for IssuerDN indicates the existence of an OCSP Responder record|object.

- CACertDir and ResponderCertDir directories should exist as user directories in the Policy Server database.
- All attributes except for AIAExtension and ResponderLocation are required.

  Note: ResponderLocation should exist or AIAExtension should be YES.
- AIAExtension is defaulted to NO.

The priority of AIAExtension and ResponderLocation is as follows:

If	Then
AIAExtension is YES	The AIAExtension is used for validations, if it is found in the certificate. Otherwise, the ResponderLocation is used.
AIAExtension is NO	The ResponderLocation is used, regardless of the value of AIAExtension in the userCertificate in the request.

Following is an example of a smocsp.conf file:

```
[
OCSPResponder IssuerDN C=US,O=U.S. Government,OU=DoD,OU=PKI,CN=DOD CLASS 3 CA-9
CACertDir localhost:389
CACertEP cn=DOD CLASS 3 CA-9,ou=PKI,ou=DoD,o=U.S. Government,c=US   ResponderCertDir
localhost:389
ResponderCertEP cn=OCSP,ou=PKI,ou=DoD,o=U.S. Government,c=US ResponderCertAttr cacertificate
ResponderLocation aristotle.jfcom.mil:80
]
```

More information:

Configure Certificate Revocation List Checking (see page 332)

Custom Mapping Expressions

You can use custom mapping expressions for complex multiple attribute mapping. This allows you to specify multiple user attributes that should be extracted from a user DN to establish a certificate mapping.

Note: Custom mapping expressions are also useful when simulating certificate-based authentications through authentication the 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 636)

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.

Example 1

If a user's certificate contains:

SubjectDN: CN=John Smith, UID=JSMITH, OU=development, O=CompanyA

You can specify the following custom mapping:

CN=%{UID}, OU=%{OU}, O=%{O}

The resulting UserDN is:

CN=JSMITH, OU=development, O=CompanyA

Example 2

The custom mapping syntax also handles more complex mappings, as illustrated in the example:

If the user's certificate contains:

Subject DN: CN=John Smith + UID=jsmith +EMAIL=jsmith@companyA.com, ou=development, o=companyA

You can specify the following custom mapping:

CN=%{CN.CN}+UID=%{CN.UID}, OU=%{O}

The resulting UserDN is:

CN=John Smith+UID=JSMITH, OU=companyA

In the above example, the CN contained multiple attributes. The syntax indicated which components of the CN to take and apply to the UserDN's CN. This was done by specifying "CN.CN or CN.UID" This syntax indicates that the custom expression uses both the CN and UID parts of the CN.

Note: You cannot use the "+" operator to disambiguate multiple attributes in a user directory. The "+" operator is used like any other character in the user DN for a user that is present in the user directory.

Example 3

Static text can be represented in a custom expression by leaving it outside of the bracket notation as shown below.

The user’s certificate contains:

Subject DN: CN=John Smith, UID=JSMITH, OU-development

You can specify the following mapping:

CN=%{UID}, OU=%{OU}, O=companyA

The resulting UserDN is:

CN=JSMITH, OU=development, O=CompanyA

For more information, see the next section.

Template String Usage

The template string is composed of text and hash-bracketed expressions %{…}. All text outside the brackets is returned unchanged. The hash-bracketed expressions are evaluated based on the following rules:

- Undistinguished variable names (e.g. DN) are resolved before being returned.
- Distinguished variable names (for example, DN.UID) are resolved to the variable component before being returned.

Map to the Certificate Serial Number or IssuerDN

Certificate Mapping supports mapping of the CertSerialNumber and IssuerDN attributes, which are not part of the subjectDN. These attribute(s) in the subjectDN of user certificates can be mapped to default or custom user-attribute(s), such as UID or CN in the user directory.

To map these attributes, add the following in the Mapping Expressions field in the Certificate Mapping pane:

- CustomAttributeinLDAP1 = %{CertSerialNumber}

Custom Certificate Mapping for Multiple Attributes of the Same Type

Some certificates may have multiple attributes of the same type in their Subject DN. 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}.

Chapter 10: Strong Authentication

This section contains the following topics:

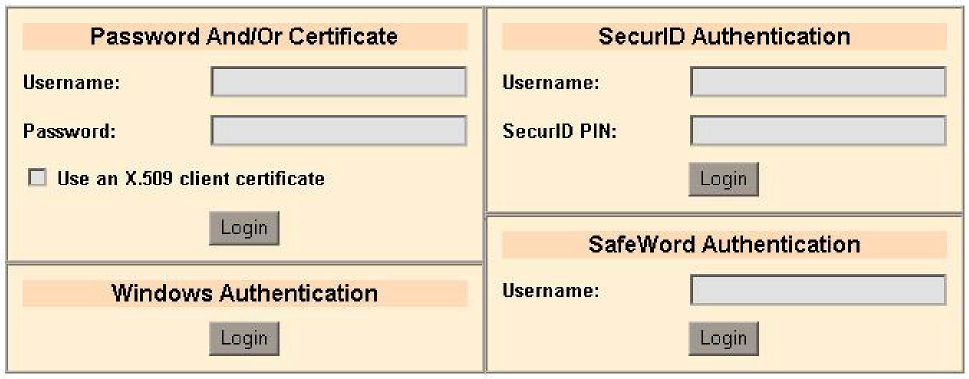
Credentials Selector Introduction

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 342)
Request Access with Windows Authentication (see page 343)
Request Access with SecurID Authentication (see page 343)
Request Access with SafeWord Authentication (see page 344)

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 348)

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];
        }
      }
```

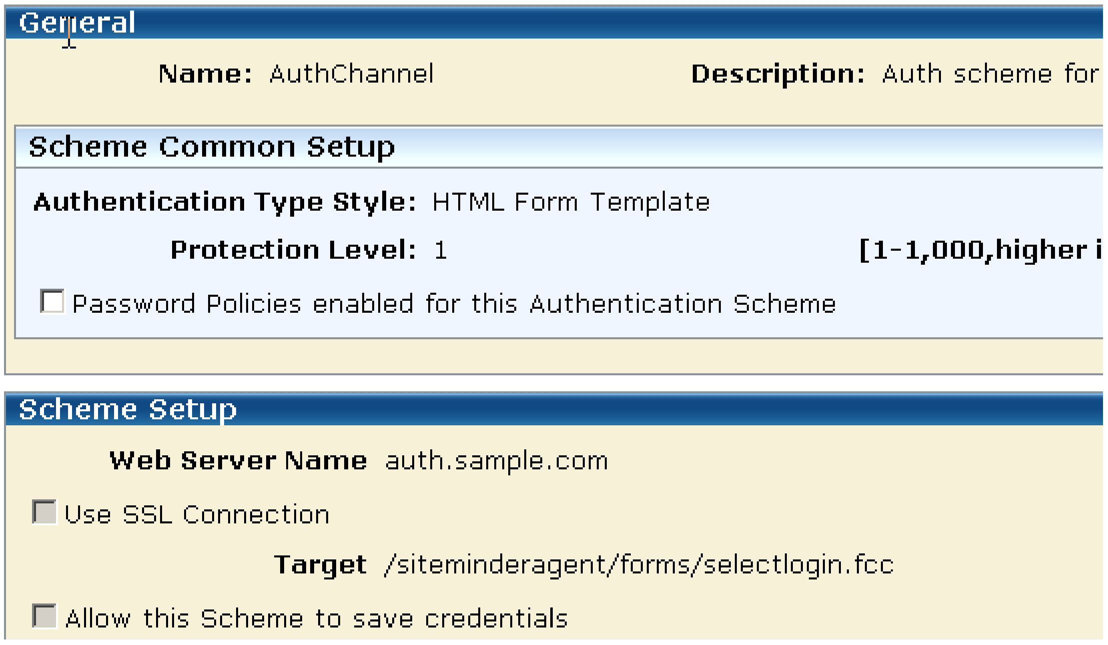
```
else if (form == 2)
{
   // SecurID authentication
   authtype = "securid";
   document.Login.USER.value       = document.Login.USER2.value;
   document.Login.PASSWORD.value = document.Login.PASSWORD2.value;
}
else if (form == 3)
{
   // SafeWord authentication
   authtype = "safeword";
   document.Login.USER.value       = 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.


General

Name: AuthChannel **Description:** Auth scheme for

Scheme Common Setup

Authentication Type Style: HTML Form Template

Protection Level: 1 **[1-1,000,higher i**

☐ Password Policies enabled for this Authentication Scheme

Scheme Setup

Web Server Name auth.sample.com

☐ Use SSL Connection

Target /siteminderagent/forms/selectlogin.fcc

☐ Allow this Scheme to save credentials

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=

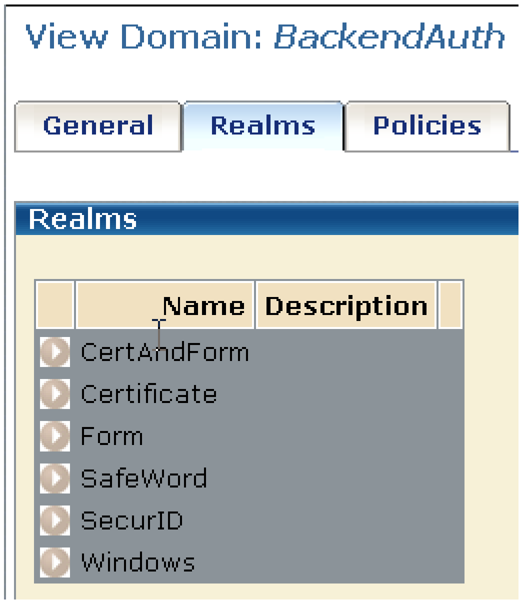
type

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.



Name	Description
CertAndForm	
Certificate	
Form	
SafeWord	
SecurID	
Windows	

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:

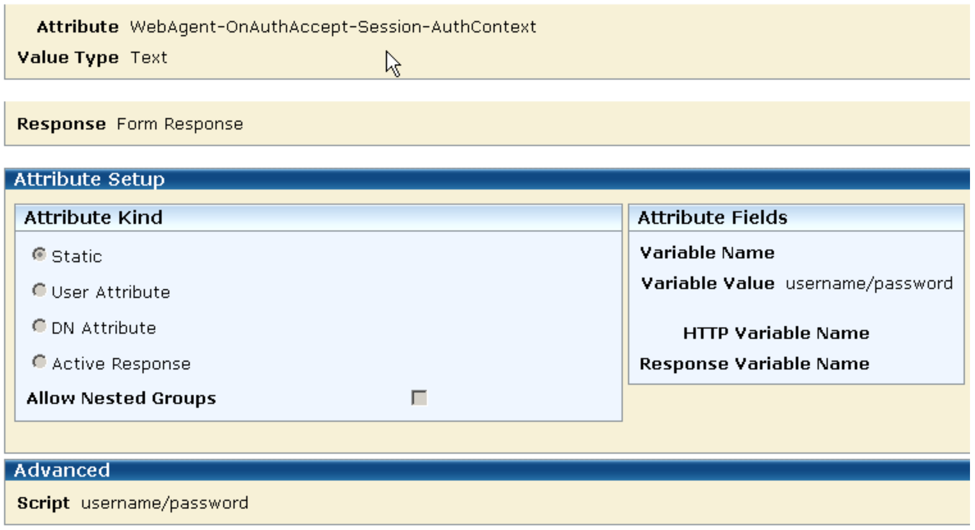
Name	Agent Type	Description
Form	Web Agent	AuthContext for username/password auth
Certificate	Web Agent	AuthContext for certificate auth

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.



Attribute WebAgent-OnAuthAccept-Session-AuthContext
Value Type Text

Response Form Response

Attribute Setup

Attribute Kind

(●) Static
( ) User Attribute
( ) DN Attribute
( ) Active Response
Allow Nested Groups ☐

Attribute Fields

Variable Name
Variable Value username/password
HTTP Variable Name
Response Variable Name

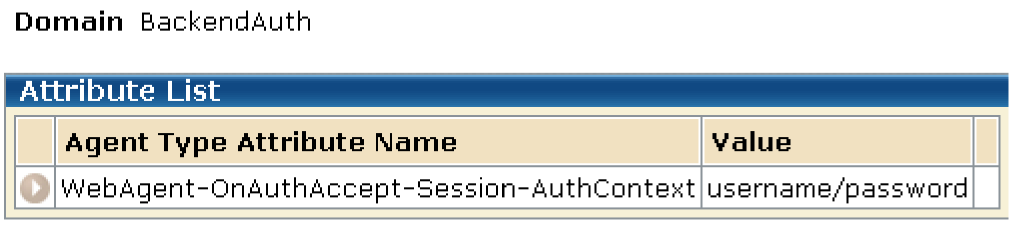
Advanced

Script username/password

As the illustration shows, the AuthContext response attribute type is static. When Federation Security Services 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

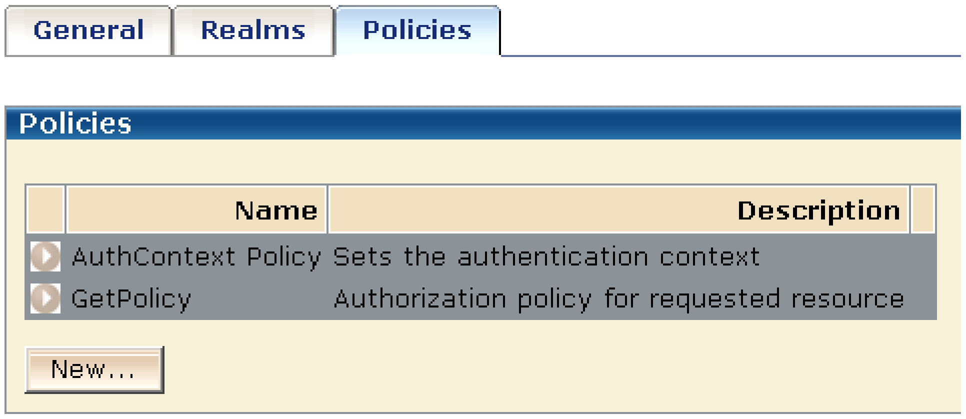
Agent Type Attribute Name	Value
WebAgent-OnAuthAccept-Session-AuthContext	username/password

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.



Policies

Name	Description
AuthContext Policy	Sets the authentication context
GetPolicy	Authorization policy for requested resource

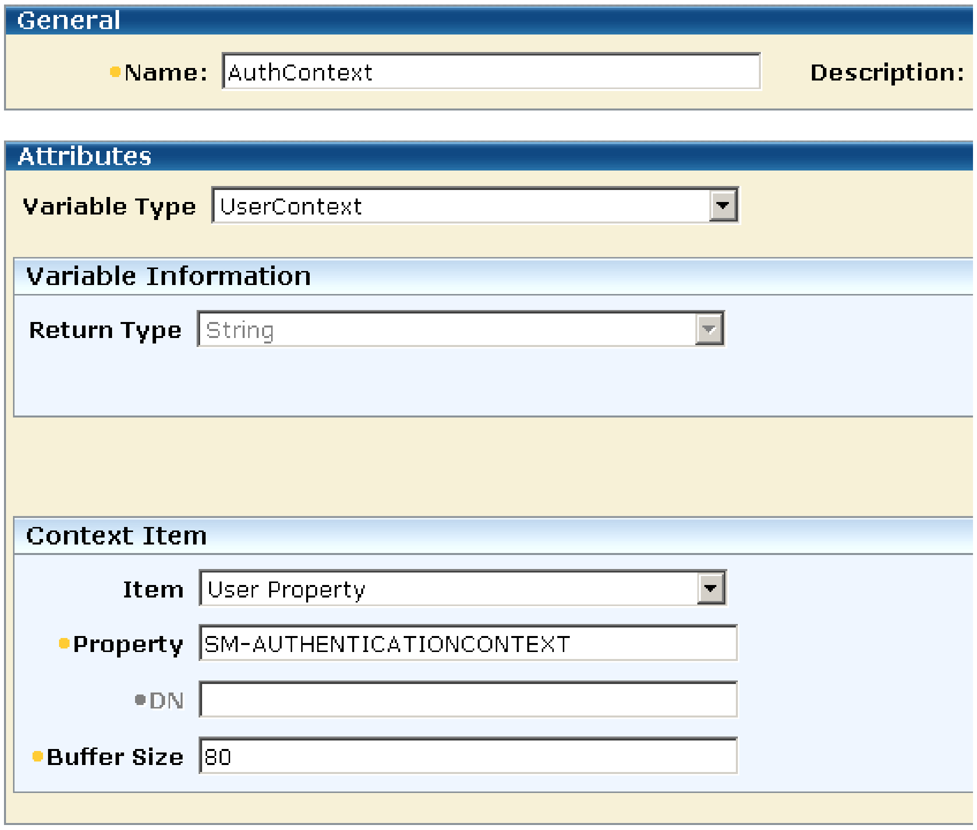
Create an Authentication Context Policy

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 Federation Security Services 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.


General

Name: AuthContext

Description:

Attributes

Variable Type: UserContext

Variable Information

Return Type: String

Context Item

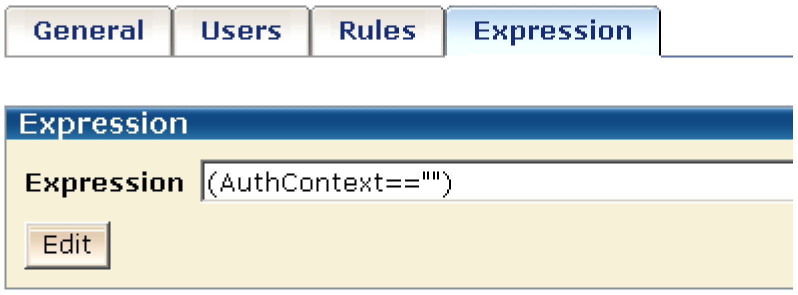
Item: User Property

Property: SM-AUTHENTICATIONCONTEXT

DN:

Buffer Size: 80

Define an active expression using the AuthContext variable in the AuthContext policy:



Expression (AuthContext=="")

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
```

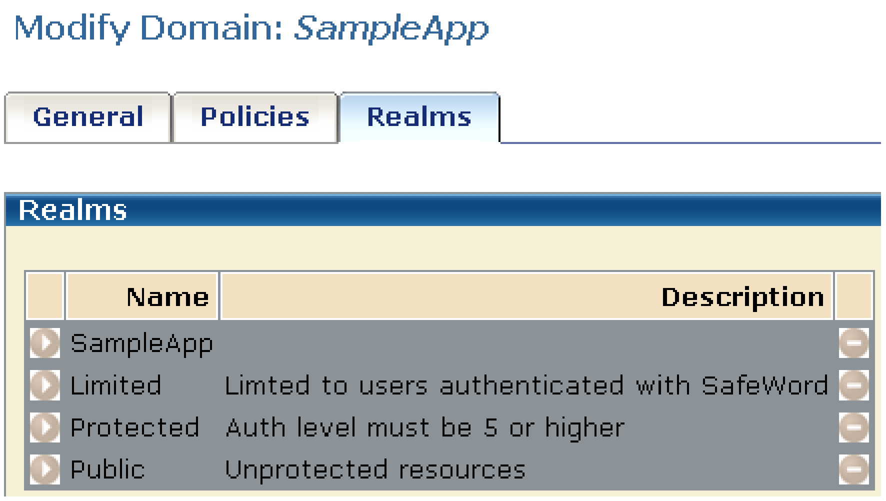
The sample application has four kinds or resources contained in different realms. The realms must each be configured with the front-end authentication scheme.

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.

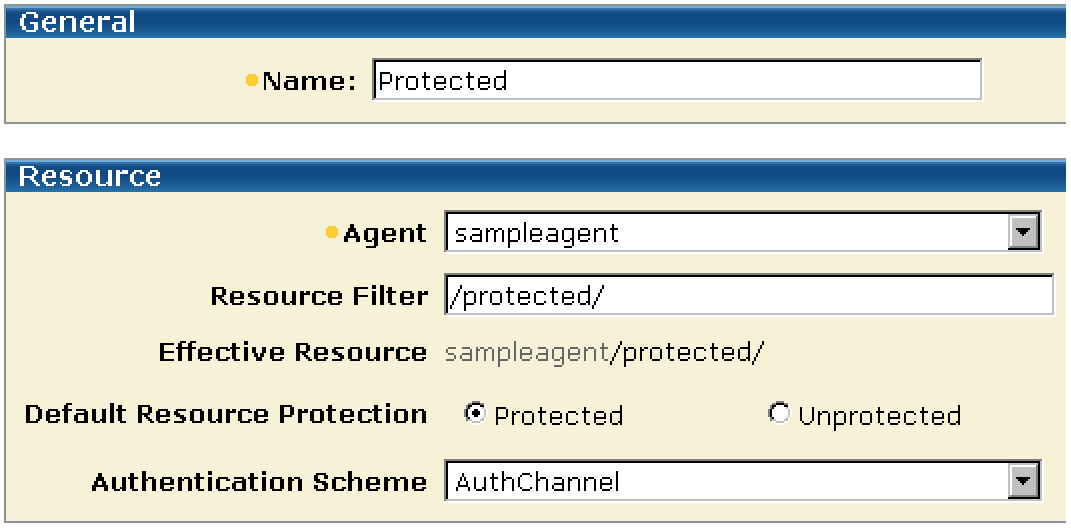


Modify Domain: *SampleApp*

Realms

	Name	Description	
	SampleApp		
	Limited	Limted to users authenticated with SafeWord	
	Protected	Auth level must be 5 or higher	
	Public	Unprotected resources	

The protected realms are configured with the AuthChannel front-end authentication scheme, as shown in the following illustration.



General

Name:	Protected

Resource

Agent	sampleagent
Resource Filter	/protected/
Effective Resource	sampleagent/protected/
Default Resource Protection	[✓ Protected] Unprotected
Authentication Scheme	AuthChannel

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

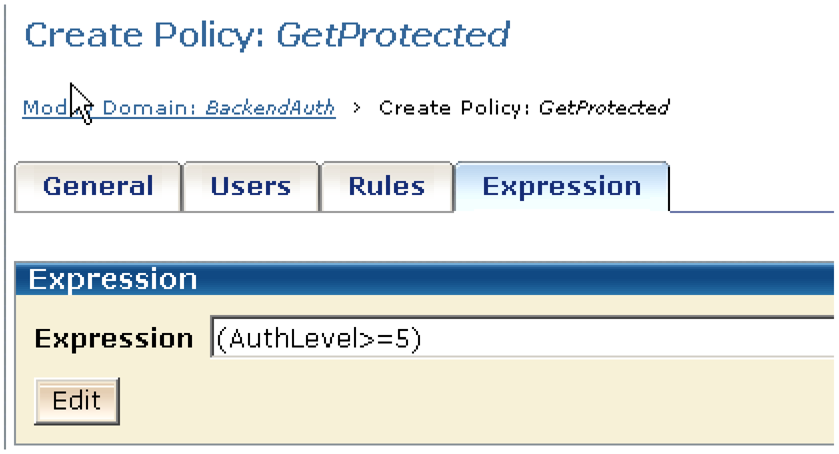
You can configure any type of rule for the realms that contain the sample application resources.

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 Federation Security Services 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.



Create Policy: *GetProtected*

Modify Domain: *BackendAuth* > Create Policy: *GetProtected*

Expression

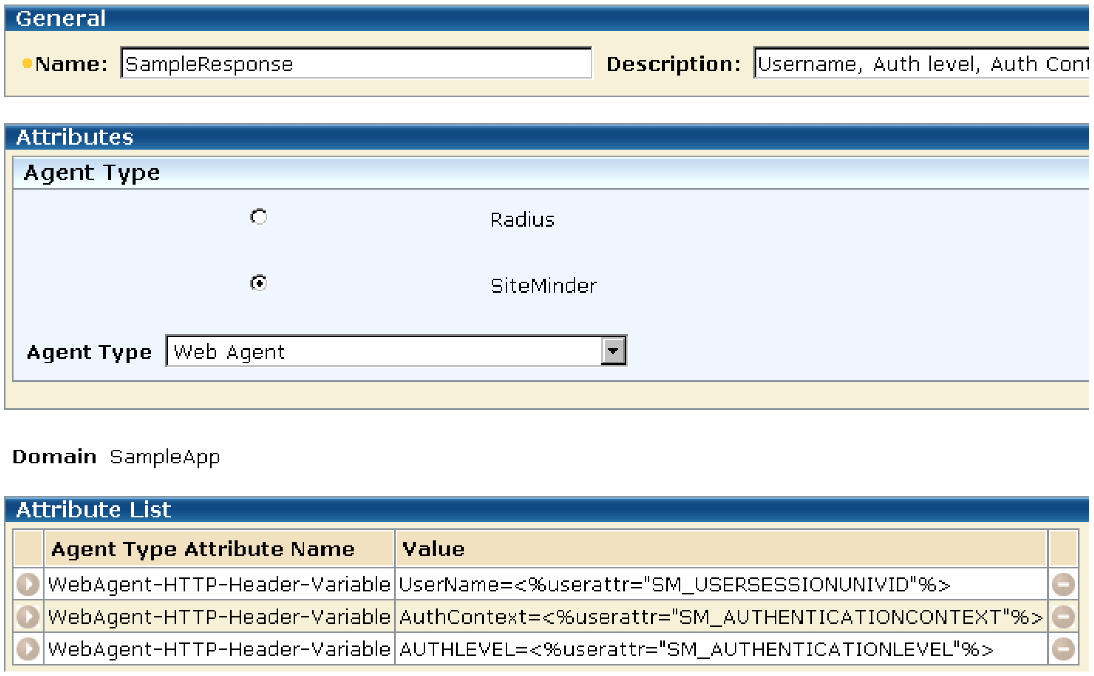
Expression (AuthLevel>=5)

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:



General

•Name: SampleResponse Description: Username, Auth level, Auth Cont

Attributes

Agent Type

Radius

(●) SiteMinder

Agent Type: Web Agent

Domain SampleApp

Attribute List

Agent Type Attribute Name	Value
WebAgent-HTTP-Header-Variable	UserName=<%userattr="SM_USERSESSIONUNIVID"%>
WebAgent-HTTP-Header-Variable	AuthContext=<%userattr="SM_AUTHENTICATIONCONTEXT"%>
WebAgent-HTTP-Header-Variable	AUTHLEVEL=<%userattr="SM_AUTHENTICATIONLEVEL"%>

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.

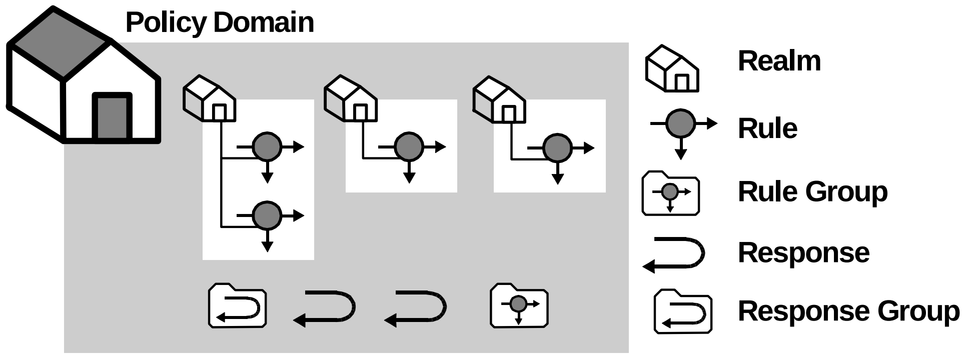
Chapter 11: Domains

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.

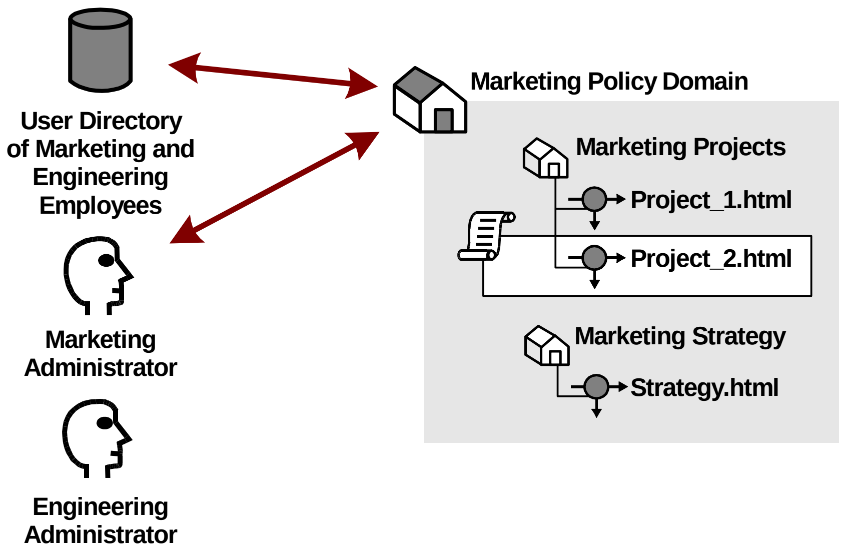
The resources in a policy domain can be grouped in one or more realms. A realm is a set of resources with a common security (authentication) requirement. Access to resources is controlled by rules, which are associated with the realm that contains the resource. The following diagram illustrates a small 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 439)

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 377)
Start the Administrative UI (see page 50)

Configure a Policy Domain

You can create a policy domain that protects logical groupings of resources.

To create a policy domain

1. Click Policies, Domains.
2. Click Domain, Create Domain.

   The Create Domain pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

3. Type the name and a description of the policy in the fields on the General group box.
4. Add User Directories and Realms.
5. Click Submit.

   The Create Domain Task is submitted for processing.

Assign User Directories

You can add one or more user directories to a policy domain. The Policy Server authenticates users by comparing the credentials that they enter to the credentials that are stored in the user directories. The Policy Server searches the user directories in the same order that they are listed in the policy domain.

To add user directories to a policy domain

1. Click Add/Remove on the User Directories group box on the Domain pane.

   The Choose user directories pane opens.

2. Select one or more user directories from the list of Available Members, and click the right-facing arrows.

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

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

3. Click OK.

   The selected user directories are added to the domain.

   Note: To create a new user directory and add it to the domain, click New... on the User Directories group box on the Domain pane.

Create a Realm

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 377)

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 SiteMinder Web Agent or Agent group. Realms group resources that have similar security requirements and share a common authentication scheme.

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

To create a realm in a domain and associate it with a SiteMinder Web Agent or Agent group

1. Click Policies, Domains, Realm, Create Realm.

   The Create Realm: Select Domain pane opens.
2. Select a domain, and click Next.

   The Create Realm: Define Realm opens.
3. Type the name and a description of the realm in the fields on the General group box.
4. Click the ellipsis button on the Resource group box.

   The Select an Agent group box opens.
5. Select a SiteMinder Web Agent or Agent group, and click OK.
6. Specify the remaining resource properties on the Resource group box.
7. Create new rules or delete existing rules on the Rules group box.
8. Create new sub-realms or delete existing sub-realms on the Sub-Realms group box.
9. Specify the session properties on the Session group box.
10. Specify the registration schemes, authorization directory mappings, and types of events the realm should process on the Advanced group box.
11. Click Finish.

    The Create Realm Task is submitted for processing.

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 allows you to 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 you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To create a realm in a domain and associate it with a RADIUS Agent or Agent group

1. Click Policies, Domains, Realm, Create Realm.

   The Create Realm: Select Domain pane opens.
2. Select a domain, and click Next.

   The Create Realm: Define Realm pane opens.
3. Type the name and a description of the realm in the fields on the General group box.
4. Click the ellipsis button on the Resource group box.

   The Select an Agent group box opens.
5. Select a RADIUS Agent or Agent group, and click OK.
6. Specify the remaining resource properties on the Resource group box.
7. Create new rules or delete existing rules on the Rules group box.
8. Specify the session properties on the Session group box.
9. Click Finish.

   The Create Realm Task is submitted for processing.

Configure a Realm with a 4.x Affiliate Agent

When you create a domain, you can create one or more realms in the domain and associate them with an Affiliate Agent or Agent group. Realms group resources that have similar security requirements and share a common authentication scheme.

Note: An Affiliate Agent must be created using the Federation Security Services Administrative User Interface. More information on Affiliate Agents exists in the *Federation Security Services Guide*.

Note: Do not confuse 4.x Affiliate Agents with SAML Affiliate Agents. SAML Affiliate Agents pass user information to a Web server so that Web applications can personalize the user's experience as part of Federation Security Services. 4.x Affiliate Agents protect resources.

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

To create a realm in a domain and associate it with an Affiliate Agent or Agent group

1. Click Policies, Domains, Realm, Create Realm.

   The Create Realm: Select Domain pane opens.

2. Select a domain, and click Next.

   The Create Realm: Define Realm opens.

3. Type the name and a description of the realm in the fields on the General group box.
4. Click the ellipsis button on the Resource group box.

   The Select an Agent group box opens.

5. Select an Affiliate Agent or Agent group, and click OK.

   Note: The remaining properties on the Resource group box are specified when the Affiliate Agent is configured on the Web server. See the *Web Agent Configuration Guide* for more information.

6. Create new rules or delete existing rules on the Rules group box.
7. Click Finish.

   The Create Realm Task is submitted for processing.

Disable Global Policy Processing for a Domain

Global policies let you associate responses with particular resources and events across all domains. By default, global policies apply to all of the resources in a policy domain.

To disable global policies for a specific domain

1. Open the domain.
2. Clear the Global Policies Apply check box, and then click Submit.

   Global policies no longer apply to the resources in this domain.

More information:

Global Policies, Rules, and Responses (see page 545)

Affiliate Domains

If you have purchased SiteMinder Federation Security Services, you can use the Federation Security Services Administrative User Interface to create Affiliate domains. An affiliate domain is a logical grouping of SAML 1.x Consumers or SAML 2.0 Service Providers associated with one or more user directories.

Note: More information exists in the *Federation Security Services Guide*.

Modify a Domain

You can change the name, description, user directory connections and administrators associated with a policy or affiliate domain. All other features of a domain are a result of peripheral configuration.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

Delete a Domain

Important! Deleting a domain destroys all of the domain user directory and administrator connections and objects: rules, rule groups, realms, responses, response groups, and policies, or affiliates contained in the domain.

It may take a short amount of time for all deleted objects to be removed from caches.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

Chapter 12: Realms

This section contains the following topics:

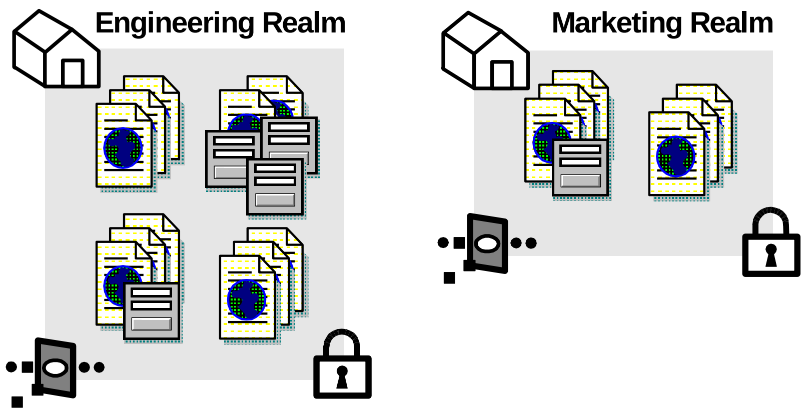
Realms Overview

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.

More information:

Advanced Policy Components for Applications (see page 507)

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/*

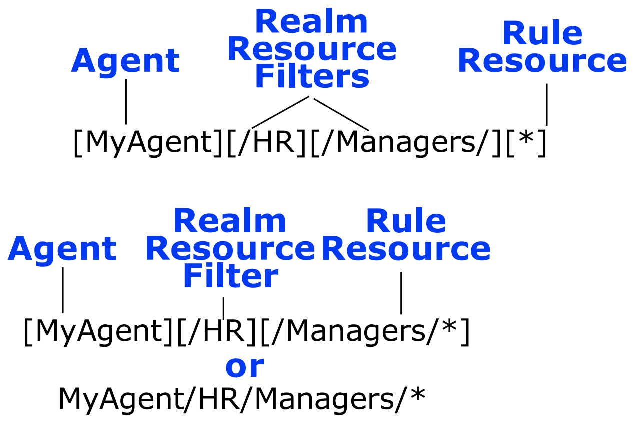
Combining the three elements, suppose that:

- 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 83)
Configure a Realm (see page 382)
Rules (see page 389)

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 323).

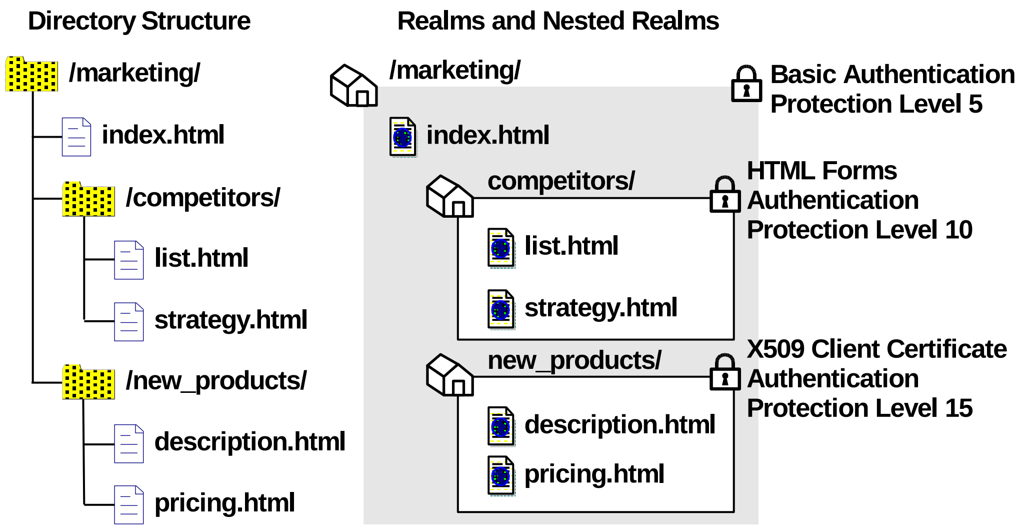
Nested Realms

In SiteMinder, 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 realm 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 389)
Policies (see page 439)

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 configure a realm to protect a group of resources that users access via a Web Server.

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

To configure the realm

1. Click Policies, Domains.
2. Click Realm, Create Realm.

   The Create Realm: Select Domain pane appears.
3. Select a domain from the Domain list, and click Next.

   The Create Realm: Define Realm pane appears.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Type the name and a description of the realm in the fields on the General group box.
5. Click the ellipsis button on the Resource group box.

   The Select an Agent group box opens.
6. Select a SiteMinder Web Agent or Agent group, and click OK.
7. Specify the remaining resource properties on the Resource group box.
8. Create new rules or delete existing rules on the Rules group box.
9. Create new sub-realms or delete existing sub-realms on the Sub-Realms group box.
10. Specify the session properties on the Session group box.
11. Specify the registration schemes, authorization directory mappings, and types of events the realm should process on the Advanced group box.
12. Click Finish.

    The Create Realm Task is submitted for processing.

Configure a Realm Protected by a RADIUS Agent

You configure a realm to protect a group of resources that users access via a Web Server.

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

To configure the realm

1. Click Policies, Domains.
2. Click Realm, Create Realm.

   The Create Realm: Select Domain pane appears.
3. Select a domain from the Domain list, and click Next.

   The Create Realm: Define Realm pane appears.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Type the name and a description of the realm in the fields on the General group box.
5. Click the ellipsis button on the Resource group box.

   The Select an Agent group box opens.
6. Select a RADIUS Agent or Agent group, and click OK.
7. Specify the remaining resource properties on the Resource group box.
8. Create new rules or delete existing rules on the Rules group box.
9. Specify the session properties on the Session group box.
10. Click Finish.

    The Create Realm Task is submitted for processing.

Configure a Realm for a 4.x Affiliate Agent

This topic refers to the legacy 4.x Affiliate Agent. The 4.x Affiliate Agent has no relationship with the affiliates, Service Providers, or Resource Partners that you can add to an affiliate domain.

Although the 4.x Affiliate Agent is an agent, it does not protect resources on an affiliate's Web site. However, to configure responses that pass information to a 4.x Affiliate Agent, you must configure a realm. Configuring a realm for a 4.x Affiliate Agent is simpler than configuring a SiteMinder Web Agent.

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

To configure the realm

1. Click Policies, Domains.
2. Click Realm, Create Realm.

   The Create Realm: Select Domain pane appears.
3. Select a domain from the Domain list, and click Next.

   The Create Realm: Define Realm pane appears.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Type the name and a description of the realm in the fields on the General group box.
5. Click the ellipsis button on the Resource group box.

   The Select an Agent group box opens.
6. Select an Affiliate Agent or Agent group, and click OK.

   Note: The remaining properties on the Resource group box are specified when the Affiliate Agent is configured on the Web server. See the *Web Agent Configuration Guide* for more information.
7. Create new rules or delete existing rules on the Rules group box.
8. Click Finish.

   The Create Realm Task is submitted for processing.

Modify a Realm

When you modify an existing realm you cannot make changes to the following:

- Agent

  The Agent that protects a server where an existing realm is located cannot be changed. If you need to change the Agent, you must delete the realm and recreate it with the new Agent.

- Resource Filter

  The resource filter of an existing realm cannot be changed. If you need to change the resource filter, you must delete the realm and recreate it with the new resource filter.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

Delete a Realm

When you delete a realm, all nested realms associated with the realm are also deleted. In addition, all rules associated with the deleted realm and its nested realms are also deleted.

It may take a short amount of time for all deleted objects to be removed from caches.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

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, Domains.
2. Click Realm, Modify Realm.

   The Modify Realm pane opens.

3. Specify search criteria, and click Search.

   A list of realms that match the search criteria opens.

4. Select a realm, and click Select.

   The Modify Realm: *Name* pane opens.

5. In the Sub-Realms group box, click Create Sub-Realm.

   The Create Realm pane opens.

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

   The Create Realm: *Name* pane opens.

7. Type the name and a description of the realm in the fields on the General group box.

8. Type the path of the resource filter in the Resource Filter field on the Resource group box.

   Note: The resource filter of the nested realm is added to the resource filter of the parent realm. For example, if the parent realm's filter is /marketing, and the nested realm's filter is /data, the entire filter is: <agent_of_the_parent_realm>/marketing/data.

   Note: Asterisk (*) and question mark (?) characters are treated as literal characters in resource filters, not wildcards.

9. Specify session properties on the Session group box.

10. In the Advanced group box, specify the following settings:

    - Authorization directory mapping
    - Events processing

11. Click Submit.

    The Create Realm task is submitted for processing.

More information:

Nested Realms (see page 380)
Start the Administrative UI (see page 50)

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 make a change to the security requirements or contents of a realm, you may want to flush the realm from the SiteMinder resource cache.

Note: If you have the Manage System and Domain Objects administrative privilege, you can flush all realms from the resource cache using the Cache Management dialog. More information exists in the *Policy Server Administration Guide*.

To flush a single realm from the resource cache

1. Login to the Administrative UI.
2. Click the Policies tab.
3. Click Domains, Realm, Modify Realm.

   The search window appears.
4. (Optional) Fill out the search form to narrow your search criteria.
5. Click Search.

   A list of realms appears.
6. Click the radio button on the left of the Realm that you want, and then click Select.

   The Modify Realm:*Realm Name* pane appears.
7. Click Flush in the Advanced group box.

   SiteMinder flushes the realm from the resource cache.

More information:

Start the Administrative UI (see page 50)

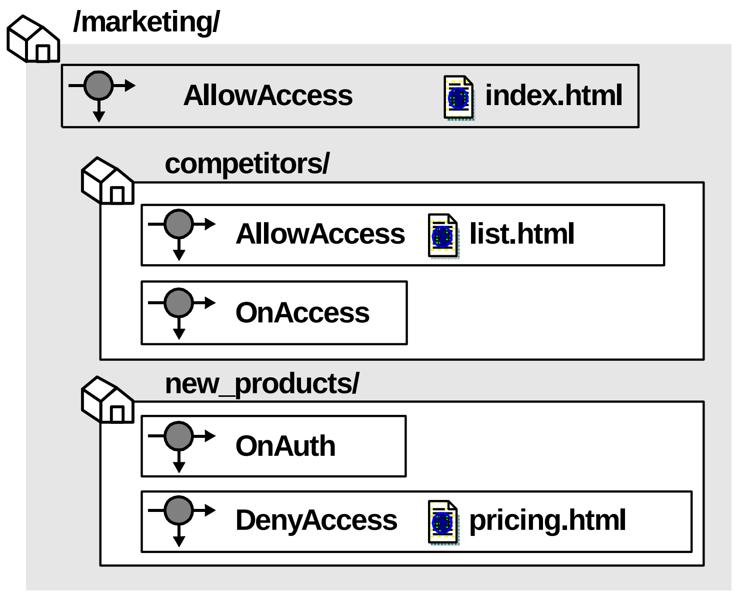
Chapter 13: Rules

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 403)
Advanced Policy Components for Applications (see page 507)

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 380)

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 439)

Web Agent Actions

Rules with a Web Agent action either allow or deny access to the resource(s) specified by a rule when one of the HTTP actions specified in the rule occur.

When a rule that specifies Allow Access fires, if a user authenticates successfully, SiteMinder allows the user to access the specified resource. If a rule specifies Deny Access, SiteMinder denies access to the successfully authenticated user. Deny access rules may be added to policies to provide an additional layer of security by rejecting specific individuals or groups who should not have access to a resource. Allow Access is the default.

Deny access rules take precedence over allow access rules. If a deny access rule and an allow access rule fire when a user attempts to access a resource, the presence of the deny access rule overrides all allow access rules.

The Web Agent rule actions are:

Get

Retrieves a resource for viewing via HTTP.

Put

Supports legacy HTTP actions.

Post

Posts information supplied by a user via HTTP.

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*.

More information:

Configure a Realm for a 4.x Affiliate Agent (see page 385)

Authentication Events

Authentication events occur as SiteMinder tries to establish a user's 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 can't create an On-Auth rule that applies to a portion of a realm.

The following is a list of possible On-Auth events:

OnAuthAccept

Occurs if authentication was successful. This event may be used to redirect a user after a successful authentication.

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 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). See section 3.3 of the *SiteMinder 4.x Web and Affiliate Agent Quarterly Maintenance Release 4 Release Notes* for more information on EnforceRealmTimeouts.

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).

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.

OnAuthAttempt

Occurs if the user was rejected because SiteMinder does not know this user (an unregistered user, for example, 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. This event should not be used 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 associated with the applicable On-Auth rule back to the requesting Agent.

Note: To optimize SiteMinder performance and limit the number of times the Web Agent must retrieve static information from the Policy Server, you can set up a rule based on the OnAuthAccept authentication event, then 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, and the static response is only returned to users who successfully authenticate.

More information:

Advanced Policy Components for Applications (see page 507)

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.

More information:

Advanced Policy Components for Applications (see page 507)

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 406)

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, Domains.
2. Click Rule, Create Rule.

   The Create Rule: Select Domain pane opens.
3. Select a domain from the Domain list, and click Next.

   The Create Rule: Select Realm pane opens.
4. Select the realm that includes the resources that you want the rule to protect, and click Next.

   The Create Rule: Define Rule pane opens.

   Note: If a realm does not exist for the resources that you want to protect, a rule cannot be created to protect those resources.
5. Type the name and a description of the rule in the fields on the General group box.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
6. Type the resource that you want the rule to protect in the Resource field.

   The Effective Resource updates to include the resource.
7. Specify whether the rules should allow or deny access to the protected resource in the Allow/Deny and Enable/Disable group box.
8. Select the Web Agent actions radio button on the Action group box.

   The Action List is populated with HTTP actions.
9. Select one or more HTTP actions from the Action list.
10. (Optional) Specify time restrictions, an active rule, or both on the Advanced group box.
11. Click Finish.

    The Create Rule task is submitted for processing.

More information:

Regular Expressions for Resource Matching (see page 403)
Advanced Rule Options (see page 396)

Configure a Rule for Authentication Event Actions

You 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. Ensure that Process Authentication Events is selected in the Advanced group box of the Realm pane.

To create a rule

1. Click Policies, Domains.
2. Click Rule, Create Rule.

   The Create Rule: Select Domain pane opens.
3. Select a domain from the Domain list, and click Next.

   The Create Rule: Select Realm pane opens.
4. Select the realm that includes the resources that you want the rule to protect, and click Next.

   The Create Rule: Define Rule pane opens.

   Note: If a realm does not exist for the resources that you want to protect, a rule cannot be created to protect those resources.
5. Type the name and a description of the rule in the fields on the General group box.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
6. Select the Authentication events radio button in the Action group box.

   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.
7. Select one or more authentication events from the Action List.

8. (Optional) Set time restrictions and or active rule settings in the Advanced group box.
9. Click Finish.

   The rule is saved and applied to the specified realm and resource.

More information:

Authentication Events (see page 393)
Configure a Realm (see page 382)
Advanced Rule Options (see page 396)

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. Ensure that the Process Authorization Events option is selected in the Advanced group box of the Realm pane.

To create a rule

1. Click Policies, Domains.
2. Click Rule, Create Rule.

   The Create Rule: Select Domain pane opens.
3. Select a domain from the Domain list, and click Next.

   The Create Rule: Select Realm pane opens.
4. Select the realm that includes the resources that you want the rule to protect, and click Next.

   The Create Rule: Define Rule pane opens.

   Note: If a realm does not exist for the resources that you want to protect, a rule cannot be created to protect those resources.
5. Type the name and a description of the rule in the fields on the General group box.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
6. Type the resource the rule is to protect in the Resource field.

   The Effective Resource updates to include the resource.

7. Select the Authorization events radio button in the Action group box.

   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.

8. Select one or more authorization events from the Action List.
9. (Optional) Set time restrictions and/or an active rule in the Advanced group box.
10. Click Submit.

    The rule is saved and applied to the specified realm and resource.

More information:

Authorization Events (see page 395)
Configure a Realm (see page 382)
Regular Expressions for Resource Matching (see page 403)
Advanced Rule Options (see page 396)

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 configure a rule for impersonation events to start impersonation sessions when resources are accessed.

To create a rule

1. Click Policies, Domains.
2. Click Rule, Create Rule.

   The Create Rule: Select Domain pane opens.
3. Select a domain from the Domain list, and click Next.

   The Create Rule: Select Realm pane opens.
4. Select the realm that includes the resources that you want the rule to protect, and click Next.

   The Create Rule: Define Rule pane opens.

   Note: If a realm does not exist for the resources that you want to protect, a rule cannot be created to protect those resources.
5. Type the name and a description of the rule in the fields on the General group box.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
6. Type the resource the rule is to protect in the Resource field.

   The Effective Resource updates to include the resource.
7. Select the Impersonation events radio button in the Action group box.

   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.
8. Select one or more impersonation events from the Action List.
9. (Optional) Set time restrictions and/or an active rule in the Advanced group box.
10. Click Finish.

    The rule is saved and applied to the specified realm and resource.

More information:

Impersonation (see page 567)
Regular Expressions for Resource Matching (see page 403)
Impersonation Realms and Events (see page 571)
Advanced Rule Options (see page 396)

Configure a Rule for an Affiliate Agent

4.x Affiliate Agents are SiteMinder Agents that communicate with SiteMinder Web Agents installed on the Web servers of a portal Web site. 4.x Affiliate Agent rules are very simple, since they do not protect the resources of an affiliate Web site. The 4.x 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.

The only action available in a rule for a 4.x Affiliate Agent is Visit. This is a special action for 4.x Affiliate Agents that let a SiteMinder portal site interact with an affiliate Web site.

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

To create a rule

1. Click Policies, Domains.
2. Click Rule, Create Rule.

   The Create Rule: Select Domain pane opens.
3. Select a domain from the Domain list, and click Next.

   The Create Rule: Select Realm pane opens.
4. Select the realm that includes the resources that you want the rule to protect, and click Next.

   The Create Rule: Define Rule pane opens.

   Note: If a realm does not exist for the resources that you want to protect, a rule cannot be created to protect those resources.
5. Type the name and a description of the rule in the fields on the General group box.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

6. Select the Affiliate Agent actions radio button on the Action group box.

   The Action List populates with Visit.

   Note: The remaining rule properties are disabled. The resources protected by an Affiliate Agent are specified when you configure the Affiliate Agent on the Web Server.

7. Click Finish.

   The rule is saved and applied to the specified realm and resource.

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
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.

More information:

Start the Administrative UI (see page 50)
Legacy Administrator Privileges (see page 63)

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 66).

To configure a time restriction

1. Click Set in the Time Restrictions group box.

   The Time Restrictions pane appears.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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 separately via the Software Development Kit

Note: More information on shared libraries exists in the *API Reference Guide for C*.

To configure an Active Rule

1. Select the Edit Active Rule check box in the Active Rule group box.

   Active rule fields appear.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

2. Specify the library name, function name, and function parameters in the respective fields.

   The active rule string appears in the Active Rule field.

3. Click Submit.

   The active rule is saved.

Delete a Rule

If you delete a rule, the rule is automatically removed from the policies that included the rule. However, the policies remain on your system. Verify that the policies function without the deleted rule.

Note: Policies must contain at least one rule.

When you delete a rule that is included in a rule group, it may take several seconds before the deleted rule is removed from the rule group. It may also take a short amount of time for all deleted objects to be removed from caches.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

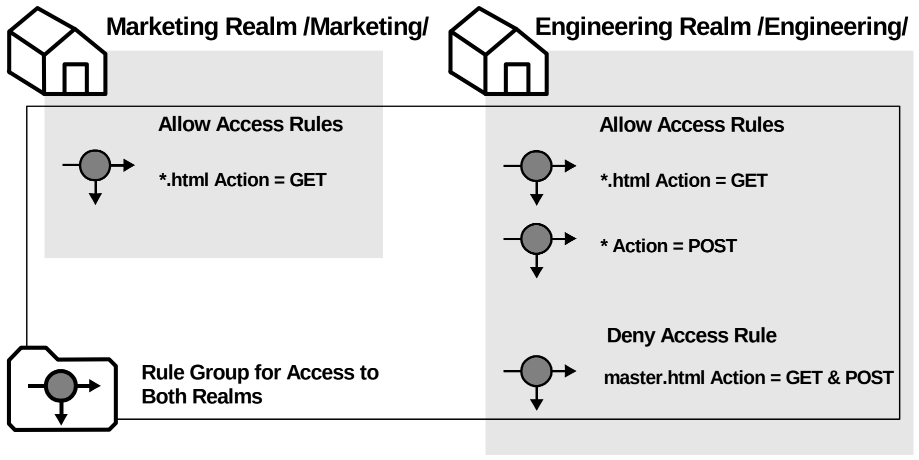
Chapter 14: Rule Groups

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.


Marketing Realm /Marketing/
Engineering Realm /Engineering/
Allow Access Rules
*.html Action = GET
Allow Access Rules
*.html Action = GET
* Action = POST
Deny Access Rule
master.html Action = GET & POST
Rule Group for Access to Both Realms

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 439)

Create a Rule Group

You can create a rule group and add it to a domain.

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

To create a rule group

1. Click Policies, Domains.
2. Click Rule Group, Create Rule Group.

   The Create Rule Group pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create Rule Group: Select Domain pane opens.
4. Select a domain name from the drop-down list, and click Next.

   The Create Rule Group: Define Rule Group pane opens.
5. Type the name and a description of the rule group in the fields on the General group box.
6. Select Radius or SiteMinder and an Agent Type on the Attributes group box.
7. Click Add/Remove on the Group Members group box.

   The Choose rules pane opens.

   The Available Members column lists all rules that are defined in the specified domain and in the realms associated with the specified Agent type. When the Agent type is Generic RADIUS, the Available Members column lists all rules that are supported by RADIUS Agents.
8. Select one or more rules from the list of Available Members and click the right-facing arrows.

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

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

   The selected rules are added to the rule group.
10. Click Finish.

    The Create Rule Group Task is submitted for processing.

More information:

Add Rules to a Rule Group (see page 411)

Add Rules to a Rule Group

You can add rules to a rule group in the same domain and of the same Agent type.

To add rules to a rule group

1. Click Policies, Domains.
2. Click Rule Group, Modify Rule Group.

   The Modify Rule Group pane opens.
3. Specify search criteria, and click Search.

   A list of rule groups opens.
4. Select a rule group, and click Select.

   The Modify Rule Group: Name pane opens.
5. Click Add/Remove on the Group Members group box.

   The Choose rules pane opens.

   Note: The Available Members column lists all rules that are defined in the specified domain and in the realms associated with the specified Agent type. When the Agent type is Generic RADIUS, the Available Members column lists all rules that are supported by RADIUS Agents.
6. Select one or more rules from the list of Available Members and click the right-facing arrows.

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

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

   The selected rules are added to the rule group.
8. Click Submit.

   The Modify Rule Group Task is submitted for processing.

Modify a Rule Group

You can modify all of the properties of a rule group, except the Agent Type for 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 on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

Delete a Rule Group

Deleting a rule group only deletes the grouping. The rules contained in the grouping are not deleted.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

Chapter 15: 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.

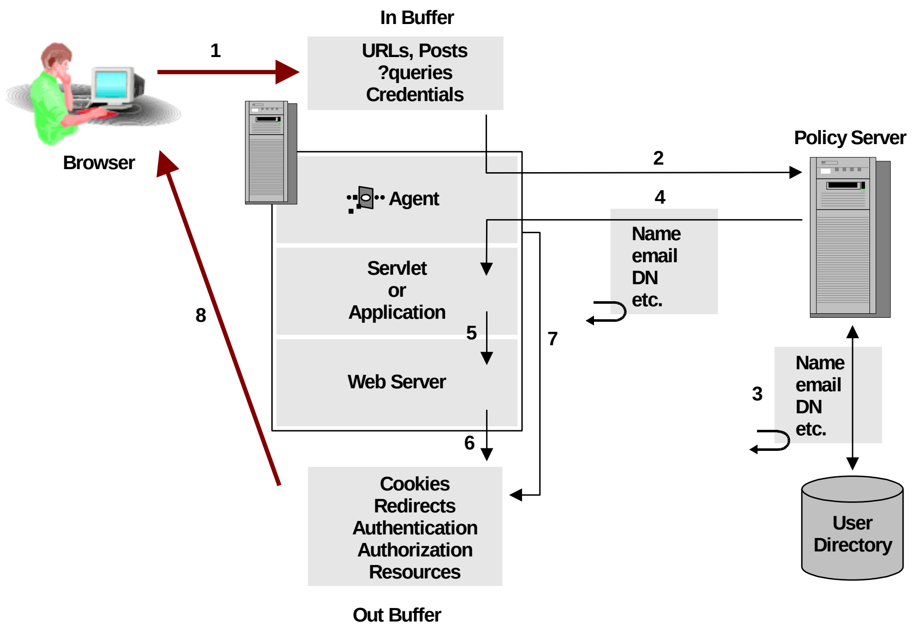
Responses take the form of name/value pairs. When a rule is triggered, the Policy Server returns the paired response to the SiteMinder Agent.

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.

More information:

Web Agent Response Attributes (see page 416)

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 418)

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 419)

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 is a list of generally available Web Agent response attributes:

WebAgent-HTTP-Authorization-Variable

Indicates an attribute defined and reserved for future SiteMinder use.

WebAgent-HTTP-Cookie-Variable

Generates a SetCookie header, which then sets a non-persistent cookie in a Web browser. The cookies only exist in the cookie domain where the Web Agent is configured. You can enter multiple WebAgent-HTTP-Cookie-Variables.

Limits: Use in accept or reject responses. Multiple instances of this attribute are allowed per response.

WebAgent-HTTP-Header Variable

Specifies an arbitrary dynamic name/value pair for use by a Web application. You can enter multiple WebAgent-HTTP-Header-Variables.

The Web Agent does not include header variables in the responses that it sends back to a Web browser. Instead, these responses, generated by the Policy Server, reside in the request headers of the Web server.

Consequently, the header variables will not be visible in the debug logs that you can enable from the Policy Server Management Console.

Limits: Use in accept or reject responses. Multiple instances of this attribute are allowed per response.

WebAgent-OnAccept-Redirect

Defines *one* of the following, depending on the type of response in which it is used:

- In an authorization response, this defines a URL to redirect the user to if the user is allowed access to a resource.
- In an authentication response, this defines a URL to redirect the user to if the user was authenticated for a security realm.

To determine whether or not this is an authorization or authentication response, include it in a policy with a rule that specifies an OnAuthAccept or OnAccessAccept event action.

Limits: Use in accept responses. Only one instance of this attribute is allowed per response.

WebAgent-OnAccept-Text

Specifies text that the Web Agent puts in the HTTP_ONACCEPT_TEXT environment variable when it redirects the user after a successful authorization or authentication attempt.

Limits: Use in accept responses. Only one instance of this attribute is allowed per response.

Note: When configuring a Web Agent OnAcceptText response, set the FCC Compatibility Mode parameter (fcccompatmode) corresponding to the Web Agent to yes. This ensures that user authentication takes place at the Web Agent and that the text in the response is available for display in the user's browser. If the FCC Compatibility Mode parameter (fcccompatmode) is set to no, user authentication takes place at the Forms Credential Collector (FCC), where 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. Once this limit is reached, the user is forced to re-authenticate. Associate this response with a rule configured with an OnAuthAccept authentication event.

Limits: Use in accept responses. Only one instance of this attribute is allowed per response.

WebAgent-OnAuthAccept-Session-Max-Timeout

Overrides the total number of seconds a user session can be active. Once this limit is reached, the user session is terminated and the user is forced to re-authenticate. Associate this response with a rule configured with an OnAuthAccept authentication event.

Limits: Use in accept responses. Only one instance of this attribute is allowed per response.

WebAgent-OnAuthAccept-Session-AuthContext

Specifies an AuthContext response attribute for an authentication scheme. The value of this response attribute is added to the SiteMinder session ticket as the value of the SM_AUTHENTICATIONCONTEXT user attribute. It is not returned to the client as a user response.

Note: The response attribute value is truncated to 80 bytes in length.

Limits: Used in accept responses. Only one instance of this attribute is allowed per response.

WebAgent-OnReject-Redirect

Defines *one* of the following, depending on the type of response in which it is used:

- In an authorization response, this defines a URL to redirect the user to if the user is denied access to a resource.
- In an authentication response, this defines a URL to redirect the user to if the user has failed to authenticate for a security realm.

To determine whether or not this is an authorization or authentication response, include it in a policy with a rule that specifies an OnAuthReject or OnAccessReject event action.

Limits: Use in reject responses. Only one instance of this attribute is allowed per response.

WebAgent-OnReject-Text

Specifies text that the Web Agent puts in the HTTP_ONREJECT_TEXT environment variable when it redirects the user after a failed authorization or authentication attempt.

Limits: Use in reject responses. Only one instance of this attribute is allowed per response.

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 for a realm. When you define a separate authorization directory, a user is authenticated based on the information contained in one directory, but authorized based on the information contained in another directory.

When you create a response and associate it with a authentication (OnAuth) event, any information retrieved from a user directory is retrieved from the authentication directory. If you create an authorization (OnAccess) event, any information retrieved from a user directory is retrieved from the authorization directory.

More information:

Directory Mapping Overview (see page 237)

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, Domains.
2. Click Response, Create Response.

   The Create Response: Select Domain pane opens.
3. Select a domain, and click Next.

   The Create Response: Define Response pane opens.

4. Type the name and a description of the response in the fields on the General box.
5. Select Radius or SiteMinder and an Agent Type on the Attributes group box.
6. (Optional) Click Create Response Attribute to create a response attribute and add it to the attribute list.

   The Create Response Attribute pane opens.
7. Click Finish.

   The Create Response Task is submitted for processing.

More information:

Configure a Web Agent Response Attribute (see page 422)
Configure an Affiliate Agent Response Attribute (see page 424)
Configure a RADIUS Response Attribute (see page 423)

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 (see page 669).

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.

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's entry in a user directory.

This type of response attribute returns information associated with a user in a directory. A user attribute can be retrieved from an LDAP, WinNT, Microsoft SQL Server or Oracle user directory.

Note: In order for the Policy Server to return values from user directory attributes as response attributes, the user directories must be configured on the SiteMinder User Directory pane.

DN Attribute

Returns profile information from a directory object in an LDAP, Microsoft SQL Server or Oracle user directory.

This type of response attribute is used to return information associated with directory objects to which the user is related. Groups to which a user belongs, and Organizational Units (OUs) that are part of a user DN, are examples of directory objects whose attributes can be treated as DN attributes.

For example, you can use a DN attribute to return a company division for a user, based on the user's membership in a division.

Note: In order for the Policy Server to return values from DN attributes as response attributes, the user directories must be configured on the 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 must conform to the interface specified by the Authorization API (available separately with the Software Development Kit; if installed, see the API Reference *Guide for C* for more information).

Note: It is up to you to make sure the value returned by an active response is valid. For example, if an active response returns a numeric type, the library and function must return a string whose value is a number.
When you configure a response attribute, the correct Value Type for the response attribute is displayed on the Response Attribute pane.

Variable Definition

Returns the value of the specified variable at runtime.

Select Variable Definition when you want to select and use a variable from a list of already-defined variables.

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 on the Attribute List group box on the Response pane.

   The Create Response Attribute pane opens.

2. Select a response attribute from the drop-down list.

   Note: Complete descriptions of response attributes exist in the *Web Agent Configuration Guide*.

3. Select an attribute type on the Attribute Kind group box.

   The fields on the Attribute Fields group box are updated to match the specified attribute type.

4. Complete the fields on the Attribute Fields group box.

   Note: A list of automatically generated SiteMinder user attributes that you can use in responses exists in SiteMinder Generated User Attributes (see page 429).

5. Specify Cache Value or Recalculate value every ... seconds on the Attribute Caching group box.

6. Click Submit.

   The Create Response Attribute Task is submitted for processing, and the response attribute is added to the Attribute List on the Response pane.

Configure 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 on the Attribute List group box on the Response pane.

   The Create Response Attribute pane opens.

2. Select a response attribute from the drop-down list.

   Note: Complete descriptions of response attributes exist in the *Web Agent Configuration Guide*.

3. Select an attribute type on the Attribute Kind group box.

   The fields on the Attribute Fields group box are updated to match the specified attribute type.

4. Complete the fields on the Attribute Fields group box.

   Note: A list of automatically generated SiteMinder user attributes that you can use in responses exists in SiteMinder Generated User Attributes (see page 429).

5. Specify Cache Value or Recalculate value every ... seconds on the Attribute Caching group box.

6. Click Submit.

   The Create Response Attribute Task is submitted for processing, and the response attribute is added to the Attribute List on the Response pane.

More information:

Configure a Web Agent Response Attribute (see page 422)

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

To create a response attribute

1. Click Create Response Attribute on the Attribute List group box on the Response pane.

   The Create Response Attribute pane opens.

2. Select a response attribute from the drop-down list.

   Note: Complete descriptions of response attributes exist in the *Web Agent Configuration Guide*.

3. Select an attribute type on the Attribute Kind group box.

   The fields on the Attribute Fields group box are updated to match the specified attribute type.

4. Complete the fields on the Attribute Fields group box.

   Note: A list of automatically generated SiteMinder user attributes that you can use in responses exists in SiteMinder Generated User Attributes (see page 429).

5. Specify Cache Value or Recalculate value every ... seconds on the Attribute Caching group box.

6. Click Submit.

   The Create Response Attribute Task is submitted for processing, and the response attribute is added to the Attribute List on the Response pane.

More information:

Configure a Web Agent Response Attribute (see page 422)

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 (see page 419) to create a response.
2. Select SiteMinder and Web Agent as the Agent Type on the Attributes group box.
3. Click Create Response Attribute on the Attribute List group box.

   The Create Response Attribute pane opens.
4. Select a response attribute from the drop-down list on the Attribute Type group box.
5. Select the type of response attribute on the Attribute Kind group box.
6. Type the name of the variable object in the Variable Name field on the Attribute Fields group box.

   Note: When this field is required, 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 box:

   Static

   Specify the value of the static variable in the Variable Value field.

   User Attribute

   Specify the name of the user attribute in the Attribute Name field.

   DN Attribute

   Specify the DN of the user or user group in the DN Spec field and the name of the user attribute in the Attribute Name field.

   (Optional) Click Lookup to search for and select one set of users or user group in a specified user directory.

   (Optional) Select the Allow Nested Groups checkbox.

   Active Response

   Specify the name of your library, the name of a library function, and optionally the names of parameters in the Library Name, Function Name, and Parameters fields.

   Note: Your library must be based on the SiteMinder Authorization API.

   Variable Definition

   Click Lookup to select an existing variable object for the Variable field.

   Note: SiteMinder uses the information that you provide in the fields on the Attribute Fields group box to determine the value that it passes to the Web Agent in the form of a name-value pair.

8. Click OK.

   The response attribute is saved.

More information:

Response Attributes (see page 416)
Select a Variable Using Variable Lookup (see page 428)

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 an 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: You can click Help for a description of fields, controls, and their respective requirements.
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 on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

Delete a Response

Deleting a response removes the response from any policies with which it is associated.

It may take a short amount of time for all deleted objects to be removed from caches.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

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.

%SM_USER

The Web Agent places the username in an SM_USER http header variable for all requests. In cases where the user does not provide a username, such as certificate-based authentication, or in the case where the username is not known, the value of the SM_USER header variable is not set.

%SM_USERDN

For an authenticated user, the Web Agent populates this http header variable with the DN as determined by the Policy Server. In the case of certificate-based authentication, this attribute can be used to identify a user.

%SM_USERNAME

For an authenticated user, this attribute holds the user DN as disambiguated by SiteMinder. For an unauthenticated user, this attribute holds the user ID as specified by the user in the login attempt.

%SM_USERIMPERSONATORNAME

If the authentication scheme performs impersonation, this attribute holds the user DN that is authenticated by SiteMinder.

%SM_USERLOGINNAME

This attribute holds the user ID as specified by the user in the login attempt.

%SM_USERIPADDRESS

This attribute holds the user's IP address 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 disambiguated by SiteMinder). 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 as specified by the user 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 is generated by the agent.

%SM_USERSESSIONSPEC

The user's session ticket.

%SM_USERSESSIONID

This attribute holds the session ID of a user who has already been authenticated, or the session ID that will be assigned 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 user's universal ID. If no universal ID directory attribute is specified in the user directory definition, the value defaults to the user's DN.

%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 user’s session type. The value is one of the following:

- 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. For this attribute to be populated, both of the following conditions must be 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 prior to the last (which is represented by SM_USERLASTLOGINTIME. This response attribute is only available for an OnAuthAccept authentication event. For this attribute to be populated Password Services must be 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.

For example, if user JSmith belongs to the group Accounts Payable, which is contained in group Accounting, SM_OUSERNESTEDGROUPS[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_OUSERNESTEDGROUPS[.

For example, if user JSmith belongs to the group Accounts Payable, which is contained in group Accounting, SM_USERNESTEDGROUPS contains Accounting and Accounts Payable. If you want only Accounting, use SM_OUSERNESTEDGROUPS[.

%SM_USERSCHEMAATTRIBUTES

This attribute holds the user attributes associated with the DN, or properties 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 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, this attribute holds the names of the policies that give the user authorization. For example, suppose that to purchase an item, you must be one of the users associated with the Buyer policy. If the Policy Server authorizes me to buy an item, then SM_USERPOLICIES will contain Buyer.

%SM_USERPRIVS

When a user is authenticated or 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 is authorized for a resource under a realm, SM_USERREALMPRIVS holds all the response attributes for all rules under that realm.

For example, suppose that there is a realm called Equipment Purchasing. Under that realm, there is a CheckCredit rule. Associated with the CheckCredit rule is a response that returns the buyer's credit limit, such as limit = $15000, as a response attribute. If the buyer attempts to purchase equipment worth $5000, the CheckCredit 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.

For more information on Sm_Api_DisabledReason_t, see the *API Reference Guide for C* (available only if the Software Development Kit is installed).

More information:

Configure a Web Agent Response Attribute (see page 422)

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_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_OUSERNESTEDGROUPS[	Yes	Yes	No	Yes	Yes	No
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 you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To configure a response group

1. Click Policies, Domains.
2. Click Response Group, Create Response Group.

   The Create Response Group pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create Response Group: Select Domain pane opens.
4. Select a domain name from the drop-down list, and click Next.

   The Create Response Group: Define Response Group pane opens.
5. Type the name and a description of the response group in the fields on the General group box.
6. Select Radius or SiteMinder and an Agent Type on the Attributes group box.

   Note: The specified Agent type must correspond to the Agent type of the responses in the group. Only responses with the specified Agent type are available for inclusion in the group.

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

   The Choose responses pane opens.

   Note: The Available Members column lists all responses that are defined in the specified domain for the specified Agent type. When the Agent type is Generic Radius, the Available Members column lists all responses that are supported by Radius agents.

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

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

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

9. Click OK.

   The selected responses are added to the response group.

10. Click Submit.

    The Create Response Group Task is submitted for processing.

More information:

Start the Administrative UI (see page 50)

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 on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

Delete a Response Group

Deleting a response group only deletes the grouping, not the individual responses contained in the group.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

Chapter 16: Policies

This section contains the following topics:

Policy Overview

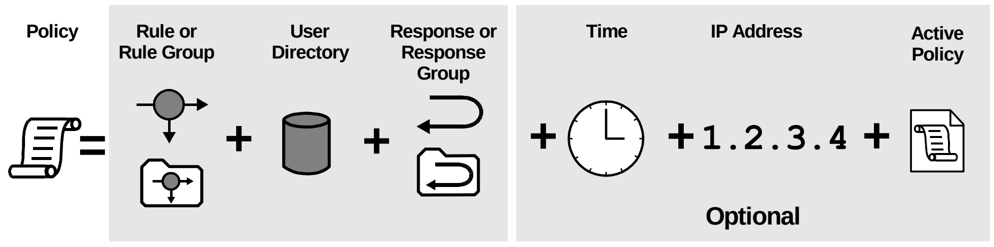
Policies define how users interact with resources. When you create policies in the Administrative UI, you link together (bind) objects that identify users, resources, and actions associated with the resources.

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 the customize the appearance of a resource. Responses can also trigger actions based on authentication and authorization events.

(Optional) IP Addresses

A policy may be limited to specific user IP addresses. Once you add an IP address restriction to a policy, if a user attempts to access a resource from an IP address not specified in the policy, the policy will not fire for the user, and therefore will not allow/deny access or process any responses.

(Optional) Time Restrictions

A policy may be limited to specific days or ranges of hours. A policy with a time restriction will not fire outside specified times, and therefore will not allow/deny access to protected resources or process any responses.

(Optional) Active Policies

An Active policy allows business logic external to 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 369)
Rules (see page 389)
Responses and Response Groups (see page 413)
User Directories (see page 119)
Allowable IP Addresses for Policies (see page 456)
Time Restrictions for Policies (see page 459)
Configure an Active Policy (see page 460)

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 247)
Policy Binding Establishment (see page 461)

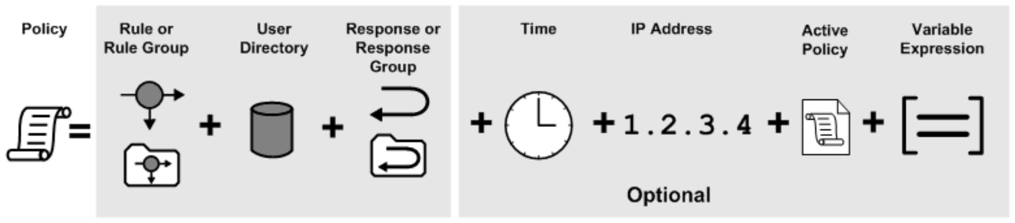
Expressions in Policies

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 454).

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 509)

How to Configure a Policy

The following process lists the steps for configuring a policy.

Note: You can also create policies using the Scripting Interface for Perl. For more information, see the *Programming Guide for Perl*.

To configure a policy

1. Create the policy (see page 443).
2. Add users to the policy (see page 444).
3. Add one or more rules to the policy (see page 445).
4. (Optional) Associate responses or response groups with rules (see page 445).
5. (Optional) Associate global responses with rules (see page 446).
6. (Optional). Configure advanced policy options (see page 456).

More information:

Start the Administrative UI (see page 50)
Add Users to a Policy (see page 444)
Add Rules to a Policy (see page 445)
Associate a Rule with a Response or Response Group (see page 445)
Associate a Rule with a Global Response (see page 446)
Advanced Policy Options (see page 456)

Create the Policy

You can create a policy by adding it to a new or existing domain. Policies define relationships between users and resources.

To create a policy and add it to an existing domain

1. Click the Policies, Domains.
2. Click Domain, Modify Domain.

   The Modify Domain pane opens.

3. Specify search criteria, and click Search.

   A list of domains that match the search criteria opens.

4. Select a domain, and click Select.

   The Modify Domain: *Name* pane opens.

5. Click the Policies tab on the Domain pane.

   The Policies group box opens.

6. Click Create.

   The Create Policy pane opens.

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

   The Create Policy: *Name* pane opens.

8. Type the name and a description of the policy in the fields on the General group box.

9. Click the Users tab.

   The User Directories group box opens.

10. Add users, user groups, or both to the policy, and click Submit.

    The Modify Domain: *Name* pane reopens.

11. Click Submit.

    The Modify Domain Task is submitted for processing.

Add Users to a Policy

You can add individual users, user groups, or both to a policy and create a policy binding between the added users and the policy. When a user tries to access a protected resource, the policy verifies that the user is part of its policy binding and then fires the rules included in the policy to see if the user is allowed to access the resource.

To add users to a policy

1. Click the Users tab on the Policy pane.

   The User Directories pane opens and contains group boxes for each user directory associated with the policy domain.

2. Add users or groups from the user directory to the policy.

   From within each user directory group box, you can choose Add Members, Add Entry, Add All. Depending on which method you use to add users to the policy, a dialog box will open enabling you to add users.

   Note: If you select Add Members, the User/Groups pane opens. Individual users are not displayed automatically. Use the search utility to find a specific user within one of the directories.

   You can edit or delete a user or group by clicking the right arrow (>) or minus sign (-), respectively.

3. Select individual users, user groups, or both using whatever method and click OK.

   The User Directories pane reopens and lists the policy's new users on the user directory's group box.

The task of binding users to the policy is complete.

More information:

View User Directory Contents (see page 196)
Policy Binding Establishment (see page 461)

Add Rules to a Policy

Rules indicate the specific resources included in a policy and whether to allow or deny access to the resources when the rule fires. Responses indicate the actions that should take place when the rule fires.

Note: You must add at least one rule or rule group to a policy.

To add rules or rule groups to a policy

1. Click the Rules tab on the Policy pane.

   The Rules group box opens.

2. Click Add Rule.

   The Available Rules pane opens.

3. Select the individual rules, rule groups, or both that you want to add to the policy, and click OK.

   The Rules group box lists the added rules and groups.

4. (Optional) Associate the rule with a response or response group.

   Note: To remove a rule or rule group from a policy, click the minus sign (-) to the right of the rule on the Rules group box. To create a new rule, click New Rule on the Available Rules pane.

Associate a Rule with a Response or Response Group

You can associate a response or response group with a rule in a policy. When the rule fires, the associated response also fires.

To associate a rule with a response or response group

1. Click Add Response for the rule or rule group for which you want to associate a response.

   The Available Responses pane opens and lists the responses and response groups that have been configured for the policy domain.

2. Select a response or response group, and click OK.

   The response opens in the Rules group box, and is associated with the respective rule.

   Note: If the response you require does not exist, click New Response to create the response.

Associate a Rule with a Global Response

You can associate a rule with an existing global response.

To associate a rule with a global response

1. Click the Rules tab on the Policy pane.

   The Rules group box opens.

2. Click the Add Response button next to the rule that you want to modify.

   The Available Responses pane opens.

   Note: Global responses, responses, and group responses are listed in that order on the Available Responses pane.

3. Select a global response, and click OK.

   The Rules group box reopens, and the selected response is added to the rule.

4. Click Submit.

   The Modify Policy Task is submitted for processing.

More information:

Global Policies, Rules, and Responses (see page 545)

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.

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 users and groups to be included 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 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. In the Policy Users/Groups Dialog, do one of the following:

   - For LDAP directories, select *one* of the following search types from the Where to Search drop-down list:

     Validate DN

     Locates the DN in the directory.

     Search Users

     Limits search to matches in user entries.

     Search Groups

     Limits search to matches in group entries.

     Search Organizations

     Limits search to matches in organization entries.

Search Any Entry

Limits searches to matches in user, group, and organization entries.WinNT directories

- For Microsoft SQL Server, Oracle, and WinNT directories, type a user name in the Manual Entry field.

  Note: For Microsoft SQL Server and Oracle, you can type a SQL query in the Manual Entry field instead of a user name.

  Example: SELECT NAME FROM EMPLOYEE WHERE JOB ='MGR';

The Policy Server will perform the query as the database user specified in the Username field of the Credentials and Connection tab for the user directory. When constructing the SQL statement for the Manual Entry field, you need to be 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: You can click Help for a description of fields, controls, and their respective requirements.

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 455)

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 66)
Add Time Restrictions to Rules (see page 406)

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: You can click Help for a description of fields, controls, and their respective requirements.
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).

User Namespace	Description
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 444)

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 452)
Add Users to a Policy (see page 444)

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 452)
Add Users to a Policy (see page 444)
Search User Directories (see page 196)

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 454)

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 454)

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 454)

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 444)

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 452)
Add Users to a Policy (see page 444)

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 452)
Add Users to a Policy (see page 444)
Search User Directories (see page 196)

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 444)

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 444)

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 452)
Add Users to a Policy (see page 444)

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 452)
Add Users to a Policy (see page 444)
Search User Directories (see page 196)

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 444)

Bind Policies to User Attributes

To bind policies to user attributes, specify a search expression that defines conditions related to user attributes that must be true.

For example, to bind a policy to all people whose area code is 555, insert the following expression in the Manual Entry field: (areacode=’555’).

Delete a Policy

When you are deleting a policy, remember that all of the elements in the policy still exist, it is only the grouping (or binding) of those elements that you remove when you delete the policy.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

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 443)

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:

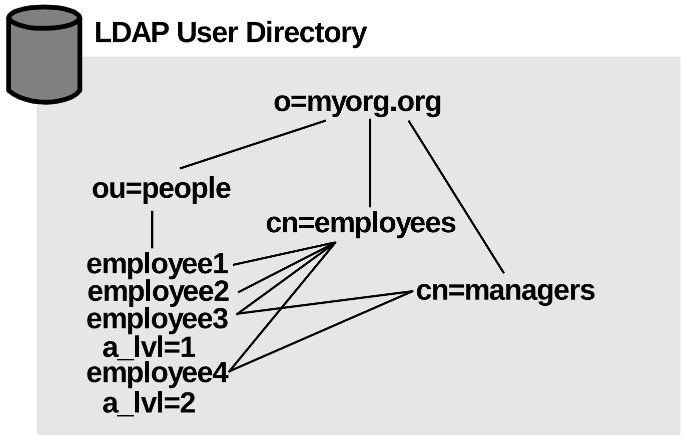
Nested Realms (see page 380)

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

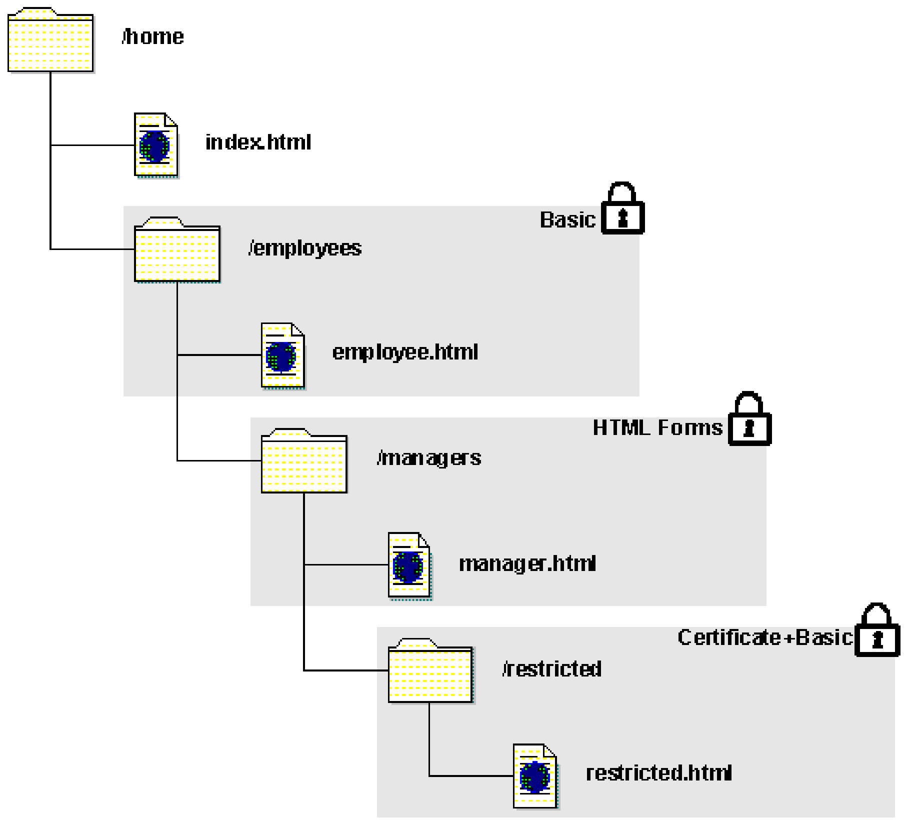
This is a group that contains all company employees as its members.

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/.

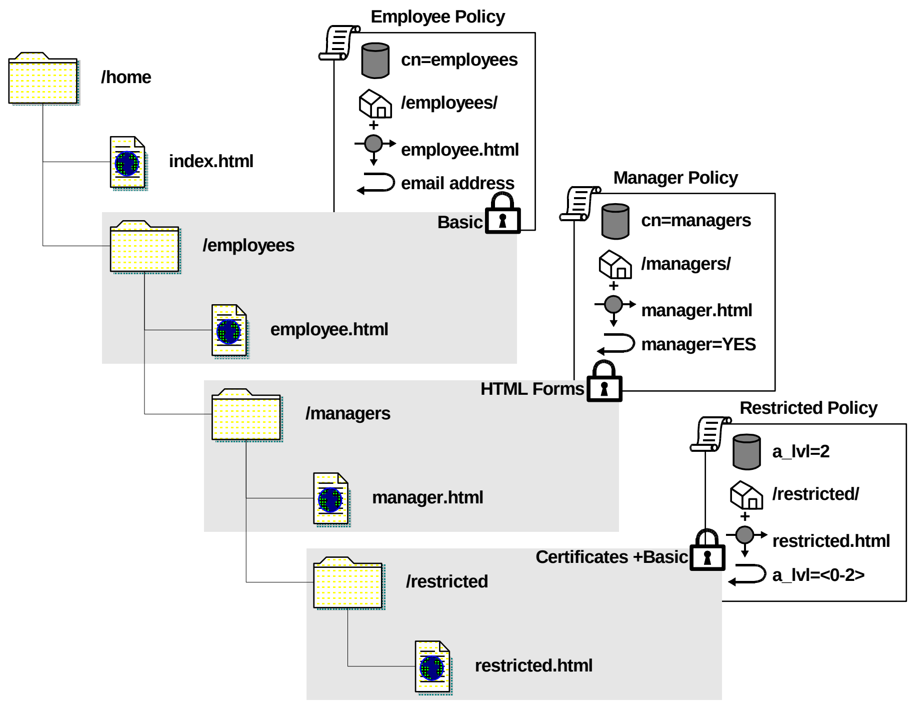
/restricted

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 428)

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 370)

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 17: Enterprise Policy Management (EPM)

This section contains the following topics:

Securing Applications Using EPM

Enterprise Policy Management (EPM) is an access management model that lets you protect businesses applications without requiring 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 are only required to provide data for configuration settings that do not have defaults; modifying other settings is optional. This makes policy configuration more straight-forward. You can manipulate additional SiteMinder settings that allow you to define more fine-grained protection of an application; however, this is not required.

For the administrator already familiar with SiteMinder, there is a relationship between the application-oriented concepts and the underlying SiteMinder components, which is reflected in the Administrative UI. The following table shows this relationship.

Application Dialogs and Group Boxes	**Underlying SiteMinder Component**
General settings	Defines the policy domain
Components	Defines the realm
Resource	Specifies the rule
Application Roles	Replaces the function of user directory lookups

EPM introduces the *application role*. An application role defines a set of users who have access to a resource or group of resources. The set of users is identified by a named or unnamed expression. Application roles lets you define privileges for users requesting access to an application.

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 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. Application roles are defined by expressions that search the user directories for users that meet the membership criteria of the application role.
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 486) 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.

To assign the application and policy administration rights

1. Click Administration, Administrator, Create Administrator.

   The Create Administrator pane opens.
2. Enter a name for the administrator in the Name field.
3. Click Create in the Rights group box.

   The Rights dialog opens.

4. In the table, choose one or more of the following:
    - application administration
    - policy administration
5. Click OK.

The administrator has been given application and policy administration privileges.

EPM Use Cases for Protecting Applications

Learn how to apply EPM to application security policies by reviewing the following use cases:

- Application Security Policy to Protect a Web Portal (see page 486)
- Application Security Policies Based on Roles (see page 491)
- Application Security Policies with User Mapping and Named Expressions (see page 499)

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

To identify the web portal and select the directory server

1. Click Policies, Application, Create Application.

   The Create Application pane opens.

2. Enter values for the fields in the General group box. Choose distinctive values that help you remember its purpose or function, as shown in the following examples:

   Note: You can click Help for a description of fields, controls, and their respective requirements.

   Name

      Sample Software Company Portal

   Description

      Allows access to all parts of the portal except the downloads area.

3. In the Components group box, specify the directory that contains the resources you want to protect. For this web portal use case, use the following example:

   Agent Type

   Web Agent

   Agent

   PortalAgent

   Resource Filter

   /downloads

4. Accept the defaults for the remaining settings.
5. In the User Directories group box, click Add/Remove.

   The Choose User Directories dialog opens.

6. Select 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.
7. Click OK.

   You return to the General tab.

8. 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 sub-directories 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. Choose 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: You can click Help for a description of fields, controls, and their respective requirements.

4. Verify that the Effective Resource matches what you want to protect. For this use case, the effective resource should be:

   Effective Resource

   /downloads*

   This string indicates that all of the resources in the /downloads directory are protected.

5. Ensure 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, a role for the web portal's registered customers must be created. A role associates resources with groups of users.

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

To create web portal roles

1. Click the Roles tab.
2. Click Create.

3. Ensure 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.

   Expression

   TRUE

   You can use the Expression Editor to complete this field or type in an expression. To access the Expression Editor, click Edit.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

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 attribute directory mapping for the user directory.
2. Create an application.
3. Select the user directory where you locate the users that meet the role criteria.
4. Specify the resources that are the sub-components of the main application.
5. Define the two roles that should have access to the application.
6. 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, Applications.
2. Click Create Applications

   The Create Application pane opens.
3. Click the General tab.
4. Enter values for the fields in the General group box. For this use case, the following data is specified:

   Name

      HR Application

   Description

      Identifies the internal human resources application
5. Enter values for the fields in the Components group box. For this use case, the following data is specified:

   Agent Type

      Web Agent

   Agent

      hrportal agent

   Resource Filter

      /benefits

Default Resource Protection

Protected

Authentication Scheme

Basic

Note: You can click Help for a description of fields, controls, and their respective requirements.

6. 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.

Create an Attribute Mapping for Group Membership

To indicate that a subset of the employees at Acme-Financial are Managers, create an attribute mapping that maps to the Managers group membership.

Note: The name of the LDAP user directory for this use case AcmeLDAP.

To create an attribute mapping for group membership:

1. Click Infrastructure, Directory, User Directory, Modify Directory.
2. Select the directory you want to modify.

   The Modify AcmeLDAP dialog opens.
3. Click Create in the Attribute Mapping List.

   The Create Attribute Mapping dialog opens.
4. Complete the fields in the General group box as follows:

   Name

   IsManager

   Description

   Defines the Managers group.

5. Select the Group radio button in the Properties group box, the enter the following:

   Definition

   cn=managers,ou=Groups,o=acme-financial.com

   The value in the Definition field reflects how the group is defined in the user directory, as indicated in the use case introduction.

6. Click OK.

   You return to the user directory dialog.

7. Click Submit to submit the changes.

You have now defined an attribute called IsManagers for the group cn=managers,ou=Groups,o=acme-financial.com.

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: You can click Help for a description of fields, controls, and their respective requirements.

The resources associated with the performance management application are now defined.

Create Employee and Manager Roles

After defining the specific components of an application that require protection, you can specify the roles that users may be assigned. Roles are the set of users who have access to a particular resource. These sets of users are defined by an expression.

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

To create a role

1. Click the Roles tab.
2. Click Create.

   The Create Role pane appears.
3. Verify that the Create radio button 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

5. Enter an expression in the Membership group box. For this use case, enter the following:

   Expression

      TRUE

   To form an expression, you can use the Expression Editor (see page 205). To access the editor, click Edit.

6. Click Submit.

7. Repeat steps 2–4 to create a second role called Managers, as follows:

   Name

      Managers

   Description

      Managers of Acme Financial Services

   Expression

      BOOLEAN(IsManager)

      IsManager is the attribute mapping that was defined for the LDAP user directory.

Note: You can click Help for a description of fields, controls, and their respective requirements.

More information:

Create an Attribute Mapping for Group Membership (see page 493)

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.

To specify metadata for the human resources 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_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 may 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 company's 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 and expression to establish roles.
4. Create a response to further customize the application.
5. Create an application security policy.

More information:

Named Expressions (see page 198)

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 216) 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 199) 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, Applications.
2. Click Create Applications

   The Create Application pane opens.
3. Click the General tab.

4. Enter values for the fields in the General group box. For this use case, the following data is specified:

   Name

      Online Catalog

   Description

      Identifies the clothing stores Web-based shopping application

5. Enter values for the fields in the Details group box. For this use case, the following data is specified:

   Agent Type

      Web Agent

   Agent

      Web Retail Agent

   Resource Filter

      /webcatalog

   Default Resource Protection

      Protected

   Authentication Scheme

      Basic

   Note: You can click Help for a description of fields, controls, and their respective requirements.

6. 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, Applications, Application, Modify Application.
2. Click Search and select the Online Catalog application. Click Select.

   The Modify Application dialog opens.
3. Select the Resources tab.
4. Click Create in the Resources group box.

   The Create Resource dialog opens.
5. Enter values for the fields in the General group box. For this use case, enter the following:

   Name

   Checkout

   Description

   Lets you total your purchases and pay for them.
6. Enter values for the fields in the Attributes group box. For this use case, enter the following:

   Resource

   total_charges.jsp
7. Select the Web Agent actions radio button in the Action group box and select the actions Get and Post.
8. Click OK.

You have created a resource called Checkout.

Note: You can click Help for a description of fields, controls, and their respective requirements.

Configure the Customer Role

After establishing a resource, you 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 you are creating a new object. You can also copy the properties of an existing object to create an object. More information exists in Duplicate Policy Server Objects.

To create this credit-based role

1. Click the Roles tab.
2. Click Create.

   The Create Role dialog appears.
3. Verify that the Create radio button 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 will use credit to pay for their purchases.
5. Enter an expression in the Membership Expression group box. For this use case, enter the following:

   Expression

   @IsUnderCreditLimit

   The role expression is the product of the two virtual user attribute expressions #Balance and #CreditLimit, which calculate whether or not the user has exceeded his credit limit.
6. Click OK.

You have created a role called 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.

To create the application security policy

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 allows all customers to make with the store's credit card provided 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.

Advanced Policy Components for Applications

EPM provides 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 prior to 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 with the EPM meet the requirements of the organization or of any regulations with which the organization must comply.

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

To create a policy using advanced options

1. Click Policies, Application, Create Application.

   The Create Application pane appears.

2. Enter information in the General and Components sections, then click Advanced.

   The Advanced pane appears.

3. Do one of the following tasks:

   - To create a new realm, click New Realm.
   - To change an existing realm, locate the realm you want to change in the list and click the edit arrow next to its name.

4. Configure the advanced options for the realm.

5. When you are finished, click OK to save your changes.

   The Advanced pane appears. The advanced options are configured for the policy.

6. Click Close.

   The Create Application pane appears.

7. Continue configuring the other parts of the policy.

More information:

Realms Overview (see page 377)
Rules Overview (see page 389)
Session Timeouts (see page 75)
Directory Mapping Overview (see page 237)
Authentication Events (see page 393)
Authorization Events (see page 395)

Chapter 18: 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 513)
Policies (see page 439)
eTelligent Rules Properties Files (see page 511)

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 511)
Modify the LoggerConfig.properties File (see page 511)

JVMOptions.txt File

The JVMOptions.txt file contains the settings that the Policy Server uses when creating the Java Virtual Machine that is used to support eTelligent Rules.

If you encounter errors related to missing classes, you may need to modify the classpath directive in the JVMOptions.txt file. For complete information about the settings contained in the JVMOptions.txt file, see your Java documentation.

Modify the LoggerConfig.properties File

On the Policy Server, the LoggerConfig.properties file allows you to specify logging features that are used when you start the SiteMinder service from a command line. The properties contained in this file are not used when the service is started from the Policy Server Management Console. The settings in this file are generally only used for debugging purposes.

You may want to modify this file to obtain more output for debugging purposes.

The following shows an example of a LoggerConfig.properties file.

```
// LoggingOn can be Y, N
LoggingOn=Y

// LogLevel can be one of LOG_LEVEL_NONE, LOG_LEVEL_ERROR,
LOG_LEVEL_INFO, LOG_LEVEL_TRACE
LogLevel=LOG_LEVEL_TRACE

// If LogFileName is set Log output will go to the file named
LogFileName=affwebserv.log

// AppendLog can be Y, N.   Y means append output to LogFileName if
specified
AppendLog=Y

// AlwaysWriteToSystemStreams can be Y, N.
// Y means log messages are written to System.out
// or System.err regardless of what the logger streams are
// set to.   If the logger streams are set to System.out
// or System.err log messages will be written multiple times.
// This facilitates logging messages to System.out/System.err
// and a file simultaneously.
AlwaysWriteToSystemStreams=N

// DateFormatPattern can be any valid input to java.text.DateFormat
constructor.
// See the Java documentation for java.text.DateFormat for details
// If not specified, the default format for the default locale is used
DateFormatPattern=MMMM d, yyyy h:mm:ss.S a
```

The settings in this file are:

LoggingOn

Enables or disables logging. Set this parameter to Y to enable logging. Set this parameter to N to disable logging.

LogLevel

Indicates the level of detail contained in logs. The LogLevel can be one of the following:

LOG_LEVEL_NONE

No messages will be logged.

LOG_LEVEL_ERROR

Only records error messages.

LOG_LEVEL_INFO

Records error messages and warnings.

LOG_LEVEL_TRACE

Records error messages, warnings, and general processing information that may be useful for tracking problems.

LogFileName

If LogFileName is set, all log output will go to the file named in this parameter.

AppendLog

Indicates whether log information should be appended to an existing file at startup or a new file should be created at startup. Set this parameter to Y to append output to the file specified in the LogFileName parameter. Set this file to N if a new file should be created at startup.

AlwaysWriteToSystemStreams

Set this parameter to Y to log messages to System.out or System.err regardless of what the logger streams are set to. If the logger streams are set to System.out or System.err, log messages will be written multiple times. This facilitates logging messages to System.out/System.err and a file simultaneously.

DateFormatPattern

DateFormatPattern can be any valid input to java.text.DateFormat constructor. See the Java documentation for java.text.DateFormat for details.

If not specified, the default format for the default locale is used.

Variables Overview

In the context of Policy Server, variables are objects that can be resolved to a value which you can incorporate into the authorization phase of a request. The value of a variable object is the result of dynamic data and is evaluated at runtime. Variables provide a flexible tool for expanding the capabilities of policies and responses.

Variable Types

The following types of variables are available:

- Static Variables (see page 514)
- Request Context Variables (see page 514)
- User Context Variables (see page 515)
- Form Post Variables (see page 515)
- Web Services Variables (see page 515)

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 518)

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 442)

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

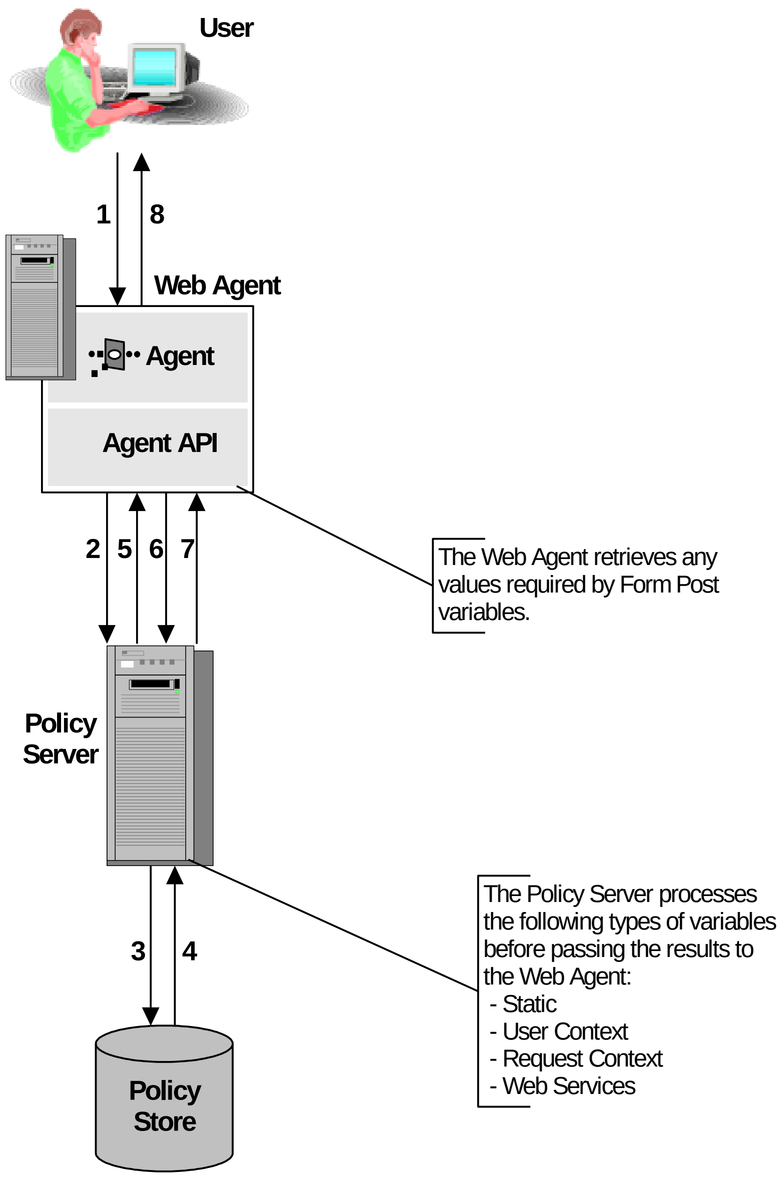
Variables are evaluated when the Policy Server processes an authorization for a request and determines that a user is authorized for the requested resource. The details of variable processing are slightly different based on whether the variable objects are contained in a response or in a policy expression.

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.


User
1
8
Web Agent
Agent
Agent API
2
5
6
7
The Web Agent retrieves any values required by Form Post variables.
Policy Server
3
4
Policy Store
The Policy Server processes the following types of variables before passing the results to the Web Agent:
- Static
- User Context
- Request Context
- Web Services

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 518)

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 may not be used for responses, all variables are resolved by the time a response fires at the Policy Server.

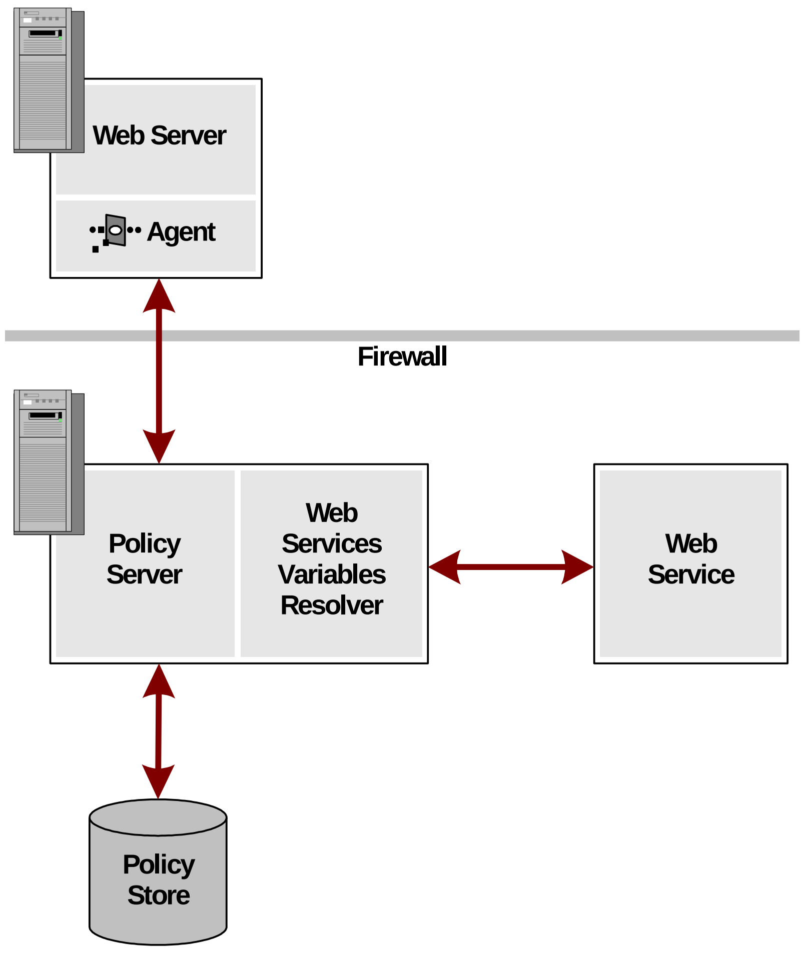
Web Service Variables

Web Service Variables provide a method for including dynamic data retrieved from a Web Service in a Policy Server 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’s 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 what is in a message 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, and thus 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, which runs in the same process space.

The user, when defining the Web Service variable, specifies the SOAP document to be sent to the Web Service, along with the authentication credentials and other parameters as defined in the rest of this document.

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, using the server-side certificates on the Web Service and a list of trusted CAs configured on the Policy Server side.

Component Requirements for Web Service Variables

Web service variables require a session server.

Note: More information on configuring a Session Server exists in the *Policy Server Installation Guide*. More information on upgrading an existing Session Server exists in 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). This is similar to the model used by SiteMinder 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. This means that the Policy Server needs to be configured with the list of trusted CAs.

Note: For SSL connections, server-side certificates should be configured for the Web Service. A listed of trusted CAs should be configured on the Policy Server. To configure trusted CAs, use the smkeydatabase tool 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 while resolving Web Services variables, you must configure a list of trusted Certificate Authorities (CAs) that can be used when the Policy Server establishes a connection to a Web Service. To accomplish this, you must set up an smkeydatabase for each Policy Server that is responsible for connecting to a Web Service.

The smkeydatabase is a flat-file key and certificate database that lets you store, manage, and retrieve keys and certificates required to sign and validate messages with WS-Security tokens. The service is also responsible for decrypting symmetric XML encryption keys that have been encrypted using the site's public key.

The SiteMinder smkeytool utility lets you create a new smkeydatabase or delete an existing one and create a new one. There can only be one key database per Policy Server.

Items Stored in the Key Database for WS-Security Documents

The following gets stored in the key database:

- An enterprise private key and certificate for signing WS-Security documents and generating X509v3 tokens.
- Public key certificates that correspond to the private keys used for signing the WS-Security documents and generating the tokens. These certificates are used to validate the WS-Security documents.

A given Policy Server may sign and/or verify WS-Security documents. Keys and certificates for signing and validation can be added to the same key database, depending on what the Policy Server is doing.

The following table shows which objects you need to add to the smkeydatabase to handle WS-Security signing and validation requirements.

Function	**WS-Security Token Type**	**Required Database Objects**
Signing	All	Private key and certificate of Web service host enterprise.
Generating X509 Tokens	X509v3	Private key and certificate of Web service host enterprise.
Signature Validation	SAML Assertion; Sender Vouches	Certificate of issuing Web service consumer application.

Function	WS-Security Token Type	Required Database Objects
	SAML Assertion; Holder-of-key	Certificates of XML request subject and issuing Web service consumer application.
	X.509v3; Username (if signed)	Certificate of trusted issuer.

Properties File for the Key Database

The smkeydatabase properties file, smkeydatabase.properties, defines the configuration parameters required to access and manage the key database.

The smkeydatabase.properties file is installed in:

- *policy_server_home*\config\properties (Windows)
- *policy_server_home*/config/properties (UNIX)

Modify this file only to change the following options:

- NativeDBName--specifies name of the key database
- DBLocation--indicates the directory where the key database resides
- DBUpdateFrequencyMinutes--specifies the frequency at which the database is read from the file system.

The smkeydatabase.properties file contains the following settings:

- DBLocation (see page 524)
- NativeDBName (see page 524)
- XMLDocumentOpsImplementation (see page 524)
- AffiliateIXMLSignatureImplementation (see page 524)
- IXMLSignatureImplementation (see page 524)
- EncryptedPassword (see page 525)
- IXMLEncryptDecryptImplementation (see page 525)
- DBUpdateFrequencyMinutes (see page 525)

Descriptions of each setting follow.

DBLocation Setting

Specifies the path to the directory where the database resides.

Enter the location that smkeytool should use when you manually create the database.

Default: *policy_server_home*/smkeydatabase

NativeDBName Setting

Identifies the name of the database.

Specify the name you want smkeytool to use when you create the database.

Default: smkeydatabase

XMLDocumentOpsImplementation Setting

Specifies the Java class that implements the XML signing and validation.

Note: Do not change this value; it is static and preconfigured.

Default: com.ca.smkeydatabase.api.XMLDocumentOpsImpl

AffiliateIXMLSignatureImplementation Setting

Specifies the Java class that implements low-level cryptographic operations for signing and validation.

Note: Do not change this value; it is static and preconfigured.

Default: com.ca.smkeydatabase.api.XMLSignatureApacheImpl

IXMLSignatureImplementation Setting

Specifies the Java class for Transactionminder that implements low-level cryptographic operations for signing and validation.

Note: Do not change this value; it is static and preconfigured.

Default: com.ca.smkeydatabase.api.XMLSignatureApacheImpl

EncryptedPassword Setting

Indicates the smkeydatabase password.

(Encrypted using the policy store key at database creation.) Prior to creating a key database, this entry contains a dummy value.

Default: *encrypted_password_string*

IXMLEncryptDecryptImplementation Setting

Identifies the Java class that implements the encryption and decryption of assertions, Name IDs, and attributes.

Note: Do not change this value; it is static and preconfigured.

Default: com.ca.smkeydatabase.api.XMLEncryptDecryptApacheImpl

DBUpdateFrequencyMinutes Setting

Indicates the frequency at which the database is read from the file system. Specifically, its the number of minutes after which the in-memory smkeydatabase expires and is reloaded.

Until this interval passes, certificates and keys added, removed, or changed in the database will not affect the Policy Server. If the value is 0, key database caching is disabled entirely. If the value is -1, the cache persists until the Policy Server is restarted.

Default: 60 minutes

Modify the Key Database Using smkeytool

Smkeytool is a SiteMinder command-line utility that allows you to populate and manage smkeydatabase. The smkeytool utility is installed with the Policy Server in the following locations:

- *policy_server_home*/bin (UNIX)
- *policy_server_home*\bin (Windows)

Use smkeytool to:

- Create and delete a key database

  You can only have one key database per Policy Server. After the database is created, you can add keys and certificates.
- Add and delete private keys
- Add and delete a partner certificate

- List all certificates stored in the key database
- Import root certificates of CAs
- Add client certificate keys

  If you are using a root or chain Certificate Authority (CA) at the consuming authority that is not listed in the smkeydatabase, you need to add it to the smkeydatabase.

  For example, a signed VeriSign CA server-side certificate is used to SSL-enable the producer-side web server installed with the Web Agent Option Pack. To use this certificate for Basic over SSL authentication, add the VeriSign certificate to the smkeydatabase at the consumer. This ensures that the consumer is communicating with a producer that can present a server-side certificate that has been verified by a trusted CA.
- Export key data from smkeydatabase
- Add, list, validate, and delete a Certificate Revocation List

Note: If you are adding a private key or certificate, delete the certificate metadata from the certificate file before trying to import it into the smkeydatabase. Import only the data starting with the --BEGIN CERTIFICATE-- marker and ending with the --END CERTIFICATE-- marker. Be sure to include the markers.

Smkeytool Command Syntax and Options

Smkeytool is a command-line utility that provides many options to manage the smkeydatabase.

Run the smkeytool utility from a command line, using the following syntax:

UNIX

smkeytool.sh -*option* [-*argument(s)*]

Windows

smkeytool.bat -*option* [-*argument(s)*]

If you enter smkeytool from a command line without any options, you will see a list of all command line options.

The smkeytool utility uses the following command options and arguments:

- createDB (see page 527)
- addPrivateKey (see page 528)
- addCertOption (see page 529)

- addRevocationInfo (see page 530)
- changepassword (see page 530)
- deleteRevocationInfo (see page 531)
- deleteDB (see page 531)
- delete (see page 531)
- export (see page 532)
- findAlias (see page 532)
- importDefaultCACerts (see page 532)
- listCerts (see page 533)
- listRevocationInfo (see page 533)
- printCert (see page 533)
- renameAlias (see page 534)
- validateCert (see page 534)
- help (see page 534)

A description of each command option follows.

createDB Option

Creates an new smkeydatabase to store keys and certificates. By default, the directory is named smkeydatabase. You can change the smkeydatabase location by modifying the smkeydatabase.properties file.

All private keys in the smkeydatabase are encrypted using FIPS-compliant algorithms.

Important! To store multiple keys in the database, you must define the first key you add with the alias defaultenterpriseprivatekey before you can add subsequent keys.

Arguments for -createDB are as follows:

-password <*password*>

Required. The password is used to store all data in an encrypted format in the key database. It can be a value from 6 to 32 characters. It is encrypted using the policy store key and added to the smkeydatabase.properties file.

-importDefaultCACerts

(Optional) Imports the default Certificate Authority (CA) certificates during the creation of the database. These certificates are imported from the cacerts.keystore file, which is installed with the Policy Server and contains all default CA certificates. This option is the same as executing the -importDefaultCACerts option.

addPrivKey Option

Adds a private key and certificate pair to the key database. You can have multiple private keys and certificates in the database, but only RSA keys are supported.

Note: Only private keys are stored in the smkeydatabase in encrypted form.

The Policy Server at the producing authority uses a single enterprise private key to sign SAML messages and to decrypt encrypted SAML messages received from the consuming authority. Typically, the enterprise key is the first private key found in the smkeydatabase.

When you use the -addPrivKey command, you can specify the key data by combining the -keyfile and -certfile options or by using the -keycertfile option alone.

Arguments for -addPrivKey are as follows:

-alias <*alias*>

Required. Assigns an alias to a single certificate or certificate/private key pair in the database.   The alias must be a unique string and should contain only alphanumeric characters.

-certfile <*cert_file*>

Specifies the full path to the location of the certificate associated with this private key. Required for keys in PKCS1, PKCS5, and PKCS8 format.

-keyfile <*private_key_file*>

Specifies the full path to the location of the private key file. Required for keys in PKCS1, PKCS5, and PKCS8 format.

-keycertfile <*key_cert_file*>

Specifies the full path to the location of the PKCS12 file that contains the private key and public certificate data. Required for keys in PKCS12 format.

-password <*password*>

Optional. Specifies the password that was used to encrypt the private key when the key/certificate pair was originally created. When a private key is added to the smkeydatabase, this password must be supplied to decrypt the private key before it gets written to the smkeydatabase.

Note: This password is not stored in the smkeydatabase.

After the key is decrypted and placed in the smkeydatabase, smkeydatabase encrypts the private key again using its own password, which is the password specified when the smkeydatabase was created.

addCert Option

Adds a certificate to the key database.V1, V2, and V3 versions for X.509 certificate format are supported DER and PEM encoding formats are supported. Restart the Web Agent when you add a Certificate Authority certificate.

If you indicate that you want to trust the certificate as a Certificate Authority, this certificate will always be treated as a CA certificate.

Arguments for addCert are as follows:

-alias <*alias*>

Required. Alias to the certificate associated with this private key in the database. Must be a unique string and should contain only alphanumeric characters.

-infile <*cert_file*>

Required. Full path to the location of the newly added certificate.

-trustcacert

Optional. Checks that the user provider certificate being added is a CA certificate. Smkeytool checks that the certificate has a digital signature extension and that the certificate has the same IssuerDN and Subject DN values.

-noprompt

(Optional) The user will not be prompted to confirm the addition of the certificate.

addRevocationInfo Option

Specifies the location of a CRL so the smkeydatabase can locate the list during the SAML authentication process. The smkeydatabase does not store the contents of a CRL, but merely reads the CRL contents when the Policy Server starts and after a refresh interval has elapsed.

Important! If you add a CRL entry to the smkeydatabase, you must restart the Policy Server.

Arguments for addRevocationInfo are as follows:

-issueralias <*issuer_alias*>

Required. Alias name of the Certificate Authority who issues the CRL.

Example: -issueralias verisignCA

-type (*ldapcrl* | *filecrl*)

Required. Specifies whether the list is a certificate file or an LDAP CRL. The options are ldapcrl or filecrl.

-location <*location*>

Required. Specifies the location of the CRL. For a file, specify the full path to the file. For an LDAP CRL, specify the full path to the LDAP server node.

Example of file location: -location c:\crls\siteminder_root_ca.crl

Example of LDAP CRL location: -location "http://localhost:880/sn=siteminderroot, dc=crls,dc=com"

changePassword Option

Permits you to change the password for the smkeydatabase. Changing the password causes all entries encrypted under the old password to be re-encrypted under the new password using FIPS-compliant algorithms.

Arguments for changePassword are as follows:

-password <*password*>

Required. Specifies the existing password used to originally create the smkeydatabase

-newpassword <new_*password*>

Required. Specifies the new password for the smkeydatabase.

deleteRevocationInfo Option

Deletes a CRL from the database.

Arguments for -deleteRevocationInfo are as follows:

-issueralias <*issuer_alias*>

Required. Name of the Certificate Authority who issues the CRL.

-noprompt

(Optional) The user will not be prompted to confirm the deletion of the CRL from the database.

deleteDB Option

Deletes the smkeydatabase based on configuration data in the smkeydatabase.properties file. All the entries in the key database and the aliases data store file will be deleted.

Argument for -deleteDB is as follows:

-noprompt

(Optional) The user will not be prompted to confirm the deletion of the database.

delete Option

Deletes an existing certificate from the smkeydatabase. If the certificate has an associated private key, the key is also deleted.

Argument for -delete is as follows:

-alias <*alias*>

Required. Alias of the certificate to be removed.

-noprompt

(Optional) The user will not be prompted to confirm the deletion of the database.

export Option

Exports an existing certificate or a private key from the smkeydatabase to a file. Certificate data is exported using PEM encoding. Private key data is exported using DER encoded PKCS8 format.

Arguments for the -export option are as follows:

-alias <*alias*>

Required. Identifies the certificate or key to be exported.

-outfile <*out_file*>

Required. Specifies the full path to the output file for the exported certificate or key.

-type (key|cert)

Optional. Indicates whether a certificate or key is being exported. If no option is specified, a certificate is the default.

-password <*password*>

Required only when exporting a private key. Specifies the password used to encrypt the private key at the time the key gets exported to a file. You do not need a password to export the certificate holding the public key because certificates are exported in clear text.

When a private key is exported, it gets exported to the output file in encrypted form using this password. To add this same private key back to the smkeydatabase, run the -addPrivKey command and use this password.

findAlias Option

Determines the alias associated with a certificate that is already in the smkeydatabase.

Argument for -findAlias is as follows:

-infile <*cert_file*>

Required. Full path to the certificate file associated with the alias you want to find

-password <*password*>

Password required only when a password-protected P12 file is specified as the certificate file.

importDefaultCACerts Option

Imports all default trusted Certificate Authority certificates from the cacerts.keystore file, which is installed with the Policy Server, into the smkeydatabase. Certificate Authority certificates are used to verify the server certificate associated with the producing authority's web server.

listCerts Option

Lists some metadata of all the certificates stored in key database.

Argument for -listCerts is as follows:

-alias <*alias*>

(Optional) Lists the metadata details of the certificate and key associated with the alias specified. This option supports the asterisk (*) as a wildcard character. You can use this wildcard at the beginning and/or at the end of an alias value. Always enclose the asterisk in quotes to avoid a command shell from interpreting the wildcard character.

listRevocationInfo Option

Displays a list of current CRLs in the smkeydatabase. The -listRevocationInfo option only prints the CRL name, type (file or ldap), and the location of all the CRLs in the database.

Argument for -listRevocationInfo is as follows:

-issueralias <*issuer_alias*>

(Optional) Name of the Certificate Authority who issues the CRL. This option supports the asterisk (*) as a wildcard character. You can use this wildcard at the beginning and/or at the end of an alias value. Always enclose the asterisk in quotes to avoid a command shell from interpreting the wildcard character.

printCert Option

Displays some metadata of the specified certificate. This command is especially useful for UNIX systems, where it is difficult to see the certificate properties.

Arguments for -printCert are as follows:

-infile <*cert_file*>

Required. Location of the certificate file.

-password <*password*>

Password required only when a password-protected P12 file is specified as the certificate file.

renameAlias Option

Renames an existing alias associated with a certificate.

Arguments for -renameAlias are as follows:

-alias <*current_alias*>

Required. Current alias associated with a certificate.

-newalias <*new_alias*>

Required. New alias name. Value must be a unique string and should contain only alphanumeric characters.

validateCert Option

(Optional) Indicates whether a certificate is revoked or not.

Arguments for -validateCert are as follows:

-alias <*alias*>

Required. Alias to the certificate associated with this private key in the database. Must be a unique string and should contain only alphanumeric characters.

-infile <*crl_file*>

Optional. Specifies the CRL file that you want smkeytool to look in for the certificate to validate it.

help Option

Shows how to use the smkeytool utility.

Smkeytool Examples for UNIX Platforms

Example: Create a key database

This example shows the command for creating an smkeydatabase:

```
smkeytool.sh -createDB -password siteminderdb
```

Example: Add a private key and certificate

This example shows the command to add a private key and certificate to the smkeydatabase. The example assumes you are running the smkeytool from the directory where the certificates and keys are located, as follows:

```
smkeytool.sh -addPrivkey -password keypswd -alias idp1privkey -keyfile privkey.pkcs8 -certfile sample.crt
```

If you are not running smkeytool from the directory where the certificates and keys are located, you need to specify the full path to directory where these items are located, as follows:

```
smkeytool.sh -addPrivkey -alias privkey1 -keyfile "export/ca/siteminder/certs/
sampleprivkey.pkcs8" -certfile "export/ca/siteminder/certs/samplecert.crt"
```

Example: Add an trusted CA certificate

This example shows the commands required to add a trusted certificate authority certificate:

Important! Before adding a trusted certificate, obtain a CA certificate from a certificate authority.

To add a trusted CA certificate:

1. Check whether it already exists in the consuming authority database by entering:
   ```
   smkeytool.sh -listCerts
   ```
2. Add the CA certificate by entering:
   ```
   smkeytool.sh -addCert -alias -sp1cacert -infile /opt/netegrity/siteminder/certs/sampleCARoot.cer -trustcacert
   ```

Smkeytool Examples for Windows Platforms

Example: Create a key database

This example shows the command for creating an smkeydatabase:

```
smkeytool.bat -createDB -password smdb
```

Example: Add a private key and certificate

This example shows the command to add a private key and certificate to the smkeydatabase. The example assumes you are running the smkeytool from the directory where the certificates and keys are located, as follows:

```
smkeytool.bat -addPrivkey -password keypswd   -alias privkey1
-keyfile sampleprivkey.pkcs8" -certfile samplecert.crt"
```

If you are not running smkeytool from the directory where the certificates and keys are located, you need to specify the full path to directory where these items are located, as follows:

```
smkeytool.bat -addPrivkey -password keypswd -alias privkey1 -keyfile "c:\program
files\ca\siteminder\certs\sampleprivkey.pkcs8"
-certfile "c:\program files\ca\siteminder\certs\samplecert.crt"
```

Example: Add an trusted CA certificate

This example shows the commands required to add a trusted certificate authority certificate:

Important! Before adding a trusted certificate, obtain a CA certificate from a certificate authority.

To add a trusted CA certificate:

1. Check whether it already exists in the consuming authority database by entering:
   ```
   smkeytool.sh -listCerts
   ```
2. To add the CA certificate enter:
   ```
   smkeytool.bat -addCert "c:\program files\ca\siteminder\certs\sampleCARoot.crt" -trustcacert
   ```

Create and Manage the Key Database Using Smkeytool

The smkeytool command-line utility allows you to populate and manage the key database. This tool is installed with the Policy Server.

Use smkeytool to:

- Create and delete a key database

  You can only have one key database per Policy Server. After the database is created, you can add keys and certificates.
- Add and delete the private key and certificate
- Add and delete an issuer certificate
- List all certificates stored in the key database

Note: smkeytool relies on values in the smkeydatabase.properties file. Ensure that this file is properly configured before running smkeytool.

smkeytool is located in the following directory:

- *<policy_server_home>*/bin (UNIX)
- *<policy_server_home>*\bin (Windows)

Run the smkeytool utility from a command line, using the following syntax:

UNIX:

```
smkeytool.sh option [argument(s)]
```

Windows:

```
smkeytool.bat option [argument(s)]
```

The options and arguments are described in the following table.

Option	Arguments	Function
-createDB or -cdb	*<password>*	Creates an empty key database to store keys and certificates. The specified password is encrypted using the policy store key and added to the smkeydatabase.properties file.
-deleteDB or -ddb	None	Deletes the key database specified in the smkeydatabase.properties file.
-addPrivKey or -apk	*<private_key_filepath> <x.509_certificate_filepath> <password>*	Adds the specified private key and corresponding certificate file to the key database. Note that *<password>* is the password used to encrypt the private key file being loaded, not the one associated with the key database.
-deletePrivKey or -dpk	*<x.509_certificate_filepath>*	Deletes the private key entry from the key database based on the specified certificate.

Option	Arguments	Function
-addCert or -ac	*<x.509_certificate_filepath>*	Adds a certificate to the key database.
-deleteCert or -dc	*<x.509_certificate_filepath>*	Deletes a certificate from the key database based on the specified certificate.
-listCerts or -lc	None	Lists the issuer/subject name (DN) and serial number of all the certificates stored in key database.
-help or -h	None	Lists smkeytool usage information.

Smkeytool Examples

- Create a key database

  UNIX:

  smkeytool.sh –cdb password

  Windows:

  smkeytool.bat –cdb password

- Add a private key and certificate:

  UNIX:

  smkeytool.sh –apk /opt/netegrity/webagent/certs/samplePrivKey.pkcs8 /opt/netegrity/webagent/certs/sampleRobm.cer passphrase

  Windows:

  smkeytool.bat –apk "c:\program files\netegrity\webagent\certs\samplePrivKey.pkcs8" "C:\program files\netegrity\webagent\certs\sampleRobm.cer" passphrase

- Add an issuer certificate:

  UNIX:

  smkeytool.sh –ac /opt/netegrity/webagent/certs/sampleCARoot.cer

  Windows:

  smkeytool.bat –ac "c:\program files\netegrity\webagent\certs\sampleCARoot.cer"

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 369)

Create a Static Variable

You create a static variable to make it available for use in policies or responses.

Note: The value of the resolved variable may be no larger than 1K.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.
3. Click Create Variable.

   The Create Variable pane opens.
4. Verify that Create a new object is selected, and click OK.

   Variable settings open.
5. Type the variable name in the Name field.
6. Select Static from the Variable Type list.

   Static variable settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
7. Specify the data type and value of the variable in the Variable Information group box.
8. Click Submit.

   The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Create a Request Context Variable

You create a request context variable to make it available for use in policies or responses.

Note: The value of the resolved variable may be no larger than 1K.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.
3. Click Create Variable.

   The Create Variable pane opens.
4. Verify that Create a new object is selected, and click OK.

   Variable settings open.
5. Type the variable name in the Name field.
6. Select Request Context from the Variable Type list.

   Request context settings open.
7. Select the variable value from the Property list.
8. Click OK.

   The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Create a User Context Variable

You create a user context variable to make it available for use in policies or responses.

Note: The value of the resolved variable may be no larger than 1K.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.
3. Click Create Variable.

   The Create Variable pane opens.

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

   Variable settings open.

5. Type the variable name in the Name field.
6. Select User Context from the Variable Type list.

   User context settings open.

7. Select the portion of the user context that provides the value of the variable from the Property list.

   The return type value appears as either string or boolean depending on the value you selected from the Property list.

8. (Required for User Property and Directory Entry) Enter the name of the directory or user attribute that provides the variable value in the Property field.
9. (Required for User Property and Directory Entry) Enter the size of the buffer (in bytes) that is to store the variable in the Buffer field.
10. (Required for Directory Entry) Enter the distinguished name of the directory entry in the DN field.
11. Click Submit.
12. The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

Create a Form Post Variable

You create a Form Post variable to make it available for use in policies or responses.

Note: The value of the resolved variable may be no larger than 1K.

To create a variable

1. Open the domain to which to you want to add a variable.
2. Click the Variables tab.

   A table lists the variables associated with the domain.

3. Click Create Variable.

   The Create Variable pane opens.

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

   Variable settings open.

5. Type the variable name in the Name field.

6. Select Post from the Variable Type list.

   Form post settings open.

7. Enter the name of the POST variable contained in the form in the Form Field Name field.

8. Click OK.

   The variable appears in the Variables tab of the domain. The variable can now be used in policy expressions or responses.

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 may be no larger than 1K.

To create a variable

1. Open the domain to which to you want to add a variable.

2. Click the Variables tab.

   A table lists the variables associated with the domain.

3. Click Create Variable.

   The Create Variable pane opens.

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

   Variable settings open.

5. Type the variable name in the Name field.

6. Select Web Service from the Variable Type drop-down list on the General group box.

   Web Service settings appear.

7. Select the data type from the Return Type drop-down list on the Definition group box.

8. Type the Web Service URL in the URL field on the Remote URL group box.

9. Type the XPath query in the XPath field on the Return Query group box.

   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.

10. (Optional) Select the Require Credentials check box on the Web Service Credentials group box and specify the user name and password that the Policy Server is to use when connecting to the Web Service.

11. (Optional) Click Variable... in the SOAP Document group box to add existing variables to the SOAP message.
12. (Optional) Click Add on the HTTP Headers group box to associate an HTTP header with the Web Service variable.
13. Click OK.

    The variable appears on the Variables tab of the domain and can now be used in policy expressions or responses.

Chapter 19: 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 439)
Authentication Events (see page 393)
Authorization Events (see page 395)

Global Policy Object Characteristics

The following sections discuss the characteristics of global policy objects, outlining the basic similarities and differences when compared to their standard (non-global) counterparts.

Global response vs. standard response

Differences:

- Defined at the system level. Only system level administrators can define a global response.
- Cannot use variables-based attributes.
- Used in any global or domain-specific policies.
- Associated with the specific agent type.
- Can be a member of any global or domain-specific response group.

Similarities:

- Can use active expressions.
- Is not returned unless it is specified in a particular policy.

Global rule vs. standard rule

Differences:

- Defined at the system level. Only system level administrators can define a global rule.
- The filter for the global rule is not bound to a specific realm. The filter for the global rule is an absolute filter, which may or may not use a regular expression.
- Bound to specific agent or agent group. The agent is explicitly specified when the rule is created.
- Available only for 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 vs. standard policy

Differences:

- Defined at the system level. Only system level administrators can define a global policy.
- Bound to all users. Specific users cannot be included or excluded from a global policy.

  Note: Individual domains can be explicitly enabled or disabled for global policy processing.
- Defined using only global rules, global responses and groups of these global objects.
- Cannot use variable-based attributes or variable expressions.
- Cannot contain allow/deny access rules. Only event rules may be included in a global policy.
- Cannot include or exclude a particular resource/realm from the global policy. The global policy is applicable to all resources that match at least one of the rule filters defined for the policy, on the domain for which global policy processing is enabled.
- Are not supported through the Java Policy Management API.

Similarities:

- Can use active expressions.
- Associated with the specific Agent. However, it’s possible to create a group containing all the Agents of the same type and bind a global rule to such group.

When the global policy is being processed, the responses defined for the fired global rules, are added to the list of other responses. A global rule fires when the following is true:

- The resource being accessed matches the absolute resource filter defined for the global rule.
- The event that occurs is as defined for the global rule.

- The resource being requested is protected by the same agent/agent group, which was specified for the rule.
- The resource/realm being accessed belongs to a domain for which global policies processing is enabled.

More information:

Disable Global Policy Processing for a Domain (see page 375)

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:

- Grouping Resources in Realms (see page 382)
- Rule Groups (see page 409)
- Rules (see page 389)
- Responses and Response Groups (see page 413)
- Policies (see page 439)

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 441)

Global Policy Processing

Policies are evaluated as described in Policy Processing (see page 473). In addition, any global rules contained in global policies will fire if the following conditions are met:

- The requested resource belongs to a domain on which global policy processing was not disabled
- The requested resource matches the absolute resource filter defined for the global rule. It is important, that in the case of global rule the filter will be obtained from the rule (not from the realm).
- The event that occurs is the same as that defined for the global rule.
- The requested resource is protected by the same agent or agent group, which was specified for the rule.

Whenever an authentication or an authorization event happens the responses defined for the fired global rules are added to the list of other responses.

How to Configure Global Policies

A global policy is comprised of global rule objects and global response objects, including response attributes. The following process lists the procedures for creating a global policy:

1. Create a Global Rule for Authentication Events (see page 551) or Create a Global Rule for Authorization Events (see page 553)
2. Configure a Global Response (see page 557)
3. Configure a Global Web Agent Response Attribute (see page 558)
4. Configure the Global Policy (see page 559)

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 556)

Create a Global Rule for Authentication Events

You create a global rule for authentication events to control actions that occur when users authenticate to gain access to a resource.

To create a global rule

1. Click Policies, Global.
2. Click Global Rule, Create Global Rule.

   The Create Global Rule pane appears.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

3. Enter the global rule name.
4. Specify agent and resource settings in the Realm and Resource group box.

   Note: If you specify an Agent Group and have also configured domain-specific rules associated with the same resource, you may adversely affect system performance by effectively duplicating processing steps. Consider domain-specific rules that may duplicate the responses generated by global rules. In such cases, only one response is returned to the Agent because the Policy Server automatically deletes duplicate responses before passing information back to the requesting Agent.
5. Select Authentication events from the Action group box.
6. Select an OnAuth event from the Action List.
7. Click Submit.

   The global rule is saved.

Global Rules for Authorization Events

Global rules that include 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 authentication events to control actions that occur when users authenticate to gain access to a resource.

To create a global rule

1. Click Policies, Global.
2. Click Global Rule, Create Global Rule.

   The Create Global Rule pane appears.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

3. Enter the global rule name.
4. Specify agent and resource settings in the Realm and Resource group box.

   Note: If you specify an Agent Group and have also configured domain-specific rules associated with the same resource, you may adversely affect system performance by effectively duplicating processing steps. Consider domain-specific rules that may duplicate the responses generated by global rules. In such cases, only one response is returned to the Agent because the Policy Server automatically deletes duplicate responses before passing information back to the requesting Agent.
5. Select Authentication events from the Action group box.
6. Select an OnAuth event from the Action List.
7. Click Submit.

   The global rule is saved.

More information:

Responses and Response Groups (see page 413)
Start the Administrative UI (see page 50)
Resource Matching and Regular Expressions (see page 403)

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.

More information:

Start the Administrative UI (see page 50)

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: You can click Help for a description of fields, controls, and their respective requirements.
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 406)

Configure an Active Global Rule

You configure an active rule for dynamic authorization based on external business logic. The Policy Server invokes a function in a customer-supplied shared library. This shared library must conform to the interface specified by the Authorization API, which is available separately via the Software Development Kit

Note: More information on shared libraries exists in the *API Reference Guide for C*.

To configure an Active Rule

1. Select the Edit Active Rule check box in the Active Rule group box.

   Active rule fields appear.

   Note: You can click Help for a description of fields, controls, and their respective requirements.

2. Specify the library name, function name, and function parameters in the respective fields.

   The active rule string appears in the Active Rule field.

3. Click Submit.

   The active rule is saved.

Delete a Global Rule

If you delete a global rule, the rule is automatically removed from any global policies that include the global rule. The global policies remain on your system. Verify that the global policies function without the deleted rule.

Global policies must contain at least one global rule.

Note: More information on modifying and deleting Policy Server objects exists in Manage Policy Server Objects (see page 51).

Global Response Objects

Global responses are the part of a global policy that define the attributes to be returned after a user triggers the authentication or authorization event specified in a global rule.

Note: You may use global responses in domain policies. In order to be returned, a global response must be added to a domain-specific or global policy. Within policies, the global response will be processed like a domain-specific response.

Configure a Global Response

You configure a global response to define the attributes that are returned after the authentication or authorization event occurs in an associated global rule.

To configure a global response

1. Click Policies, Global.
2. Click Global Response, Create Global Response.

   The Create Global Response pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Enter the global response name.
4. Select a SiteMinder Agent type from the SiteMinder Agent type list.
5. Click Submit.

   The global response is saved. You can now add response attributes to the response. See the next section for details about adding response attributes for each Agent type.

More information:

Start the Administrative UI (see page 50)

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 on the Attribute List group box on the Response pane.

   The Create Response Attribute pane opens.

2. Select a response attribute from the drop-down list.

   Note: Complete descriptions of response attributes exist in the *Web Agent Configuration Guide*.

3. Select an attribute type on the Attribute Kind group box.

   The fields on the Attribute Fields group box are updated to match the specified attribute type.

4. Complete the fields on the Attribute Fields group box.

   Note: A list of automatically generated SiteMinder user attributes that you can use in responses exists in SiteMinder Generated User Attributes (see page 429).

5. Specify Cache Value or Recalculate value every ... seconds on the Attribute Caching group box.

6. Click Submit.

   The Create Response Attribute Task is submitted for processing, and the response attribute is added to the Attribute List on the Response pane.

More information:

Global Response Attribute Types (see page 557)

How to Configure Global Policy Objects

Configuring a global policy requires you to complete the following procedures:

1. Create the Global Policy (see page 559)
2. Add Global Rules to the Global Policy (see page 559)
3. (Optional) Associate a Global Rule with a Response (see page 560)

More information:

Start the Administrative UI (see page 50)

Create the Global Policy

You create a global policy to define how users interact with resources.

To create a global policy

1. Click Policies, Global.
2. Click Global Policy, Create Global Policy.

   The Create Global Policy pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Enter the global policy name.
4. Add global rules and global responses.

Add Global Rules to a Global Policy

Global rules indicate the specific resources included in a global policy. You must add at least one global rule to a global policy.

To add global rules to a global policy

1. Click the Rules tab.

   The Rules group box opens.
2. Click Add Rule.

   The Available Rules pane opens and lists the available global rules.

   Note: If the global rule you require does not appear, click New Rule. Rules you create in this manner are added to the global policy.
3. Select the global rules you want to add, and click OK.

   The Rules group box lists the selected rules and rule groups.
4. (Optional) Associate the rule with a response or response group.

Associate a Global Rule with a Response

Global responses indicate the actions that should take place when the rule fires. When the rule fires, the associated response also fires.

To associate a response with a global rule

1. Click Add Response for the global rule for which you want to associate a response.

   The Available Responses pane opens and lists the available responses, response groups, and global responses.

2. Select a response, response group, or global response, and click OK.

   The response opens in the Rules group box, and is associated with the respective rule.

   Note: If the response you require does not exist, click New Response to create the response.

Enable and Disable Global Policies

The Administrative UI allows you to enable and disable global policies. By default, when you create a global policy, the policy is enabled. When a global policy is enabled, global rules contained in the global policy fire when users attempt to access the resources specified in the global rules.

If you disable a global policy, the rules contained in the policy do not fire.

To enable or disable a policy

1. Open the policy.
2. Select or clear the Enabled check box.

   If the check box is selected, the policy is enabled. If the check box is cleared, the policy is disabled. A disabled policy does not fire.

3. Click Submit.

   The policy is saved.

More information:

Start the Administrative UI (see page 50)

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 460)

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 456)

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: You can click Help for a description of fields, controls, and their respective requirements.

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 66)
Time Restrictions for Policies (see page 459)

Chapter 20: 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: SM_SERVERSESSIONSPEC

- 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 above situations. In all cases, passwords are not disclosed to let one user to impersonate another user.

The following terms will be used when describing impersonation:

Impersonated session

A user session created for the purpose of assuming another user's identity.

Impersonatee

The user whose identity can be assumed by a privileged user.

Impersonator

The privileged user that has the ability 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 know as user session. This is the time between authenticating and logging out.

Session Specification

Also know as the Session Ticket or Session Spec, it 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 "Session Management" chapter of the *Administration Guide*. For information about Web Agents, see the *Web Agent Configuration Guide*.

Impersonation Process

The process of impersonating another user consists of the following:

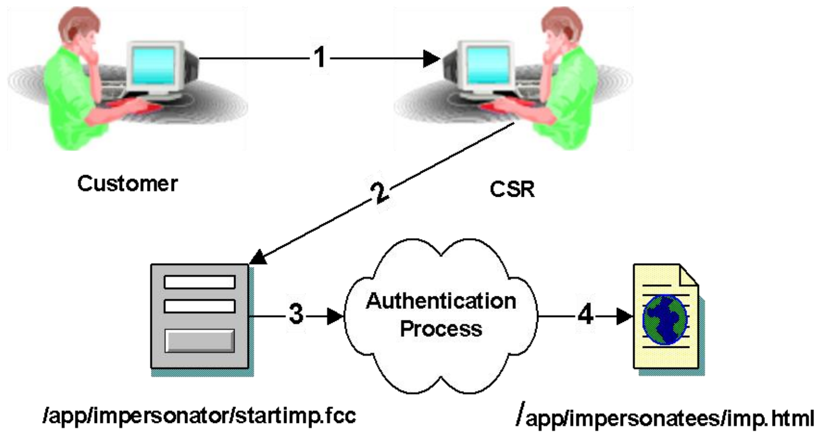
- A privileged user (impersonator) establishes a SiteMinder session by accessing a resource protected by SiteMinder. This can be the Administrative UI or some other application or resource protected by a SiteMinder policy.
- The impersonator, using the currently established identity, accesses a form 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.

Restricting Impersonation

For an impersonation to take place, the impersonator and impersonatee must be bound to policies in an impersonation realm. The impersonator policy must have a rule that fires for the ImpersonateStart event and the impersonatee policy must fire for the ImpersoanteStartUser event. The processes for configuring these objects are described in Policy Server Objects for Impersonation.

Impersonation Realms and Events

Impersonation originates in realm specifically configured to begin the Impersonation process. When an impersonator requests a resource in the impersonation realm, the impersonator is challenged by an impersonation authentication scheme. The scheme prompts the impersonator for credentials via a form. Instead of prompting for a username and a password as is usual, the form prompts the impersonator to supply the username of an impersonatee. Within the .fcc file for the form, there is logic that sets the password to the impersonator’s current Session Specification.

The Policy Server uses the impersonation authentication scheme to verify that the impersonator’s current session is valid, and then the Policy Server attempts to find the user to be impersonated in the directories included in the policy domain associated with the impersonation realm. If the Policy Server finds the user, authentication proceeds.

Once the Policy Server locates the impersonatee in a directory, it must verify that the impersonator has the right to impersonate, and that the potential impersonatee can be impersonated. These rights are configured and enforced using SiteMinder Policies and Rules, as well as two impersonation events.

Impersonation events are similar to authentication and authorization events, but are not passively invoked. Policies must specifically bind both the impersonator and the impersonatee to Impersonator event rules. If policies do not exist, or do not include impersonator event rules for the impersonator and for the impersonatee, impersonation will not be allowed.

Impersonation can be allowed or disallowed in any Realm using SiteMinder policies and rules with impersonation events. Realms can be configured to disallow all impersonation, or to restrict possible impersonators and impersonatees.

Policy Server Objects for Impersonation

In order to implement impersonation in an enterprise, a number of Policy Server objects must be configured. The combination of objects provides the authentication and policy entitlements that are required to enable one user to impersonate another. The following objects are required for impersonation:

Infrastructure Objects:

Agent

Impersonation requires a Web Agent and its associated Policy Server Agent object.

Note: To implement impersonation, you must have at least one SiteMinder Web Agent installed in your deployment. More information on installing a Web Agent exists in the *Web Agent Installation Guide*.

Authentication Scheme

Impersonation requires an authentication scheme object based on the Impersonation Template.

User Directory

Impersonation requires one or more user directory objects that point to user stores which contain impersonators and impersonatees. The two populations of users should be distinguishable due to an attribute value or group membership.

Domain

Impersonation requires a policy domain object that includes the user directory object(s) mentioned in the previous bullet.

Domain Objects:

Realms

Impersonation requires a minimum of two configured realm objects. One realm contains the resources accessible by the impersonatees. The other realm is the impersonation realm, and contains the resources and rules required to initialize an impersonation session.

Rules

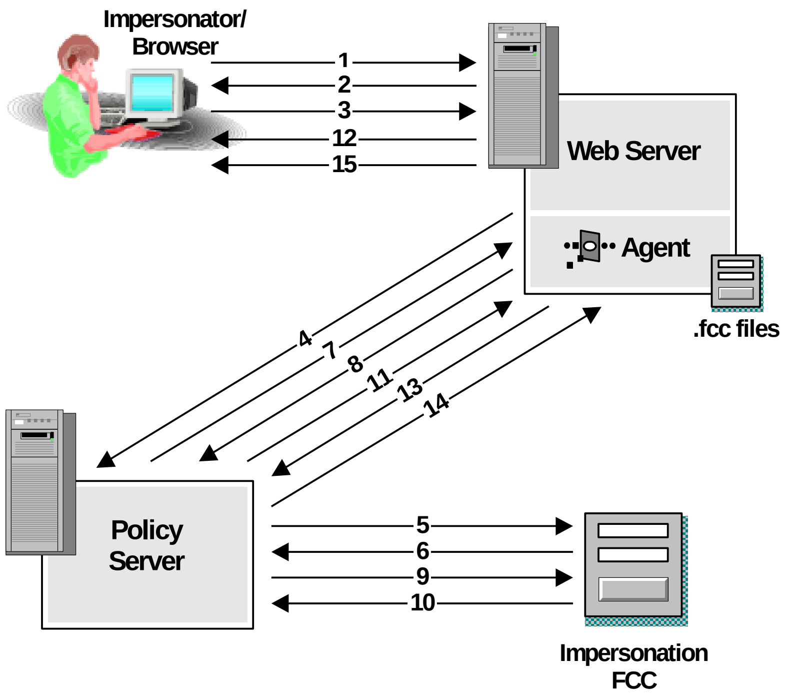
Impersonation requires access control rules to be in place. In addition, a rule with the ImpersonateStart event must exist for impersonators to begin an impersonation session. A rule with the ImpersonateStartUser event must exist in to allow a user to be impersonated.

Policies

Besides the policies that must be in place to protect a set of resources, impersonation requires additional policies to allow access to resources in the impersonation realm, to qualify users as impersonators, and to limit the set of impersonatees.

How an Impersonation Session is Initiated

The following figure is a detailed process flow for how a privileged user initiates an impersonation session.



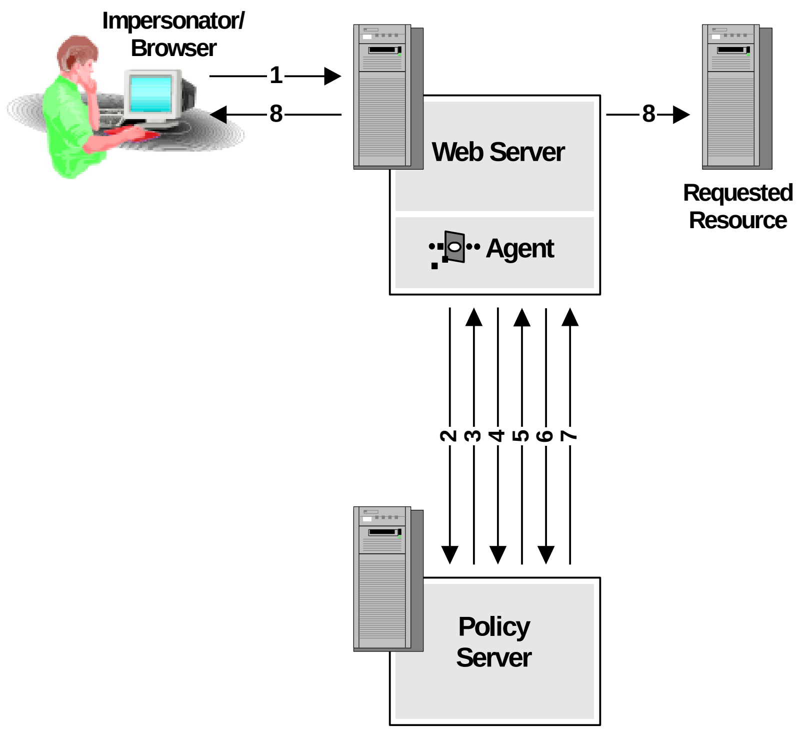
1. A previously authenticated impersonator attempts to impersonate a user by accessing an Impersonation .fcc file.
2. The .fcc is not protected, or the CSR is allowed access (decision not shown). The Web Agent presents the form (.fcc) that was requested in Step . The FCC has a hidden form field configured to indicate that the target is in a realm protected by the Impersonation authentication scheme.
3. The impersonator indicates the name of the user to be impersonated.
4. The Web Agent uses the target field of the .fcc to determine the realm.
5. The Policy Server determines the credentials required by the Impersonation authentication scheme.
6. The required credentials are returned.

7. The Policy Server indicates the realm that is being used to protect the .fcc target.
8. The Web Agent attempts to authenticate the impersonated user using the impersonator’s session spec as a password.
9. The Impersonation authentication scheme’s functionality is invoked to authenticate the user given the supplied credentials (the impersonatee name and the impersonator’s session spec). The impersonator’s session spec is validated. The Policy Server disambiguates the user and authenticates the impersonation session.
10. The Policy Server verifies that there is an event policy tied to the ImpersonateStart event. If there is a policy that applies to the impersonator, a similar check is performed to determine if there is am ImpersonateStartUser event bound to a policy that applies to the impersonatee. Responses can be tied to rules in either policy.
11. Once the impersonator has been authenticated using the impersonation authentication scheme, he can now attempt to impersonate the user in any application.

    Note: Since the impersonator has not accessed any resources, no impersonation has taken place.

    The session specification, is returned to the Web Agent. It contains the DN of the impersonatee and the DN of the impersonator, as well as the directories where both the impersonator and the impersonatee were located.

    The Web Agent moves the existing SMSESSION to SMSAVEDSESSION and sets a new SMSESSION cookie equal to the new session spec due to a @pushsession directive in the FCC.
12. The Web Agent attempts to determine if the impersonatee is authorized for the .fcc target resource.
13. The Policy Server indicates that the impersonatee is authorized for the .fcc target resource.
14. The impersonator accesses the resource that was indicated in the impersonation .fcc. For example, this can be a .jsp page that will indicate the DN of the impersonator and of the impersonatee.



1. The impersonator (now impersonating a user) navigates to a new application.
2. The Web Agent calls the Policy Server to determine if the resource is protected.
3. The impersonator access the requested resource.
4. Since the resource is protected and an SMSESSION cookie exists in the impersonator’s Web Browser for this cookie domain, the Web Agent calls the Policy Server to validate the session spec.
5. The session spec is validated according to current authentication logic except that additional checks are made against the impersonator and the user because the validation logic can determine that impersonation is occurring due to the contents of the session spec. The impersonator must be bound to an event policy in the current realm that has a rule for an ImpersonateStart event. The user must be bound to a similar policy for the ImpersonateStartUser event that applies to users that can be impersonated.
6. The Web Agent asks the Policy Server if the user is authorized for the resource.

7. The Policy Server responds with an authorization response. The authorization response can indicate that the access is allowed or that the access is denied for a multitude of reasons (no rule fired for this resource or access was explicitly forbidden for the user, auth level was too low, session timed out, etc). All of these access reject reasons mirror the user experience. However, Protection Levels are not enforced for impersonated sessions.
8. The requested resource is returned.

More information:

Enable Impersonation through an .fcc File (see page 577)

Configure an Agent to Use startimp.fcc

To invoke an impersonation session, an .fcc file, such as the sample startimp.fcc must be specified in the Impersonation authentication scheme and it must be processed by a Web Agent. By default, Web Agents do not protect files with the .fcc file extension.

One of the following two options is required in order for Web Agents to allow the initialization an Impersonation session:

- The Web Agent must be configured so that the .fcc extension is not ignored.
- The .fcc file used to invoke impersonation must use a different (not ignored) file extension, (for example, .ifcc).

Configure an Impersonation Authentication Scheme

Impersonation requires an Impersonation authentication scheme. This scheme is used as a method for an impersonator to initiate the impersonation process. To begin an impersonation session, an Impersonator directly accesses an .fcc file which is a resource protected by the Impersonation authentication scheme.

More information:

Impersonation Authentication Schemes (see page 327)

Enable Impersonation through an .fcc File

A critical part of impersonation is the authentication process using an appropriate authentication scheme and .fcc files. The .fcc files that enable the impersonation authentication scheme are files installed on a SiteMinder-enabled Web server. The files must end in the extension: .fcc. Files with this extension can take advantage of special processing by the Web Agent. The impersonation authentication scheme is similar to an HTML forms authentication scheme.

An impersonator can invoke the impersonation authentication scheme by directly accessing the .fcc file for the scheme. The proper authentication scheme is invoked through the .fcc file's target parameter. This value can be hard-coded in the .fcc file using the @target directive, or by setting the target (hidden form post variable) to the appropriate Web page.

Note: The target resource must reside in a realm that is protected by the impersonation authentication scheme.

More information:

HTML Forms Authentication Schemes (see page 261)

Basic .fcc Requirements for Impersonation

The FCC that begins the impersonation process must also include the @smpushsession=true directive, which instructs the Web Agent to save the current session cookie contents in another cookie so that the session cookie can now hold the session spec of the impersonated user.

The proper credentials must be presented to the Web Agent processing the impersonation authentication scheme in order for authentication to take place. The username should be the username of the user to be impersonated. The password should be set to the session spec of the impersonator, pre-pended, if necessary, with additional attributes. The FCC's facility for substituting the contents of cookies or headers into directives at the time the form is posted should be used for this purpose. Using this facility, the FCC sets the @password directive to the Users Session Specification along with other data if necessary. For more information on .fcc files, see HTML Forms Authentication Schemes (see page 261).

To end the impersonation process, another .fcc file can be included in the realm protected by the impersonation authentication scheme. This .fcc file should set the @target directive to point to a resource in the restricted realm that was used to begin the impersonation process. In addition, the @smredirect directive should be set to the same resource in order force an end for the authentication process. Finally, the @smpopsession=true directive should be used to restore the original session cookie.

FCC Directives for Impersonation

When constructing an .fcc file for impersonation, the following directives should be used in the file:

@logout

This directive logs the user out of SiteMinder and removes the SMSESSION cookie.

@smheaders

This directive adds HTTP request headers to the FCC namespace. For impersonation, this directive provides the contents of the session specification header, SMSERVERSESSIONSPEC (or SM_SERVERSESSIONSPEC; see Note about SMSERVERSESSIONSPEC and LegacyVariables), to the FCC namespace so that it is available for use as a password.

@smpushsession

This directive allows a user to “impersonate” another user and then return to the original session. This directive must be set to "true".

@smpopsession

This directive returns to the original session after @smpushsession has been used. This setting must be set to "true".

@smredirect

This directive redirects requests to the specified target.

@target

This directive tells the FCC where to redirect to after processing a URL.

@password

This directive specifies what the contents of the password to be passed to the Policy Server.

@smaltcreds

Allows custom authentication schemes to send credentials larger than 4KB.This may be used in the same manner that the @password directive is used.   When credentials are posted to an FCC using @smaltcreds, its value is sent to the Policy server during login as a byte buffer avoiding the password field which is restricted to 4k bytes.   The @smaltcreds directive may *not* be used with existing out-of-the box authentication schemes, but it may be used for custom authentication. Developers of custom authentication schemes must code their authentication scheme libraries to look for the @smaltcreds credentials in the lpszCertBinary field of the user credential struct passed through the Agent API during login.

@username

This directive specifies the username to be passed to the Policy Server.

% and $ replacement functionality

The "%" and "$$" functionality is used for data replacement similar to scalar variables in Perl. "%NAME%" is used to replace "NAME" with the data associated with "NAME" on a post. "$$NAME$$" is used to replace "NAME" with the data associated with "NAME" on a get.

Obtain the Session Specification using an FCC

The Web Agent includes the session specification in the set of headers that it includes with every request. The session specification header can be submitted as a password using a several FCC directives. The @smheaders directive is used to include the SMSERVERSESSIONSPEC header in the FCC’s namespace, and the @password directive is used to set the password to the contents of the SMSERVERSESSIONSPEC header.

The SM_SERVERSESSIONSPEC/SMSERVERSESSION header is only available if DisableSessionVars is set to its default value of false in the Web Agent configuration file.

Note: If the LegacyVariables Web Agent parameter is set to Yes, then the header should be SM_SERVERSESSIONSPEC. If the LegacyVariables Web Agent parameter is set to No, then the header should be SMSERVERSESSIONSPEC. The LegacyVariables parameter is not supported for Web Agents on IIS 6.0 web servers. For Web Agents on IIS 6.0, SMSERVERSESSIONSPEC is always the correct header.

Example .fcc Files for Impersonation

The following .fcc file is invoked directly by an impersonator, or called by the impersonation authentication scheme to begin the impersonation process:

```
@username=%USER%
@smheaders=%SMSERVERSESSIONSPEC%
@password=%SMSERVERSESSIONSPEC%
@smpushsession=true

<html>
<head><title>Sample Impersonation Form</title><head>
<body>
<h3> Please enter your Impersonation Information</h3>
<form method=post><table>
<tr>
  <td>User Name:</td>
  <td><input type=text name=USER></td>
 </tr>
 <input type=hidden name=target value="server.company.com/app/
     impersonatee/successimp.htm ">
<tr><td><input type=submit></td></tr>
</table></form></body>
</html>
```

The following .fcc file is invoked by impersonators to return to their original sessions.

```
@smpopsession=true
@target=/impersonators/end.htm
@smredirect=/impersonators/end.htm
```

Effects of FCCCompatMode

While traditional Web Agents set fcccompatmode to yes by default, framework Web Agents set fcccompatmode to no. When the mode is set to yes, the Web Agent can process impersonation requests. However, when the mode is set to no, the following two requirements must be met in order for the Web Agent to process impersonation requests:

- The Web Agent configuration parameter EncryptAgentName must be set to false.
- The Web Agent configuration parameter AgentName must be coded in the fcc file that initiates impersonation or in the query data that is sent to the Agent when an impersonation session is initiated.

Impersonation Event Configuration

Impersonation events are rule events that must be configured and included in policies in order to allow a privileged user to impersonate another user.

You must configure at least one of each of the following impersonation events in order to enable impersonation:

ImpersonateStart

When included in an appropriate policy, a rule that includes this event allows an impersonation session to begin.

ImpersonateStartUser

When included in an appropriate policy, a rule that includes this event allows a set of users to be impersonated.

More information:

Configure a Rule for Impersonation Event Actions (see page 401)

Configure Policies for Impersonation

In order for impersonation to function correctly, multiple policies must be configured.

For the resource that initiates impersonation:

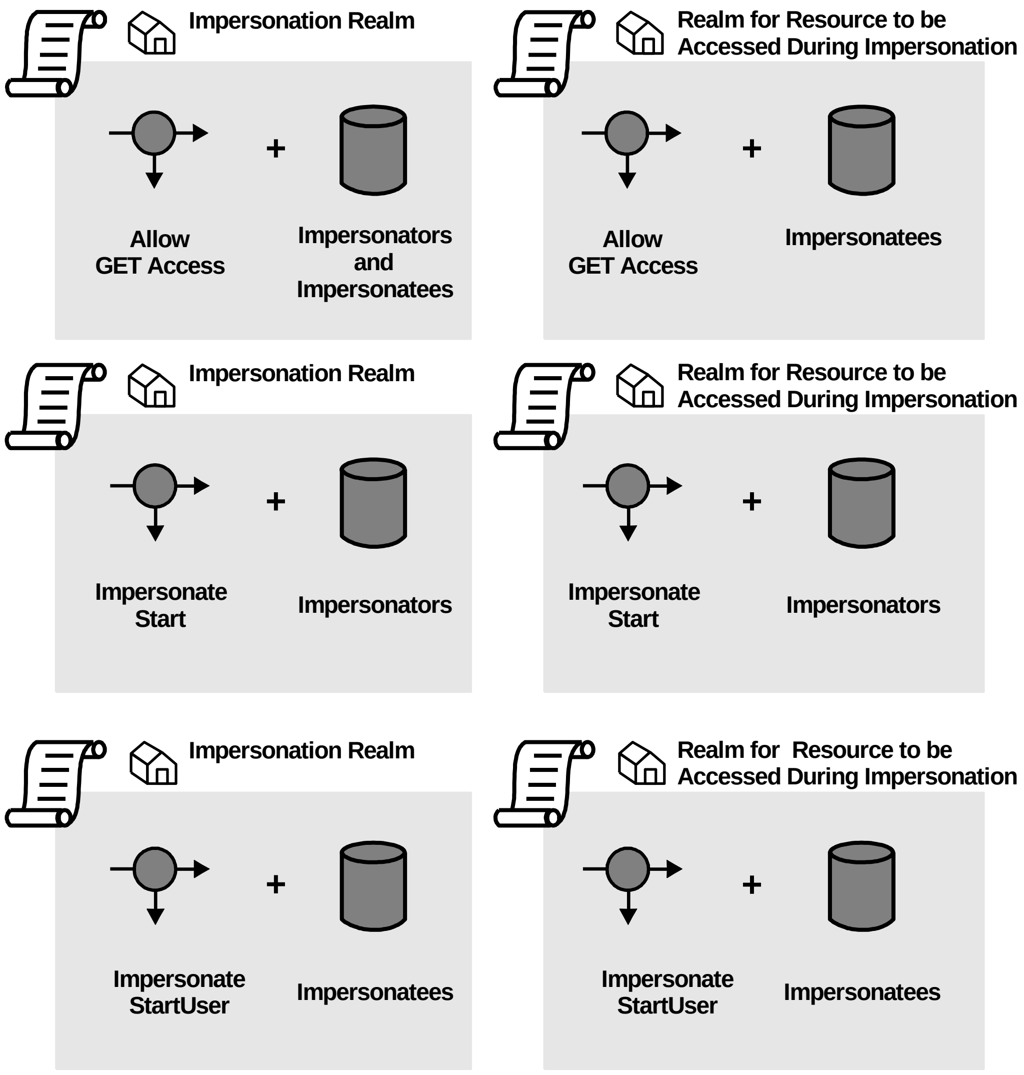
- Access control policy for both impersonators and impersonatees. For example, a policy that includes a GET access rule for both impersonators and impersonatees.
- Impersonation policy that includes an ImpersonationStart rule and impersonators.
- Impersonation policy that includes an ImpersonationStartUser rule and impersonatees.

For the resource to be accessed by the impersonator:

- Access control policy for impersonatees. For example, a policy that includes a GET access rule for impersonatees.
- Impersonation policy that includes an ImpersonationStart rule and impersonators.
- Impersonation policy that includes an ImpersonationStartUser rule and impersonatees.

Note: Policies must be in place for each resource that may be accessed by an impersonator.

The following figure shows the minimum policies required for an impersonator to access a resource:



More information:

Policies (see page 439)

Multiple Cookie Domain Support

If a site is composed of multiple cookie domains, an Impersonator’s identity could be confused while moving between resources in separate cookie domains. To avoid this problem the SMSESSION cookies in all of the cookie domains other than the current one must be cleared. This must be accomplished by modifying the forms that are used to begin and to end Impersonation. These forms should be augmented by HTML or script that will call server-side code to clear the SMSESSION cookies in all of the cookie domains other than the current one.

For example, consider Web Agents installed at yourcompany.com, subsidiaryA.com, and subsidiaryB.com. A Web Agent that carries out Impersonation could be located in the yourcompany.com domain. The .fcc files to start and end impersonation would need to call server-side functionality (JSP pages possibly) in subsidiaryA.com and subsidiaryB.com to clear out the SMSESSION cookies in those cookie domains for Impersonation to function correctly.

Sample Implementation of Impersonation

This section contains a description of a simple implementation of impersonation. The minimum Policy Server objects required to implement impersonation are:

Infrastructure Objects:

Agent

Impersonation requires a Web Agent and its associated Policy Server Agent object.

Note: To implement impersonation, you must have at least one SiteMinder Web Agent installed in your deployment. More information on installing a Web Agent exists in the *Web Agent Installation Guide*.

Authentication Scheme

An impersonation authentication scheme based on the Impersonation Authentication Scheme Template is required. For the sample defined in this section, the authentication scheme is named "Impersonation Auth".

User Directory

Impersonation requires one or more user directory objects that point to user stores which contain impersonators and impersonatees. The two populations of users should be distinguishable due to an attribute value or group membership.

Domain

A policy domain is required. For the sample defined in this section, the policy domain is named "Impersonation Domain".

Domain Objects:

Realms

For the sample described in this section, two realms are required: "Impersonation" and "App1". The "Impersonation" realm should use the "Impersonation Auth" authentication scheme. The “App1” realm can use any authentication scheme.

Rules

For the sample described in this section, you must configure a rule under the "Impersonation" realm that allows access to all resources for the "Get" action. In other words, an asterisk should be entered in the Resource field for the rule. You must also configure the rules for the impersonation events. One rule allows impersonation if the impersonator is included in an applicable policy, and the other rule allows an impersonatee to be impersonated if included in a different, applicable policy.

Rules

A similar set of rules to those in the “Impersonation” Realm should be created under the “App1” realm.

Policies

For the sample described in this section, six policies are needed. One policy must be defined for each rule in the "Impersonation" realm, and one policy must be defined for each rule in the "App1" realm.

Sample Impersonation Implementation Assessment

Once all of the required Policy Server objects are in place, an administrator initiates an impersonation session by doing the following:

1. The administrator who will become the impersonator logs into a SiteMinder protected network.
2. The administrator provides credentials and is authenticated and authorized by SiteMinder.
3. The administrator accesses the imp.fcc file using a Web browser.
4. The administrator is prompted to enter a user ID for the person to be impersonated. The administrator may also be required to provide additional information about the user to be impersonated.
5. The administrator submits the information.

6. The Policy Server uses the policies defined for impersonation to determine the following:
   - Is the administrator allowed to act as an impersonator?
   - Is the user allowed to be impersonated?
7. If both are true, the impersonator impersonates the impersonatee.

   Note: Without custom development (.jsp pages, servlets, etc.) the impersonation session can be tracked using the Policy Server's audit logging. However, it may be beneficial for an enterprise to create some custom Web applications to monitor and track impersonation sessions.

Sample Forms for Impersonation

Impersonation requires a number of Web-based forms that must be customized for an appropriate interface for an enterprise. This section outlines the basic requirements of these forms and makes several suggestions about the contents of the forms.

Forms for Initiating the Impersonation Process

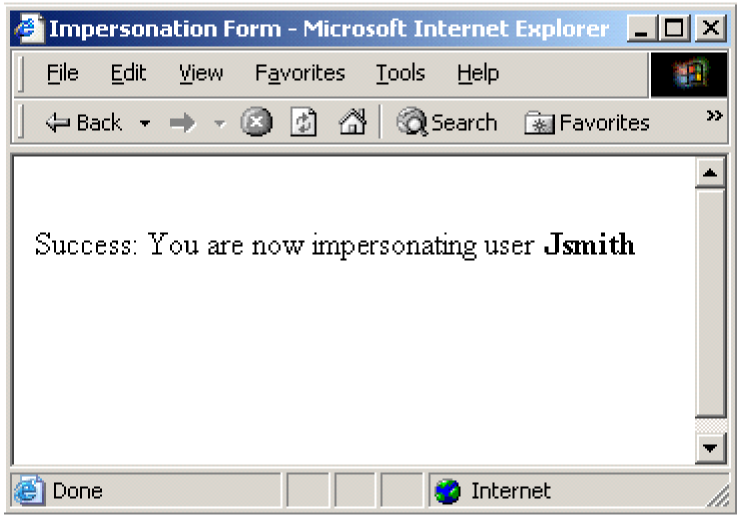
Forms contain the basic elements that should be presented by the .fcc file that invokes the impersonation process.

The hint field indicates that it is possible to configure impersonation to require additional attributes which can be used for disambiguating an impersonatee. The hint is configurable in the impersonation authentication scheme as a list of additional attributes, which can include information such as an employee number, or any other user attribute that uniquely defines a user in the user directory.

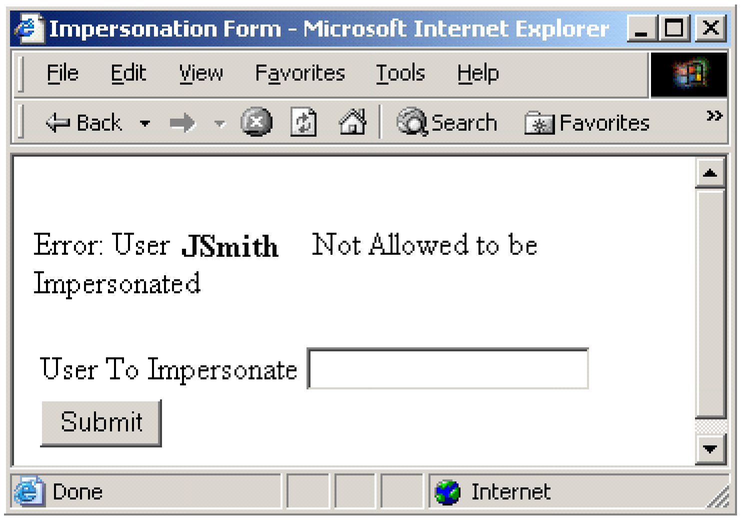
Impersonation Result Forms

The results of an impersonation attempt must be communicated in a form to represent both successful and failed impersonation attempts.

Note: The following forms are provided to illustrate the basic requirements for each form. All impersonation forms must be customized and referenced from the .fcc file.

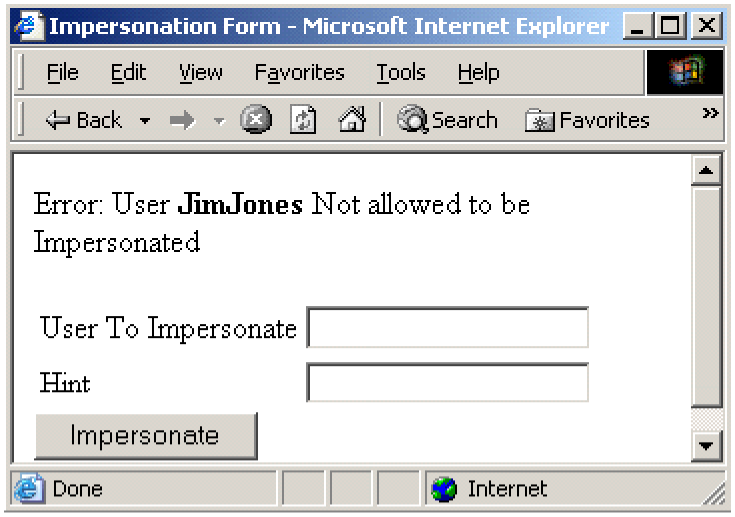

Impersonation Form - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Done
Internet

Success: You are now impersonating user **Jsmith**


Impersonation Form - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Submit
Done
Internet

Error: User **JSmith** Not Allowed to be Impersonated

User To Impersonate


Impersonation Form - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Impersonate
Done
Internet

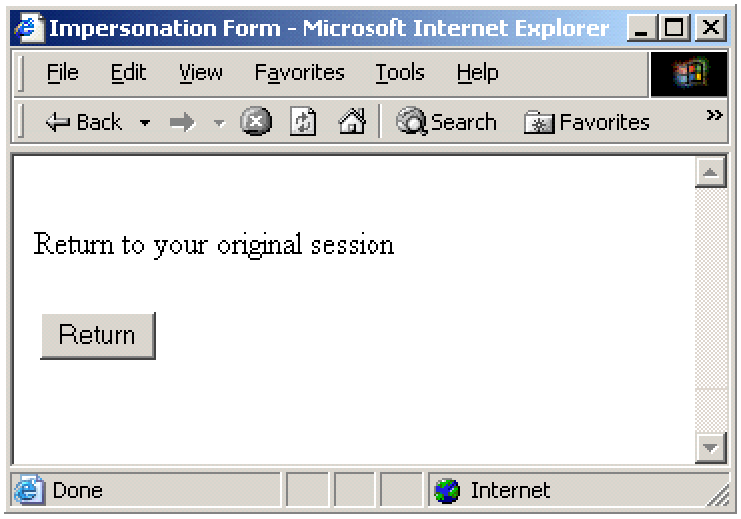
Error: User **JimJones** Not allowed to be Impersonated

User To Impersonate	
Hint	

Impersonation Logoff Forms

A form for ending an impersonation session must be configured.

Note: The following form is provided to illustrate its basic requirements. Impersonation forms must be customized and referenced from the .fcc file.


Impersonation Form - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Return
Done
Internet

Return to your original session

Chapter 21: Password Policies

This section contains the following topics:

Password Services Overview

Password Services provide an additional layer of security to protected resources by allowing you to manage user passwords in LDAP user directories or relational databases. To manage user passwords, you create password policies that define rules and restrictions governing password expiration, composition, and usage.

In addition to managing passwords, Password Services enables users to select their own passwords:

- At the times you specify, such as when a password expires
- When users feel it is necessary to change their passwords
- When new users register using Registration Services

The password policy ensures that the user will select a valid password without additional administrative involvement.

When Password Services are active, SiteMinder invokes a password policy whenever a user attempts to access a protected resource. Password Services then evaluates the user's credentials. If the user's password has expired based on criteria defined in the password policy, the user's account can be disabled to prevent unauthorized access to the resource or the user can be forced to change his password. If disabled, the user's account must be re-activated by an administrator.

Password policies can be associated with an entire user directory or database, or a subset of the directory or database, called a namespace. Multiple password policies can be configured for the same user directory or namespace, in which case they are applied according to priorities you can specify for them.

Password Services provides two mechanisms for implementing password services:

- CGI-based

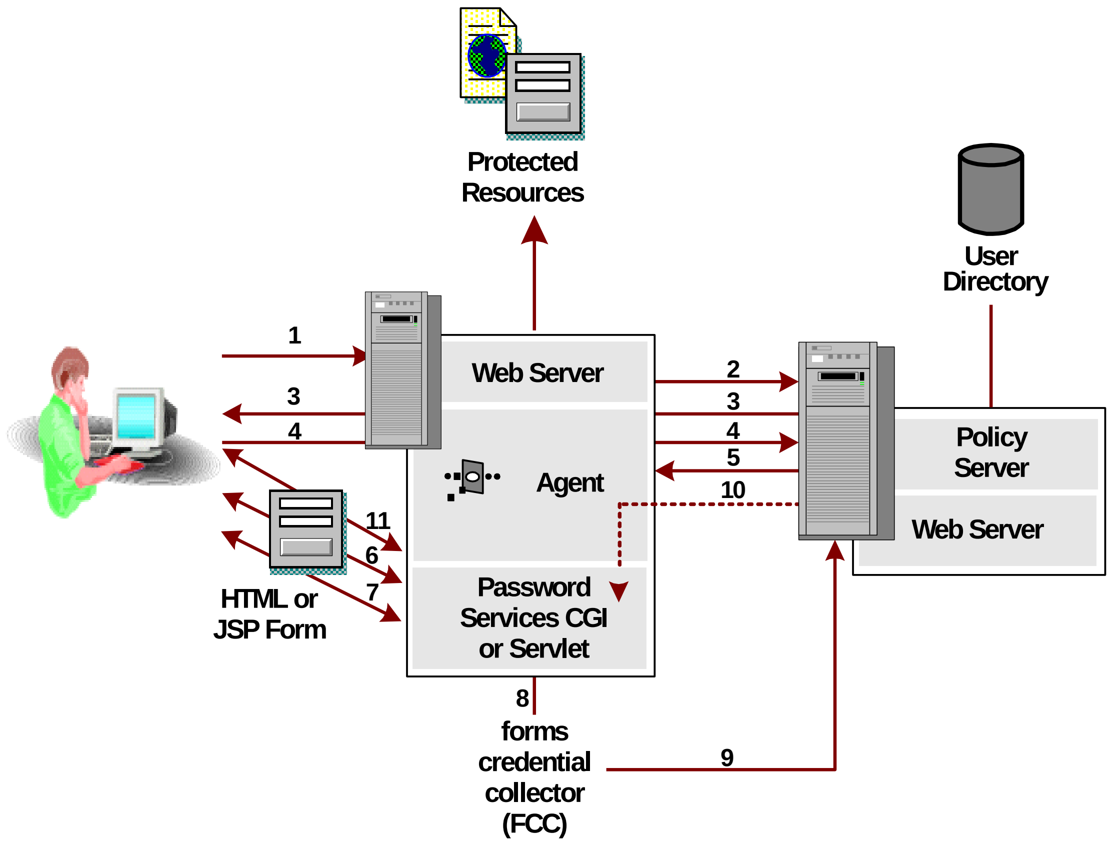
  Password Services CGI with customizable HTML forms (the default mechanism).

- Servlet-based

  Password Services servlet with customizable JSP forms.

How Password Services Work

The following illustration depicts an example of how Password Services work in a SiteMinder environment configured to protect web resources. In the example, a user's password has expired and must be changed.



When a user attempts to access a protected resource:

1. The user attempts to access a Web page by sending a request to the Web Server.
2. The SiteMinder Web Agent intercepts the request and checks with the Policy Server to see if the requested resource is protected.
3. If the resource is protected, the Policy Server requests user credentials from the browser.
4. The user sends credentials to the Policy Server.
5. The Policy Server authenticates the user.

   Once the user is authenticated, the Policy Server checks to see if user information is stored in a directory or database associated with a password policy. If it is, the Policy Server makes sure that the user's password is valid based on the password policy criteria.
6. If the password has expired, the Policy Server sends a redirection URL pointing to the password services servlet to the Agent.

   The Agent redirects the browser to request the password services servlet using the redirect URL.
7. The password services CGI or servlet determines which HTML/JSP (as appropriate) form to present to the user from a set of HTML/JSP templates and displays that form in the browser.

   The form displays a message explaining why the user has been redirected. It prompts users to enter their old password and new password, then confirm the new password by re-entering it.
8. Once the user has completed the form, the Password ServicesCGI or servlet passes the information received from the Agent and the encrypted information the user entered to the Forms Credential Collector (FCC).
9. The FCC passes the information received from the Password ServicesCGI or servlet to the Policy Server.

   The Policy Server checks the new password against the password policy to ensure that it is valid, then changes the password.
10. The Policy Server again sends a request to the Agent to redirect the browser to the Password Services CGI/servlet.
11. The Password Services servlet displays a message informing the user that the password has been successfully changed.

    Once the user has read the message, she is redirected to the page she originally requested.

Apache Web Server Prerequisites

Depending on the operating system on which your environment is running, you must manually edit the Apache httpd.conf configuration file to ensure that an implementation of CGI-based Password Services works.

To modify the httpd.conf configuration file for Solaris, add the PassEnv line after the ServerRoot line.

Example:

```
ServerRoot "/export/home/smuser/apache/"
PassEnv LD_LIBRARY_PATH "/export/home/smuser/netegrity/webagent/bin"
```

To modify the httpd.conf configuration file for Windows, add the following lines:

```
# To use CGI scripts:
AddHandler cgi-script .exe
AddHandler smformsauth-handler .fcc
AddHandler smcookieprovider-handler .ccc
```

Password Services Considerations

When setting up Password Services, it is important that you consider the following:

- Time Differential Effect on Password Services on Multiple Policy Servers

  If you use Password Services with multiple Policy Servers, be sure the time settings on the Policy Server systems are synchronized, to prevent inadvertent disabling of user accounts or pre-mature forced password changes.

- Working With Directory Server Password Policies

  Directory Servers such as Netscape LDAP Directory Server allow you to create simple password policies which are evaluated by the directory before SiteMinder policies. If the Directory Server policies are more restrictive than SiteMinder policies, the Directory Server will accept or reject passwords without notifying SiteMinder. SiteMinder will not know that a user is attempting to login and will not provide any of the password management features.

Important! To use Password Policies, you must disable any Directory Server policies or make them less restrictive than the Password Policies.

Custom Password Services Directory Considerations

Setting up a custom Password Services directory on a non-default web server requires you to:

1. Copy the smpwservicescgi.exe to a location accessible by the non-default web server.

   The CGI extracts an agent_path from the Redirect URL to replace the default /siteminderagent/pw path.

2. Create two virtual directories: pw and pwcgi.

   The latter is a CGI directory. For example:

   - <custom directory>/pwcgi
   - <custom directory>/pw

When a user is redirected to the Password Services CGI or servlet, it takes the information from the Policy Server, determines why the password is invalid, and displays a form that provides information or requests additional credentials from the user.

Make sure that the Password Services CGI or servlet is not protected. If SiteMinder is protecting directories above the servlet, create a realm that specifies the following:

- Resource Filter if using CGI: siteminderagent/pwcgi/smpwservicescgi.exe
- Resource Filter if using Servlet: siteminderagent/pwservlet/PSWDChangeServlet
- Default Resource Protection: Unprotected

Do not create a policy for this realm.

Note: If a user who is accessing resources through an Agent that is not using an SSL connection must change passwords, the user's new password information will be received over the non-secure connection. To provide a secure change of passwords, set up a password policy that redirects the user over an SSL connection using the Redirection URL field.

Change the Default Redirect URL

By default, a SiteMinder Web Agent installation includes the following:

/*web_agent_install_dir*/pw/smpwservicescgi.exe

When a user cannot be properly disambiguated by the authentication server, the user is redirected to this location.

To customize the redirection destination, set the NETE_PWSERVICES_REDIRECT environment variable to the relative URL of the redirect destination, and restart the Web server associated with the Web Agent.

Create Password Policies

You create a password policy to provide an additional layer of security to protected resources.

To create a password policy object

1. Click Policies, Password.
2. Click Password Policy, Create Password Policy.

   The Create Password Policy pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
3. Enter the policy name.
4. Select the user directory to which the policy applies from the Directory list.
5. Specify if the policy applies to the entire directory or part of the directory.

   Note: If the policy is to only apply to part of the directory, click Lookup to specify the part of the directory that the policy is to apply.
6. Specify the redirection URL in the Redirection URL field:
   a. If you want to use the Password Services CGI, add the following to the default path:

      http://<*server.company.org*>

      The completed entry should be:

      http://<*server.company.org*>/siteminderagent/pwcgi/smpwservicescgi.exe

      Note: The default Password Service CGI path may differ if you have set up a custom Password Services directory.
   b. If you want to use the Password Services servlet, specify the following:

      http://<*server.company.org*>/siteminderagent/pwservlet/PSWDChangeServlet
7. 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: You can click Help for a description of fields, controls, and their respective requirements.

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: You can click Help for a description of fields, controls, and their respective requirements.

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: You can click Help for a description of fields, controls, and their respective requirements.

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: You can click Help for a description of fields, controls, and their respective requirements.

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: You can click Help for a description of fields, controls, and their respective requirements.

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.

Add a Change Password Link to the Password Services CGI

The link to the Password Change Request form is:

```
<a href="/siteminderagent/pwcgi/smpwservicescgi.exe?SMAUTHREASON=
34&TARGET=<URL_of_Protected_Web_Page_After_Password_Change>">Change Password</font></a>
```

where <URL_of_Protected_Web_Page_After_Password_Change> is the URL of a protected Web page to which users are directed after they change their password. For example, http://server.myorg.org/AfterPasswordChange.html.

Note: Adding this link to an unprotected page will allow all users to change their own passwords. The URL to which users are directed after they change their password must be protected, otherwise the password change will fail.

In addition, SMAGENTNAME needs to specific if you set the FCCCompatMode = NO.

For example, if you have FCCCompatMode set to "NO", you need to specify the SMAGENTNAME into your link, such as:

Change Password Link is:
"<ahref="/siteminderagent/pwcgi/smpwservicescgi.exe?SMAUTHREASON=
34&SMAGENTNAME=$$SMAGENTNAME$$&TARGET=$TARGET$">Change Password</font></a>"

Allow Specific Users to Change Their Passwords in CGI

If you want to allow only certain users to change their own passwords, complete the following procedure.

To allow specific users to change their passwords

1. Modify the permissions for the PWLogin.template file:
    a. Navigate to the following location:

       web_agent_installation_dir/pw

       where *web_agent_installation_dir* is the installed location of the Web Agent.
    b. In File, Properties, deselect the read-only attribute.
2. Edit the text in the PWLogin.template file:
    a. Open PWLogin.template in a text editor.
    b. Add the following line to the template at an appropriate location:
    <a href="/siteminderagent/pwcgi/smpwservicescgi.exe?
    SMAUTHREASON=34&TARGET=$$TARGET$$">Change Password</a>
    c. Save the file.
3. Access the Administrative UI.
4. Create an Authentication Scheme with the following settings:
    - Server Name: The name of the server where the Password Services CGI resides. For example:

      myserver.mycompany.org
    - Target: /siteminderagent/pwcgi/smpwservicescgi.exe
5. Optionally, create a new policy domain.

   The policy domain should include the user directory that contains the users that are allowed to change their own passwords.

   If you do not create a new policy domain, select an existing policy domain.

6. Create a realm that specifies the directory that you are protecting. In the Authentication Scheme list box, select the authentication scheme you created in step 4.
7. Create a rule under the realm that specifies the resource(s) that you are protecting.

   Note: If you create a rule that specifies all of the resources (*) in the directory that you are protecting, you do not have to create separate rules for localized Password Services.
8. Create a policy that binds the rule you created and the users/groups who are allowed to change their passwords.

More information:

Create and Use a Localized Password Services Properties Files (see page 609)

Add a Change Password Link to the Password Services Servlet

The link to the Password Change Request form is:

```
<a href="/siteminderagent/pwservlet/PSWDChangeServlet?
SMAUTHREASON= 34&TARGET=<URL_of_Web_Page_After_Password_Change>">
Change Password</font></a>
```

where *URL_of_Web_Page_After_Password_Change* is the URL of the Web page where users are directed after they change their password. For example, http://server.myorg.org/AfterPasswordChange.html.

Adding this link to an unprotected page will allow all users to change their own passwords.

Allow Specific Users to Change Their Passwords in Servlet

If you want to allow only certain users to change their own passwords, complete the following procedure.

To allow specific users to change their passwords

1. Modify the permissions for the PWLogin.jsp file:
   a. Navigate to the following location:

      web_agent_installation_dir/pwservlet/default

      where *web_agent_installation_dir* is the installed location of the Web Agent.
   b. In File, Properties, deselect the read-only attribute.

2. Edit the text in the PWLogin.jsp file:
   a. Open PWLogin.jsp in a text editor.
   b. Add the following:
      ```
      <p align="center"><font size="1">
      <a href="/siteminderagent/pwservlet/
      PSWDChangeServlet?SMAUTHREASON=34&TARGET
      =<%TARGET%>&SMAGENTNAME=<%=agentname%>">Change Password</font></a>
      ```
   c. Save the file.
3. Access the Administrative UI.
4. Create an Authentication Scheme with the following settings:
   - Server Name: The name of the server where the Password Services servlet resides. For example:

     myserver.mycompany.org
   - Target: /siteminderagent/pwservlet/PSWDChangeServlet
5. Optionally, create a new policy domain.

   The policy domain should include the user directory that contains the users that are allowed to change their own passwords.

   If you do not create a new policy domain, select an existing policy domain.
6. Create a realm that specifies the directory that you are protecting. In the Authentication Scheme list box, select the authentication scheme you created in step 4.
7. Create a rule under the realm that specifies the resource(s) that you are protecting.

   If you create a rule that specifies all of the resources (*) in the directory that you are protecting, you do not have to create separate rules for localized Password Services.
8. Create a policy that binds the rule you created and the users/groups who are allowed to change their passwords.

More information:

Localize Servlet-based Password Services (see page 613)

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's 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\Authentication\DisallowUsernameInURL

2. Set the DWORD value to one of the following:

   - 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\Authentication\DisallowUsernameInURL
4. Set the DWORD value to one of the following:
   - 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.

Note: If you choose to modify the registry setting so that the login ID is not appended to password services redirects, password services automatically uses the templates with the PWnn prefix. These templates are described in CGI-based Password Services HTML Form Templates (see page 605).

More information:

How Password Services Work (see page 590)

CGI-based Password Services HTML Form Customization

The Password Services CGI uses HTML forms to allow users to change their passwords. You can customize these forms to create a branded look and modify the information provided on the forms.

CGI-based Password Services HTML Form Templates

The Password Services CGI uses HTML forms to collect password information from users and send it to the Policy Server. The Policy Server checks the information against the password policy and then sends a message back to the Password Services using another form.

A SiteMinder Web Agent installation provides form templates that you can customize. The sample forms are installed in the following locations:

- Windows: <installdir>\SiteMinder Web Agent\pw
- UNIX: <installdir>/siteminder/webagent/pw

The templates include the following:

Template	Description
PWAccountInactive.template	Informs users that their account has been disabled due to inactivity.

Template	Description
PWnnAccountInactive.template	Identical to PWAccountInactive.template, except that the login ID is not appended to password services redirects.
PWChange.template	Requires users to change their password before accessing the protected resource. Users must enter their old password, new password, then confirm the new password.
PWnnChange.template	Identical to PWChange.template, except that the login ID is not appended to password services redirects.
PWChangeAccept.template	Informs users that their password has been changed successfully.
PWChangeErr.template	Informs users that their password has not been changed because of an internal problem, such as a server overload, which caused the password change request to fail.
PWnnChangeErr.template	Identical to PWChangeErr.template, except that the login ID is not appended to password services redirects.
PWExcessLogin.template	Informs users that they have exceeded the permissible number of login attempts. Users must wait a specified amount of time before attempting to log in again or contact the administrator to reactivate their password.
PWnnExcessLogin.template	Identical to PWExcessLogin.template, except that the login ID is not appended to password services redirects.
PWChangeMismatch.template	Informs users that the password they selected is not valid and prompts them to select a new one.
PWnnChangeMismatch.template	Identical to PWChangeMismatch.template, except that the login ID is not appended to password services redirects.

Template	Description
PWChangeOption.template	Allows users whose passwords are about to expire to change their passwords now or be reminded later. Users can enter their old password, new password, and confirm a new password.
PWnnChangeOption.template	Identical to PWChangeOption.template, except that the login ID is not appended to password services redirects.
PWExpired.template	Informs users that their password has expired.
PWnnExpired.template	Identical to PWExpired.template, except that the login ID is not appended to password services redirects.
PWAccountDisabled.template	Informs users to contact an Administrator because their account has been disabled.
PWnnAccountDisabled.template	Identical to PWAccountDisabled.template, except that the login ID is not appended to password services redirects.
PWSelfChangeLogin.template	Allows users to change their passwords from a login page.

More information:

Localize CGI-based Password Services (see page 609)
Remove the Login ID When Redirecting for Password Services (see page 604)

Modify CGI-based Password Services HTML Templates

You can customize the layout and formatting of Password Services HTML templates just as you would customize any other HTML document.

However, when you customize a Password Services template be careful *not* to alter:

- JavaScript functions at the beginning of the template file.
- <form> HTML tags. Password Services CGI uses the <form> output to pass information to the appropriate scripts for processing.
- Text that uses the $*text*$ notation. The Password Services CGI uses the information held in variables denoted by $ symbols.

Changes to any of these elements may cause Password Services to malfunction.

Password CGI-based Services Properties Files

Password Services properties files determine the text displayed on the HTML pages used to collect information from users and display status messages. This text includes:

- Field names
- Dialog titles
- Status messages and instructions for users
- Links and button text

To generate HTML pages, the Password Services CGI uses the Password Services templates. When the CGI encounters a parameter in the templates, it searches for a corresponding identifier defined in a Properties file. The Properties files pairs the identifier with a text string, which is added to the generated HTML page.

The following is a sample of the contents of a Password Services Properties file:

```
szPWNewPWChange = New Password
```

You can modify the text string to the right of the equal sign (=) to change the label that appears in the generated HTML pages; however, you should not modify the identifier on the left.

Note: Password Services requires that all characters in the properties files are UTF-8 encoded. Any characters that are not UTF-8 encoded (such as umlauts or tildes in the Windows or the 8-bit ASCII character sets) will not be processed correctly.

Password Services installs Properties files for the following languages:

- English (US-EN): PasswordServicesUS-EN.properties
- French (FR-FR): PasswordServicesFR-FR.properties
- Japanese (JP-JP): PasswordServicesJP-JP.properties

These files are located in the following directory:

- Windows:*web_agent_installation*\pw
- UNIX: *web_agent_installation*/pw

where *web_agent_installation*> is the installed location of the SiteMinder Web Agent.

Localize CGI-based Password Services

The Password Services template files provide Unicode support for multiple languages. The default encoding is UTF-8, which enables users to enter a password in any UTF-8 supported language.

To localize the Password Services templates for French or Japanese, modify the sample French and Japanese Properties files described in Password CGI-based Services Properties Files (see page 608).

To configure your site to use localized Password Services, see Overview of Servlet-based Password Services JSP Forms (see page 610).

Create and Use a Localized Password Services Properties Files

Prior to 5.x QMR 2, the Web Agent passed on the value of the SMLOCALE variable so that the proper language encoding was set for Password Services change forms. This required you to append the variable SMLOCALE=<*country*>-<*language*> to the Target URL for the protected resource—for example:

http://server.mysite.org/dir1/targetpage.html?SMLOCALE=FR-FR

Beginning at 5.x QMR 2, the language encoding is read from the Web browser so no manual intervention is required. The Password Services CGI reads the ACCEPT_LANGUAGE variable from the Web browser to determine the language in which to display the password change forms.

For each language, the change forms included with the SiteMinder Web Agent use a different properties file. The properties file defines certain aspects of the text displayed on the HTML pages.

The following properties files are automatically installed with the SiteMinder Web Agent:

- English: PasswordServicesUS-EN.properties
- French: PasswordServicesFR-FR.properties
- Japanese: PasswordServicesJP-JP.properties

If the ACCEPT_LANGUAGE indicates a language for which no properties file is present the English properties file is used.

To create and use a properties file for another language

1. Translate one of the properties files provided by SiteMinder into the language of your choice.

   For example, copy the PasswordServicesUS-EN.properties file and translate anything to the right of the equals sign (=).

2. Rename the file according to the convention for Password Services, which is:

   PasswordServices<*country*>-<*language*>.properties

   Note: RFC 3066 defines the language tags that you should use as part of the naming convention (http://www.ietf.org/rfc/rfc3066.txt).

   Note: Be sure the browser is set for the correct language.

   If you set multiple languages for the browser, Password Services will only use the first language in the list. If the Password Services CGI cannot find a properties file for that language, the English properties file is used. The other languages specified for the browser are ignored.

Servlet-based Password Services JSP Form Customization

The Password Services Servlet uses *JavaServer Page* (*JSP*) forms that allow users to change their passwords. You can customize these forms to create a branded look and modify the information provided on the forms.

Overview of Servlet-based Password Services JSP Forms

The Password Services servlet uses JSP forms to collect password information from users and send it to the Policy Server. The Policy Server checks the information against the password policy and then sends a message back to the Password Services servlet using another form.

The SiteMinder Web Agent installation provides form templates that you can customize. The sample JSP forms are installed in the following locations:

- Windows: <*Web-Agent-Install-Dir*>\jpw
- UNIX: <*Web-Agent-Install-Dir*>/jpw

Where <Web-Agent-Install-Dir> is the SiteMinder Web Agent installation directory.

The templates include the following:

Template	Description
PWAccountInactive.jsp	Informs users that their account has been disabled due to inactivity.
PWChange.jsp	Requires users to change their password before accessing the protected resource. Users must enter their old password, new password, then confirm the new password.
PWChangeAccept.jsp	Informs users that their password has been changed successfully.
PWChangeErr.jsp	Informs users that their password has not been changed because of an internal problem, such as a server overload, which caused the password change request to fail.
PWExcessLogin.jsp	Informs users that they have exceeded the permissible number of login attempts. Users must wait a specified amount of time before attempting to log in again or contact the administrator to reactivate their password.
PWChangeMismatch.jsp	Informs users that the password they selected is not valid and prompts them to select a new one.
PWChangeOption.jsp	Allows users whose passwords are about to expire to change their passwords now or be reminded later. Users can enter their old password, new password, and confirm a new password.
PWExpired.jsp	Informs users that their password has expired.
PWAccountDisabled.jsp	Informs users to contact an Administrator because their account has been disabled.
PWSelfChangeLogin.jsp	Allows users to change their passwords from a login page.

Template	Description
PWLogin.jsp	Allows a user to enter credentials and login. This .jsp file generates the form shown to the user for login if the redirect path in the HTML Forms Based authentication scheme to the servlet (i.e. /siteminderagent/pwservlet/ PSWDChangeServlet).
UserMsg.jsp	UserMsg.jsp returns the messages depending on the reason of user rejection.
PWExpiredLogin.jsp	When a user’s session expires, this .jsp displays a login screen to allow a user to establish a new session.
PSWDChangeError.jsp	This jsp informs users that their password has not been changed because of an internal problem, which caused the password change request to fail.
PWAuthLevelLogin.jsp	If a user with an established session attempts to access a resource that is protected at a higher level than the original resource requested at login, this .jsp presents a form that allows the user to authenticate again, gaining access to the more secure resource.
Resource.jsp	Specifies the content type as UTF-8 and indicates the resource bundle that contains localized strings for all password services forms.

More information:

Modify Servlet-based Password Services JSP Forms (see page 612)

Modify Servlet-based Password Services JSP Forms

You can customize the layout and formatting of Password Services JSP forms just as you would customize any other HTML document.

Do *not* alter the following:

- JavaScript functions at the beginning of the template file.
- <form> HTML tags

  The Policy Server uses <form> output to pass information to the Password Services servlet for processing
- Parameters enclosed in <% %> notation

  These <%*text*%> parameters are used by the Password Services servlet. Although you cannot change the existing ones, you can add your own <%*text*%> parameters or use Password Services-specific <%*text*%> parameters for your own purposes, if desired.

  Different <%*text*%> parameters are used in different JSP template files.

Localize Servlet-based Password Services

Servlet-based password services uses a single set of template .jsp files that are populated at runtime using the strings/messages from an externalized set of properties files. The language used by the .jsp files is determined by the locale setting of a users' browser.

If a browser is configured for multiple locales, the specified locales are compared, in order, and the first locale that matches a supported language for password services forms determines the language used for the forms displayed to users.

Note: If no language preference is specified in a user's browser, the default properties file, PasswordServices.properties will be used.

Modify Strings Displayed in Forms

In the .jsp files, the strings and messages represented by a tag in the following format:

```
<%=getI18NParameter (resourcebundle, TitleName)%>
```

For example, in the PWChange.jsp, the title "SiteMinder Password Services" is now replaced with the following tag:

```
<%=getI18Nparameter (resourcebundle, “PWChange.Title”)%>
```

where resourcebundle is the object that contains the name-value pair strings that are loaded from the properties file and "PWChange.Title" is the key whose value will be retrieved from the properties file.

When a tag occurs in the .jsp file, it is replaced with the value of the corresponding identifier defined in a properties file.

The following is a sample of the format for contents of a Password Services properties file:

```
PWChange.Title = SiteMinder Password Services
```

The text string to the right of the equal sign can be modified to change the label that appears in forms displayed to users.

Add New Strings

To add a new string or message to a form, you must add a new tag to the .jsp file and corresponding entry in the associated properties file. For example, to add a new tag called NewTag to a .jsp file called PWScreen.jsp, you must do the following:

1. Add a new property to the properties file:
   ```
   PWScreen.NewTag=NewValue
   ```
2. Add a new tag to the .jsp file:
   ```
   <%=getI18NParameter (resourcebundle, “PWScreen.NewTag”)%>
   ```

At runtime, the tag in the .jsp file is replaced with the string: NewValue.

Properties Files

SiteMinder includes the following properties files by default:

PasswordServices.properties

The default properties file, this file contains the key-value pairs for English (EN).

PasswordServicesJA.properties

This file contains the key-value pairs for Japanese (JA).

All characters in the properties files are UTF-8 encoded. If you create your own properties files or add to the existing files, the files must be UTF-8 encoded.

Properties must be named according to one the following conventions:

- PasswordServices<*LANGUAGE*>-<*COUNTRY*>.properties

  or
- PasswordServices<*LANGUAGE*>.properties

where <*LANGUAGE*> is the two-letter code for a language (for example, EN for English, FR for French, JA for Japanese), and <*COUNTRY*> is the two letter code for country (for example, US for United States, JP for Japan). For example, the properties file for French in Canada would be named PasswordServicesFR-CA.properties.

Support a New Language

Using properties files, you can add additional language support.

To support a new language

1. Create a UTF-8 encoded properties file.
2. Name it in the required format.
3. Place it in the <*Web-Agent-Install-Dir*>\jpw directory, along with the .jsp files.

Set a Language Other Than English as the Default Language

If no language preference is specified in a user’s browser, the default properties file, PasswordServices.properties will be used.

To set a language other than English as the default language

1. Rename the following file to some other name:
   web_agent_install_dir\jpw\PasswordServices.properties
2. Create a new default properties file and name it:

   PasswordServices.properties
3. Save the new PasswordServices.properties file in the following directory:

   <*Web-Agent-Install-Dir*>\jpw\

Authentication Schemes Requiring Additional Attribute Information

To configure Password Services to work with Authentication Schemes requiring additional attributes you must add the following to Password Services’ HTML forms .template files and the sample .jsp forms files before users submit any data to the Policy Server:

```
document.PWChange.OLDPASSWORD.value="PASSWORD="+escape(
document.PWChange.OLDPASSWORD.value)
+"&<attribute>="+escape(document.PWChange.<attribute>.value)
```

Password Services’ .template files and the sample .jsp files are installed in the following locations:

- *web_agent_installation*/pw
- *web_agent_installation*/jpw/default
- *web_agent_installation*/pwservlet/default

As an example, the following shows how to modify the sample PWChange.jsp file (the text highlighted in bold are the new lines to add):

```
<html>
<head>
<title>SiteMinder Password Services</title>
…
…
function CheckForm(form)
{
…
…
else {
document.PWChange.OLDPASSWORD.value="PASSWORD="+escape(document.PWChange.OLDPAS
SWORD.value)+"&<attribute>="+escape(document.PWChange.<attribute>.value)
document.PWChange.submit()
return true
}
}
```

In another example, the following shows how to modify the sample PWChange.template file (the text highlighted in bold are the new lines to add):

```
<!-- SiteMinder Encoding=UTF-8; -->
<html><head><meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
<title>$TITLE$</title>
<script LANGUAGE="JavaScript">
<!-----
// Function for Validating Forms
function CheckForm(form)
{
…
…
else {
document.PWChange.OLDPASSWORD.value="PASSWORD="+escape(document.PWChange.OLDPAS
SWORD.value)+"&<attribute>="+escape(document.PWChange.<attribute>.value)
document.PWChange.submit()
return true
}
}
```

The Password Services Servlet provides Unicode support for multiple languages. The default encoding is UTF-8, which enables users to enter a password in any UTF-8 supported language.

The Password Services Servlet uses the HTTP protocol standard ACCEPT-LANGUAGE request header set by the browser to determine user language preferences. If the servlet can obtain the value of ACCEPT-LANGUAGE (which represents a standard language encoding, such as EN-US, or FR), it looks for appropriately localized JSP forms in a correspondingly named directory:

- NT: *web_agent_installation_dir*\jpw\<encoding>
- UNIX: *web_agent_installation_dir*/jpw/<encoding>

Where *web_agent_installation_dir* is the root installation directory and <encoding> is a standard language encoding.

As installed, Password Services provides a set of sample American English JSP files in the jpw directory. Password Services uses the contents of this folder whenever it cannot find a JSP directory corresponding to the ACCEPT-LANGUAGE value or it cannot obtain an ACCEPT-LANGUAGE value.

To localize servlet-based Password Services for another language, you must therefore create a new, appropriately named directory each containing a *complete* set of translated JSP forms.

Example localized JSP forms directory names:

Language	**Folder Location**
English (American)	EN
English (British)	EN-GB
French	FR
German	DE
Japanese	JP

If you want to localize for a different default language, you should rename the installed default directory to EN and create a new default directory containing a complete set of JSP forms translated into the new default language.

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 Specify Directory Attributes (see page 133).

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 you specify a value other than 0 or 1, the key is disabled and the Policy Server will redirect users to the login screen without the invalid current password message.

Chapter 22: 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 623).

You can configure the Agent that the Test Tool simulates during a test in the SiteMinder Agent group box.

Select a Test Mode

You can use one of the test modes in the following table 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.

Mode	**Description**
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 script file. This file can subsequently be used as an input file to repeat the test in playback mode. You can run multiple tests and record them to the same script file. The Test Tool appends the test results to the end of the file. You can then use the script file for regression and stress testing. Optionally, you can add a comment to the output script file by entering it in the Comment field before running a test. To stop recording, specify a new mode. **Note:** When you select Record, you must enter the path and filename for the text file where the test results will be stored in Output Script field.

Mode	Description
Basic Playback	Uses script files created in the Record mode to automate sequential tests for regression testing. When you select Basic Playback, the only active fields in the Test Tool are the Input Script and Output Script fields.
Advanced Playback	Runs multi-threaded stress tests. When you select Advanced Playback, the only active field in the Test Tool is the Control Script field.

More information:

Perform a Regression Test (see page 632)
Stress Tests (see page 633)

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 Configurations

To avoid re-entering user-supplied information, such as Agent, resource, and user information, you can save and load settings.

Use the Save Settings button to save the current settings to a plain text .ini file. When you click Save Settings, the Test Tool prompts you to enter a file name.

To retrieve saved information, click Load Settings, select the file, and click Open. The user-configured fields are automatically filled in with the values defined in the file.

You can configure the Agent that the Test Tool simulates during a test in the SiteMinder Agent group box.

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 622)
Calculate an Average Elapsed Time (see page 632)

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.

Stress Tests

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:

Thread Control Files (see page 633)

Thread Control Files

A thread control file determines the number of repetitions and threads the Test Tool runs in the Advanced Playback mode. It 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 format: <*script file name*>, <*number of script repetitions*>, <*number of threads*>.

For example:

# c:\temp\test_data.txt, 8, 6

This line indicates that:

- The input script is c:\temp\test_data.txt
- The Test Tool will run the script eight times
- There will be six simultaneous threads running the script

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.
.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.
.connect *<setting_file>*	Initializes the Test Tool using information from the 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.

```
.output
.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 Test Tool allows you to simulate user authentication and authorization. In the case of certificate-based authentication some additional configuration is required.

The format of the issuer DN and other attributes of a certificate may differ by Web Server vendor. For example, the issuer DN for a certificate on an IIS Web Server is different from the issuer DN for the same certificate on an iPlanet Web Server. To test certificate-based authentication schemes you must configure certificate mappings that will work correctly for different Web Server vendors.

Certificate Attributes that Require Custom Mappings

The representation of some common certificate attributes differ among the popular Web Server vendors. The attributes that can cause errors in the Test Tool are:

Email Address

Represented by E or Email depending on vendor

US State

Represented by S or ST depending on vendor

User ID Number

Represented by UID or UserID depending on vendor

Custom Attribute Mappings for Testing

If a user uses the test tool for a certificate authentication scheme, it will fail, even if it works normally (through a browser and the web server). In the authentication log, it appears that the test tool expects an attribute of the issuer DN to be represented differently than it actually appears in the DN. For example, the IssuerDN may use ST for state, when the test tool expects it to be S. The situation is similar for the other attributes listed in Certificate Attributes that Require Custom Mappings (see page 636).

This situation occurs when the issuer DN and other attributes may differ according to the Web Server Vendor (and the test tool's expectations). For example, the issuer DN for a certificate on an IIS Web server is different from the issuer DN for the same certificate on a Netscape Web server.

To resolve this situation, an administrator should create mappings for the Issuer DNs, so that the Policy Sever can accept the IssuerDn from different web servers.

Issuer DN Mapping

Different web servers can interpret the same Issuer DN in different ways. For example, one web server can give 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 web server can give 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 both possibilities, the administrator needs to create mappings for both issuer DNs.

Note: It is recommended that administrators create mappings to support various Web servers and the SiteMinder Test Tool.

Create Custom Certificate Mappings

You can use the Policy Server's certificate mapping feature to provide custom mappings for certificates.

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

To create 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: You can click Help for a description of fields, controls, and their respective requirements.
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 636).

7. Click Submit.

   The custom mapping is saved. The Policy Server now handles requests from both Web Servers and the 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 636).

More Information:

Certificate Mapping (see page 329)

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 640) 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 644).

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.

Install the Netscape 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 a command line utility. See the Netscape documentation for the details.

To install the Netscape Web Server Certificate

1. Once you create a key, request a certificate. This is done under Keys & Certificates: Request Certificate in Netscape Server Administration.
2. After creating the certificate request, go to the Certificate Authority and load the request. The Certificate Authority then creates a Server Certificate for you.
3. Once the Server Certificate is created, load it into the Web Server. This is done under Keys & Certificates: Install Certificate. When filling out this information be sure to select This Server under the Certificate For section.

4. Once the certificate is loaded, verify that it is there and trusted by going to Manage Certificates. Look for the ServerCert certificate. Select the Certificate and verify that it is trusted.
5. 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.

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.

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

There are five additional steps to take if you are still not receiving a certificate prompt.

- Verify that all Netscape browsers are configured to ask every time (see page 645)
- Verify that all web servers are configured to use SSL and require certificates (see page 645)
- Verify the following settings for each SiteMinder Virtual Directory (see page 646)
- Check the web server's certificate expiration (see page 646)
- Verify browser certificate validity (see page 647)

Verify That All Netscape Browsers Are Configured to Ask Every Time

Netscape 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.

To verify that all Netscape browsers are configured to ask every time

1. In the Netscape Browser, select Security from the tool bar.
2. Select Navigator.
3. In the Certificates to Identify You to a Web Site section, be sure it is set to Ask Every Time in the drop-down box.

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 (see page 640).

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

Netscape Browsers

1. In the Netscape Browser, select Security from the tool bar.
2. In the Security Info panel, click Yours under Certificates.
3. In the list box, under These are your certificates: you should see the certificate. If it does not appear, you need to install one.
4. If the certificate is in the list, click Verify to ensure that it is a valid certificate.

Internet Explorer Browsers

1. In the IE browser menu select View, Internet Options.
2. Select Certificates, Personal.
3. Verify that the certificate is listed there and that it is valid. If it is not there or is not valid, install a new certificate.

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 371)
Certificate Mapping (see page 329)

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 247)

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 652)

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 652)
2. Configure a LanMan User Directory Connection (see page 653)

Configure Registry Keys for a LanMan Directory Connection

The first procedure in configuring a LanMan directory connection is configuring the appropriate registry keys.

To configure registry keys for a LanMan directory connection

1. Select Run from the Windows Start menu.

   The Run dialog opens.

2. Enter **regedit,** and click OK.

   The Registry Editor opens.

3. Modify the following registry key:

   - In HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\Siteminder\CurrentVersion\Ds, the NameSpaces key is set to the following string value:

     "LDAP:,ODBC:,OCI:,WinNT:,Custom:,AD:"

   - Add the following string to the value data for the NameSpaces registry key: Lanman.

4. Create the following registry key:

   \HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Ds\Lanman_DC

5. Create a registry key of the NT Domain Name under the Lanman_DC key:

   \HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Ds\Lanman_DC\<*NT_domain_name*>

   For example:

   \HKEY_LOCAL_MACHINE\SOFTWARE\Netegrity\SiteMinder\CurrentVersion\Ds\Lanman_DC\MyDomain

6. Create a registry value named NumUserDir of type DWORD under the newly created NT Domain key. For the value data, enter the actual number of separate sets of user directories (maximum 16) in this NT domain.
7. Create String registry values of UserDir0, UserDir1, …, UserDirN, in sequential order starting from 0, for each failover list of BDCs.
8. Enter comma delimited strings for each failover list. SmDsLanman will read the lists and will find the first active BDC in each failover list to look up NT users and groups.
9. Repeat steps 5 through 7 for other NT domains.
10. Restart the Policy Server services.

    Note: More information on starting and stopping the Policy Server exists in the *Policy Server Management* guide.

Configure a LanMan User Directory Connection

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

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

To configure a LanMan user directory connection

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

   The Create User Directory pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create User Directory: *Name* pane opens.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Type the name and a description of the user directory in the fields on the General group box.
5. Select LanMan from the Namespace list.

   LanMan settings open.
6. Type the name of the NT Domain that you configured in the registry keys in the Domain Controller Key field.
7. Click Submit.

   The Create User Directory task is submitted for processing.

More information:

User Directories (see page 119)
Configure Registry Keys for a LanMan Directory Connection (see page 652)

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 196)

Appendix C: CA SSO/WAC Integration

This section contains the following topics:

Overview

SiteMinder provides single sign-on from SiteMinder to CA SSO environments. Users log into 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 re-enter 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 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, Secure Proxy Server, or both. You use a Web Agent or Secure Proxy Server 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, SiteMinder Secure Proxy Server, 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 656)

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 657)).
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 659)).
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 660)).

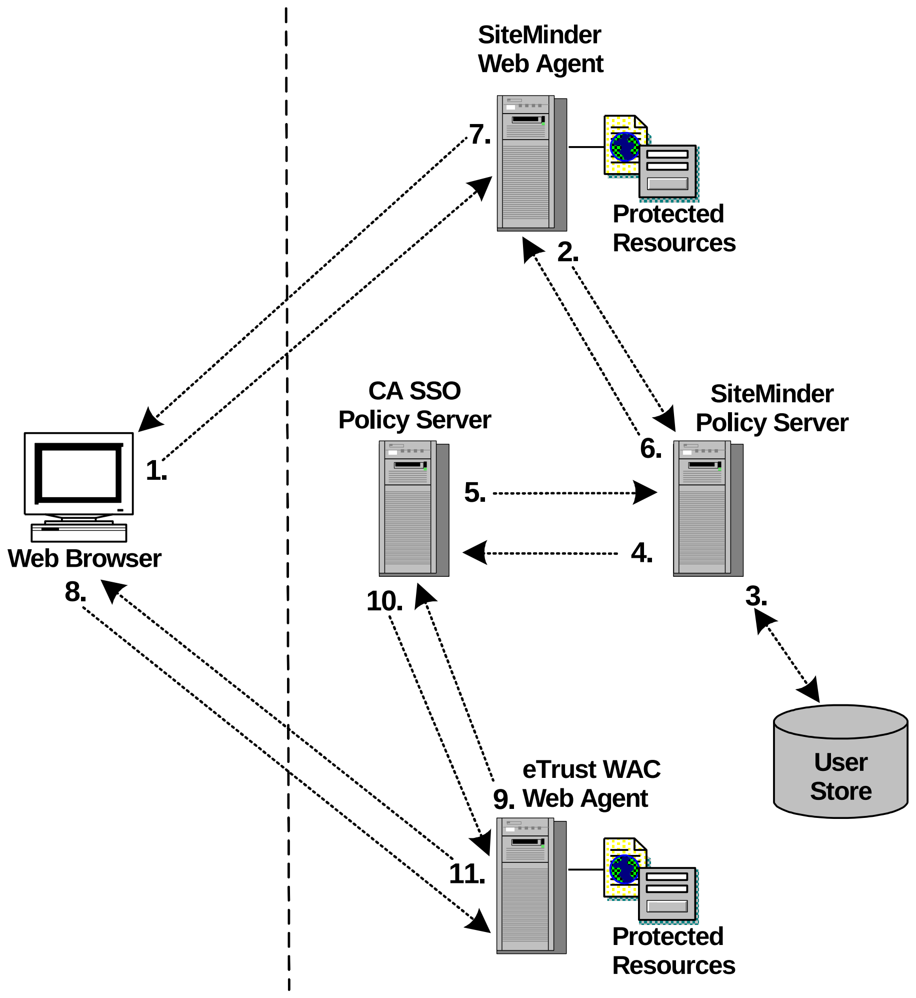
Note: In these examples, the SiteMinder and CA SSO Policy Servers authorization access steps to protected resources are omitted for clarity.

More information:

Configure Single Sign-On from CA SSO Client to SiteMinder (see page 666)

User Accesses SiteMinder-Protected Resource Before CA SSO

The following example illustrates a user accessing SiteMinder-protected resource before a WAC-protected resource:

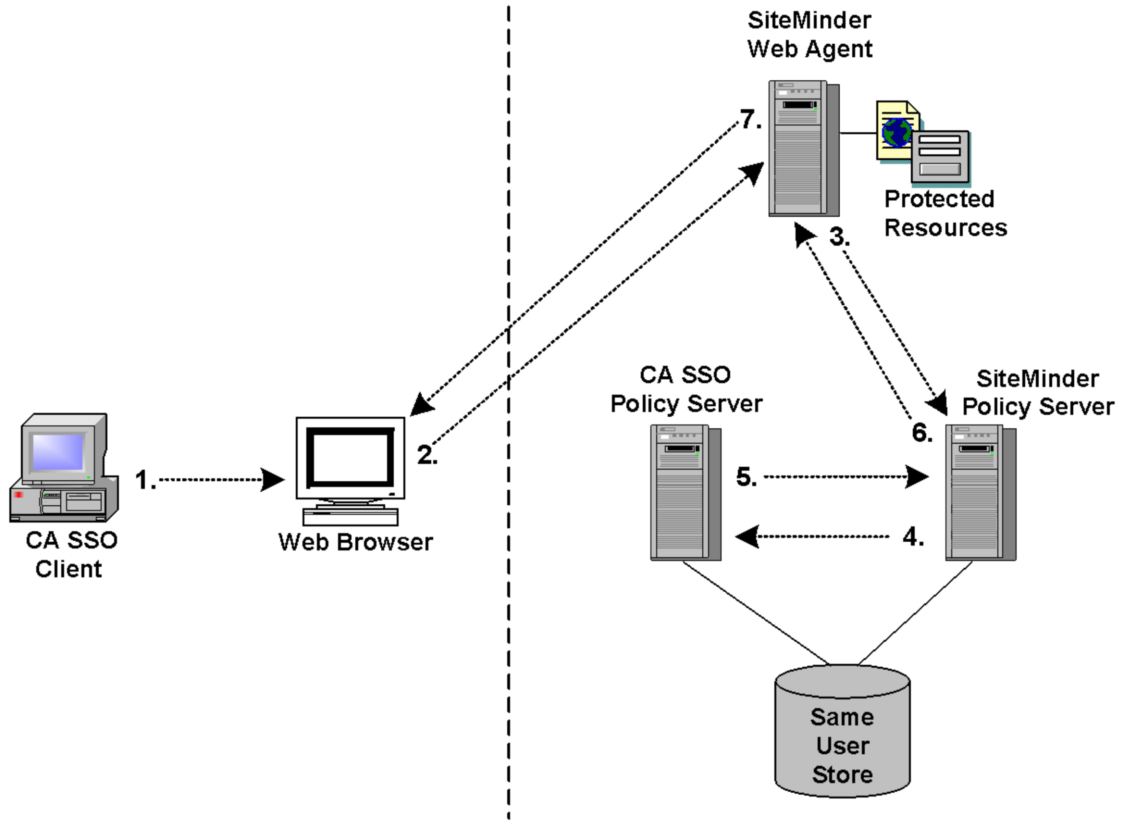


1. The user tries to access a SiteMinder-protected resource and the SiteMinder Web Agent/Secure Proxy Server intercepts the request. The user provides the Agent/SPS with authentication credentials.
2. The Web Agent/Secure Proxy Server forwards the credentials to the SiteMinder Policy Server for validation.
3. The SiteMinder Policy Server makes sure 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/Secure Proxy Server.
7. The SiteMinder Web Agent/Secure Proxy Server 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:



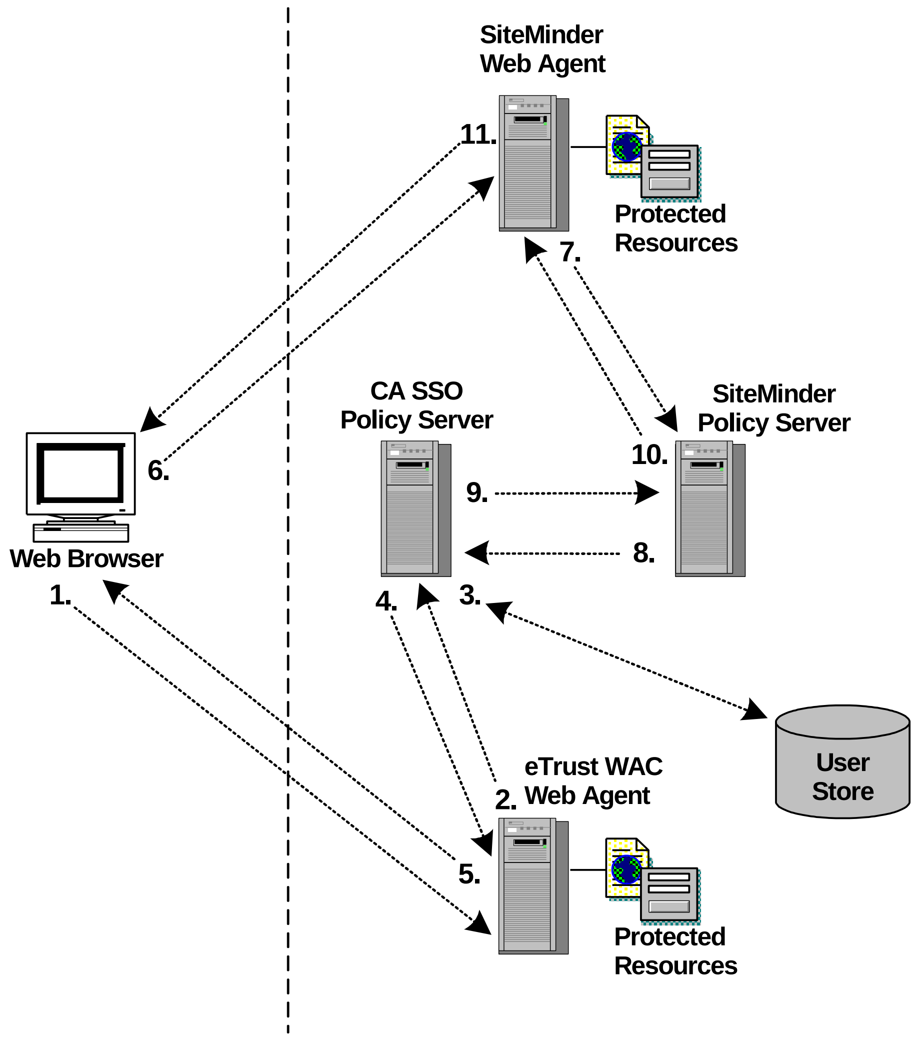
1. An authenticated CA SSO Client user launches a Web browser. While this is happening, the CA SSO Client places an CA SSO Web authentication cookie into the browser.
2. The user tries to access a SiteMinder-protected resource using the Web browser and the request is intercepted by the SiteMinder Web Agent/Secure Proxy Server.
3. The SiteMinder Web Agent/Secure Proxy Server 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/Secure Proxy Server.
7. The SiteMinder Web Agent/Secure Proxy Server 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/Secure Proxy Server intercepts the request.
7. The SiteMinder Web Agent/Secure Proxy Server 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/Secure Proxy Server.
11. The SiteMinder Web Agent/Secure Proxy Server 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: Ensure you install and configure the CA SSO Policy Server in an LDAP database. Installing the CA SSO Policy Server in an eTrust Access Control database causes the CA SSO/SiteMinder integration to fail. More installation information exists in the CA SSO documentation. When installing the CA SSO Policy Server, keep track of 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.

2. Ensure 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:

   - (Windows) If your SiteMinder Policy Servers are operating at r12 SP1 CR3 or later, install the VC8 Redistributable package on each SiteMinder Policy Server that will be communicating with the CA SSO Policy Server. The VC8 Redistributable package installs runtime components of Visual C++ Libraries required for the integration.

     Note: The package (vcredist_x86.exe) is located in SiteMinder\bin.

   - If the integration is to operate in FIPS-only mode, SiteMinder Policy Servers must be operating at r12 SP1 CR3 or later. If necessary, upgrade the SiteMinder Policy Servers that will be communicating with the CA SSO Policy Server.

4. Refer to the r12 SP1 SiteMinder Platform Support Matrix to ensure 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.

To locate the support matrix from the Support site

1. Click Technical Support.
2. Click Support By Product or Solution.

3. Select CA SiteMinder Web Access Manager from the Select a Product or Solution Page list.
4. Click Platform Support Matrices in the Product Status group box.

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 Secure Proxy Server

Enable the SiteMinder SSO Plug-in installed with the Web Agent or Secure Proxy Server:

For the r12 SP1 IIS 6.0 or Apache 2.0 Web Agent

- Remove the comment (#) character from the following line in the WebAgent.conf file:

  #LoadPlugin=<Path to eTSSOPlugin.dll or libetssoplugin.so>

- To locate the WebAgent.conf for IIS 6.0/Apache 2.0, see the "Modifying the WebAgent.conf File (All Web Agents)" section in the *Web Agent Configuration Guide*.
- For instructions on enabling/disabling the plug-in, see the "Loading Web Agent Plugins for IIS 6.0 and Apache 2.0" section in the *Web Agent Configuration Guide*.

Note: Restart the Web server after you modify the WebAgent.conf file so the new configuration settings take effect.

For the 6.0 Secure Proxy Server

- 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 the Secure Proxy Server 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:

Configure an smauthetsso Custom Authentication Scheme (see page 671)
Configure an smetssocookie Web Agent Active Response Attribute (see page 669)
Domains (see page 369)
Realms (see page 377)
Rules (see page 389)
Configure a Rule for Web Agent Actions (see page 397)

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:

Configure an smauthetsso Custom Authentication Scheme (see page 671)
Domains (see page 369)
Realms (see page 377)
Rules (see page 389)

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 Secure Proxy Server Configuration Steps:

1. Enable the SSO plug-in installed with the Web Agent or Secure Proxy Server, 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 SP1 IIS 6.0 or Apache 2.0 Web Agent

   See the Web Agent Configuration Guide.

   6.0 Secure Proxy Server

   SPS_install_dir\proxy-engine\conf\defaultagent\

   SPS_install_dir

   Secure Proxy Server 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.

More information:

Configure an smauthetsso Custom Authentication Scheme (see page 671)

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, Domains.
2. Click Response, Create Response.

   The Create Response pane opens.
3. Verify that Create a new object is selected, and click OK.

   The Create Response: Select Domain pane opens.
4. Click the radio button on the left of the domain you want, and then click Next.

   The Create Response: Define Response pane opens.
5. Define a Name and Description for the response.
6. Ensure that the SiteMinder radio button is selected and that Web Agent appears in the Agent Type drop-down list.
7. Click Create Response Attribute.

   The Create Response Attribute pane opens.
8. Verify that Create a new object is selected, and then click OK.

   The Create Response Attribute: *Name* pane opens.
9. Click the Attribute drop-down list and select WebAgent-HTTP-Cookie-Variable.
10. Click the Active Response radio button on the Attribute Kind group box.

    Additional fields appear in the Attribute Fields group box.
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 pane reopens.

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 an 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 an 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.

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

To configure the authentication scheme

1. Click Infrastructure, Authentication.
2. Click Authentication Scheme, Create Authentication Scheme.

   The Create Authentication Scheme pane opens.
3. Click OK.

   Authentication scheme settings open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
4. Select Custom Template from the Authentication Type Style list.

   Scheme-specific fields and controls open.

   Note: You can click Help for a description of fields, controls, and their respective requirements.
5. Enter **smauthetsso** in the Library field.
6. Enter and confirm the password of the CA SSO Policy Server administrator in the Secret and Confirm Secret fields.
7. 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 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.

8. Click Submit.

   The authentication scheme is saved and may be assigned to a realm.

More information:

HTML Forms Authentication Schemes (see page 261)

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.

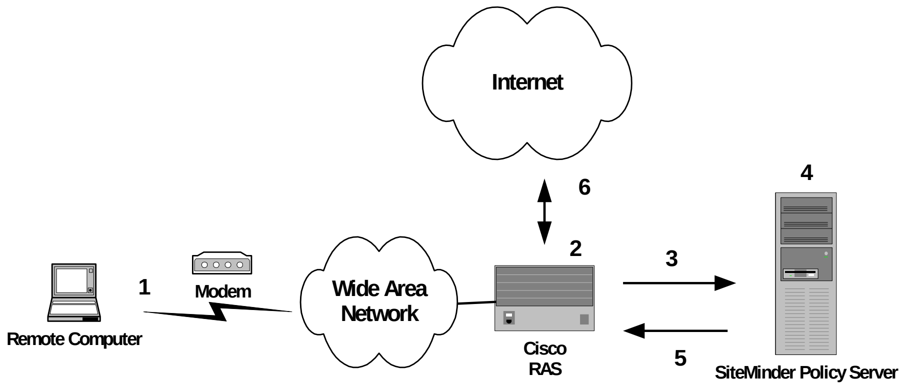
More information:

Generate RADIUS Logs for Accounting and Debugging (see page 704)

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.

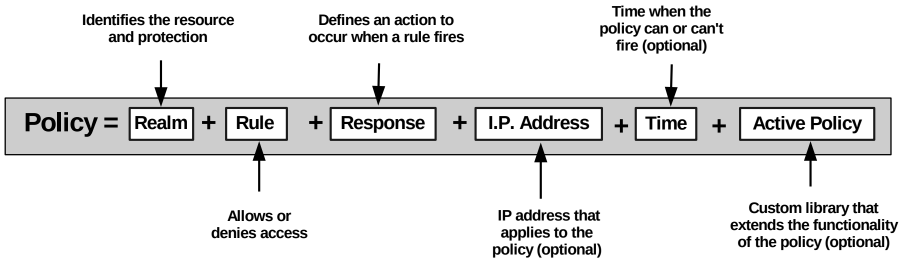
     The RAS notifies the Policy Server that the session has begun and when the session ends.

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.

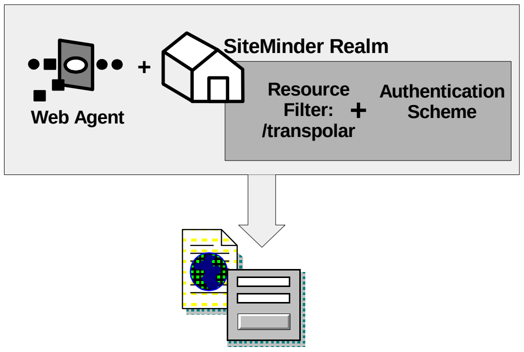

Identifies the resource and protection
Defines an action to occur when a rule fires
Time when the policy can or can't fire (optional)
Policy = Realm + Rule + Response + I.P. Address + Time + Active Policy
Allows or denies access
IP address that applies to the policy (optional)
Custom library that extends the functionality of the policy (optional)

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.

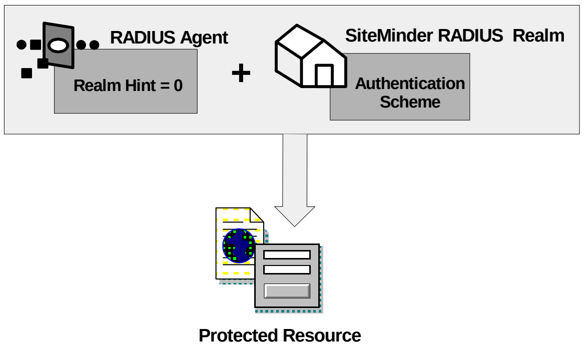
Policy Component	**In a RADIUS Policy, this item:**	**In a SiteMinder Agent Policy, this item:**
Realm	■ Identifies the Agent. ■ Identifies the authentication scheme. ■ Defines session timeouts.	■ Defines the resource filter (directory within the domain that the SiteMinder Agent will govern). ■ Identifies 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

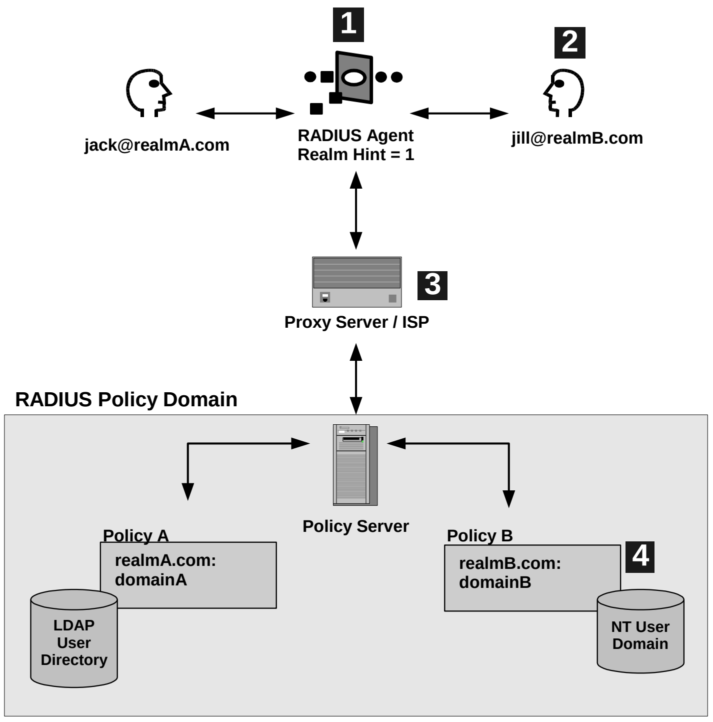
How does a RADIUS Agent protect a NAS device that must authenticate users in different domains, such as domainA and domainB? A realm hint is a RADIUS attribute that enables SiteMinder to determine the correct domain in which to authenticate a user. You must provide a RADIUS Agent with one of the following 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.

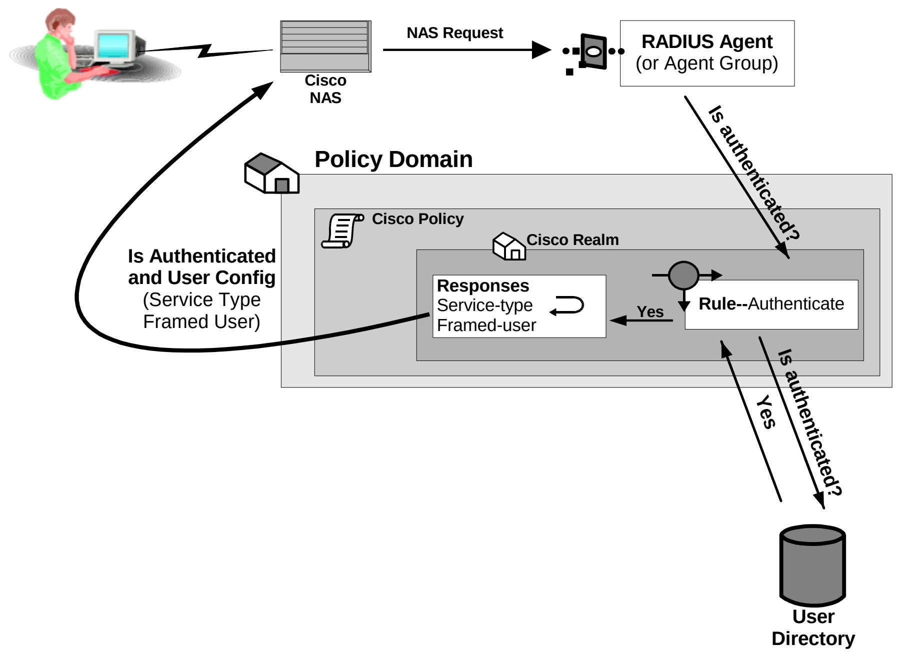
Using responses, you can provide the NAS device with user profile information that assigns privileges to the user. For example, you could allow one user unlimited access to a resource, yet limit another user's access to the same resource. Used in this way, responses give you the ability to authorize 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 684).

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-AVpair* 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 685)

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 685)

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 693).
- 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 How Authenticating Users in Heterogeneous RADIUS Environments with Two User Directories Works (see page 697).

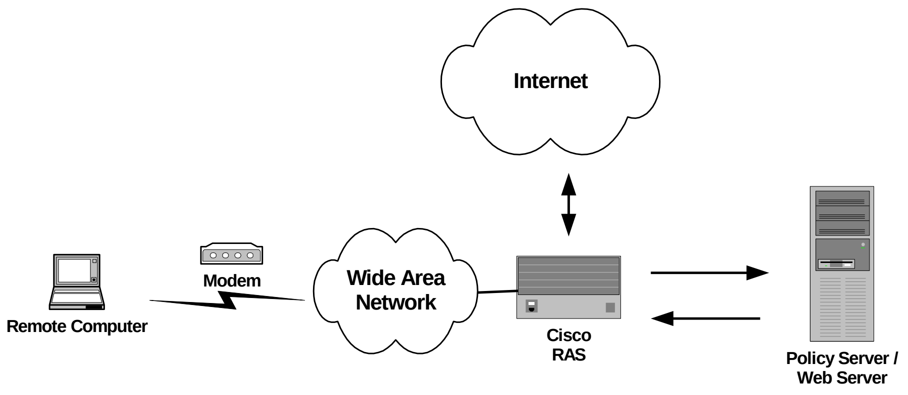
Guidelines for Protecting RADIUS Devices

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:



To setup SiteMinder in a homogeneous RADIUS environment

1. Configure the system:
   a. Define the RADIUS Agent, as explained in Configure a RADIUS Agent (see page 112).
   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 (see page 384).
   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 (see page 398).
   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 (see page 419) and RADIUS Agent Response Attributes (see page 419).
   e. Create a policy that binds the rule and response with the user directory, as explained in Configure a Policy (see page 443).

More Information:

How RADIUS Authentication Works with the Policy Server (see page 674)

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.

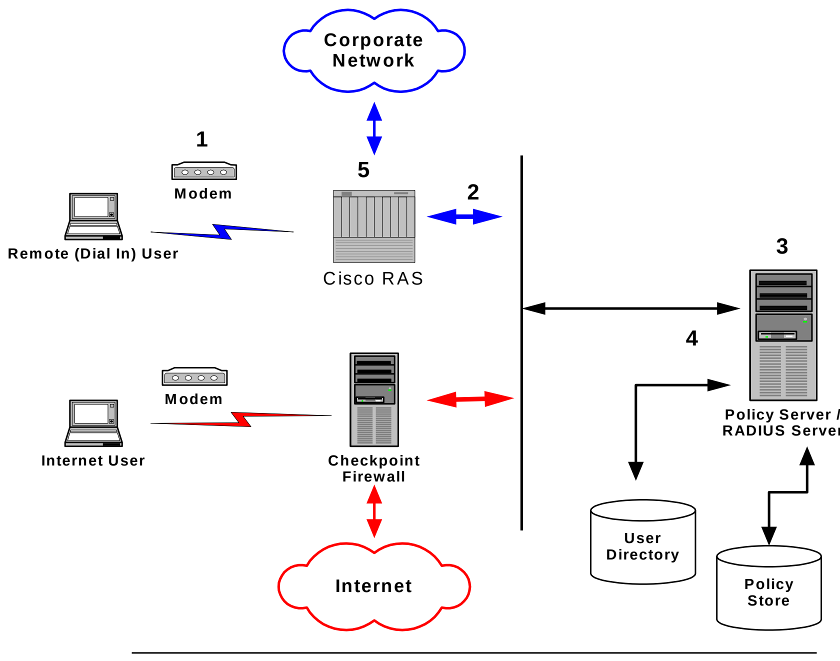
Authenticate Users in Heterogeneous RADIUS Environments with One User Directory

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

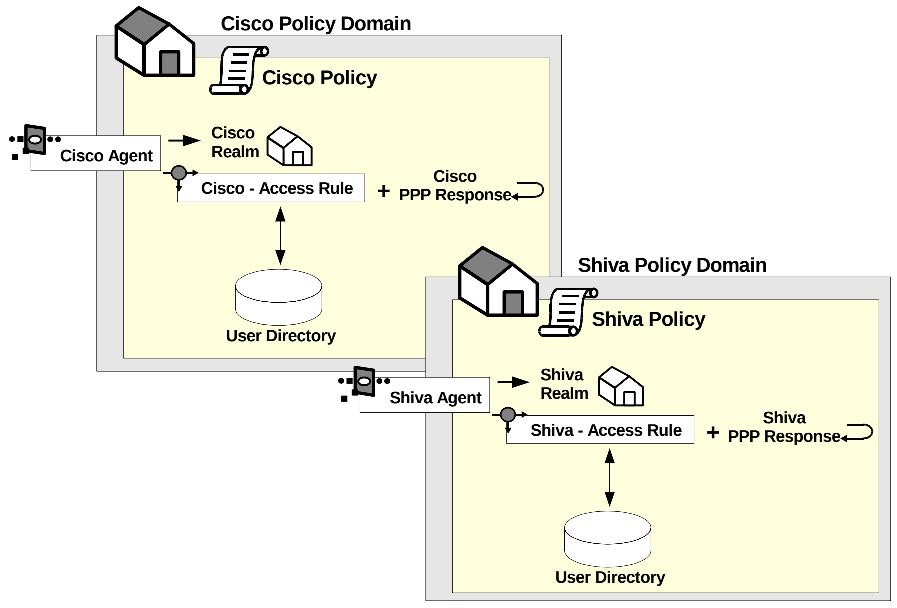
This system configuration differs from the homogeneous environment; you must now create two Agents.

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 696).
   b. Setup a user directory against which to authenticate RADIUS users, as described in Configure the User Directory (see page 696).
   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.

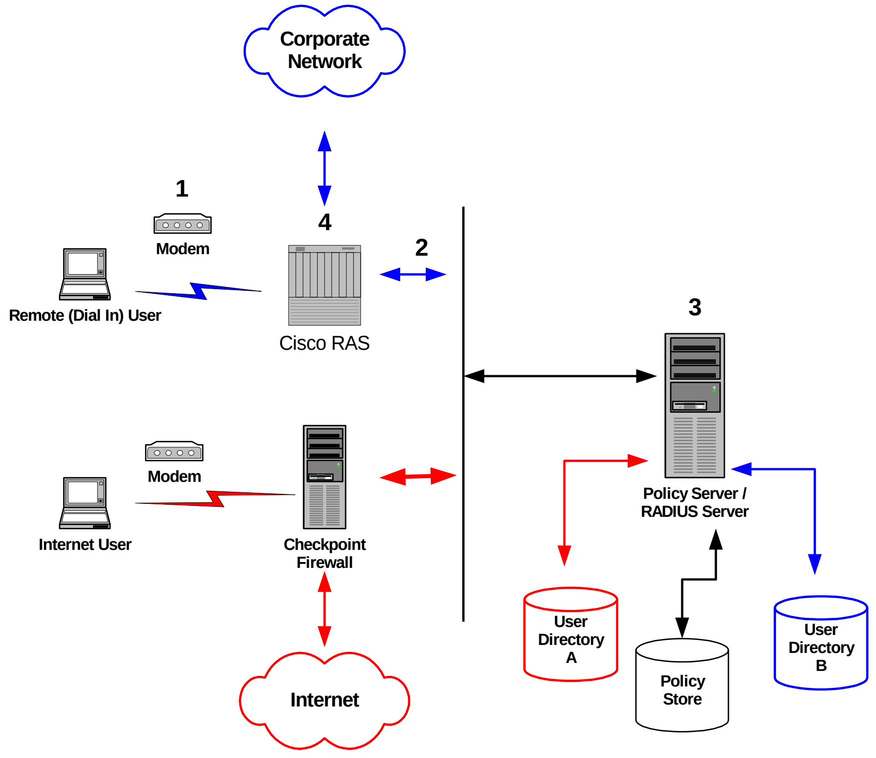
How to Authenticate Users in Heterogeneous RADIUS Environments with Two User Directories

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:



In the heterogeneous network topology shown in the previous diagram, the Policy Server authenticates users of two NAS devices: a Cisco RAS and a Checkpoint firewall.

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.

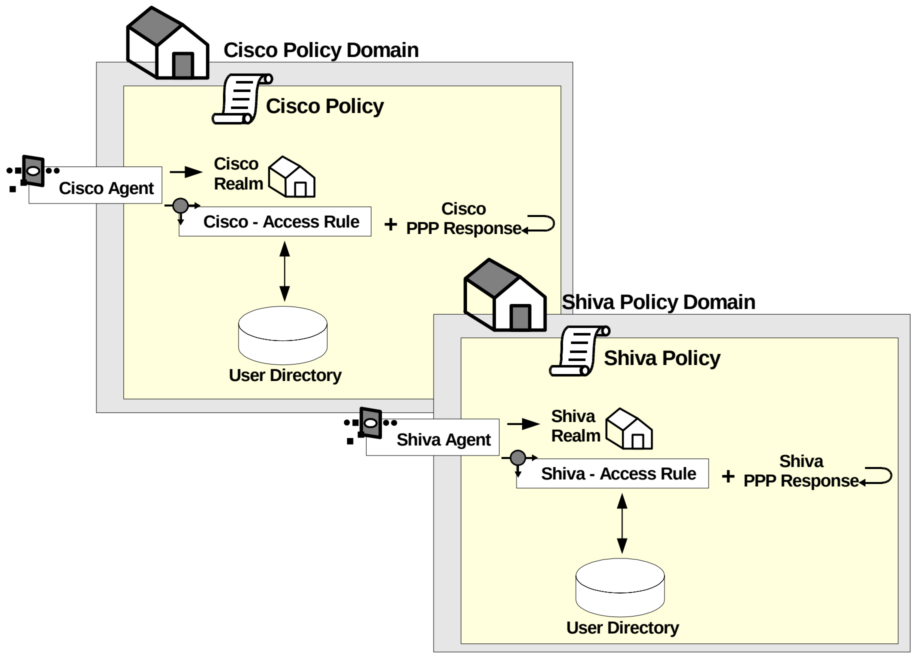
When the Internet user attempts to dial into the Internet Service Provider by using the Checkpoint Firewall, this same process of authentication occurs. However, the Policy Server evaluates the Internet user’s authentication information against a different user directory.

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 700).
   b. Set up the user directories, as described in Set Up User Directories (see page 700).
   c. Create two policy domains, as described in Create Two Policy Domains (see page 700).
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 696)

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.

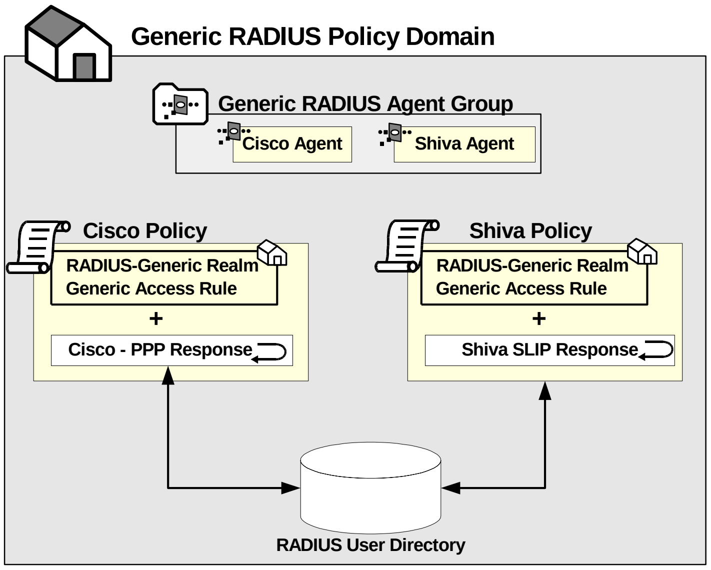
Set Up RADIUS Agent Groups

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.



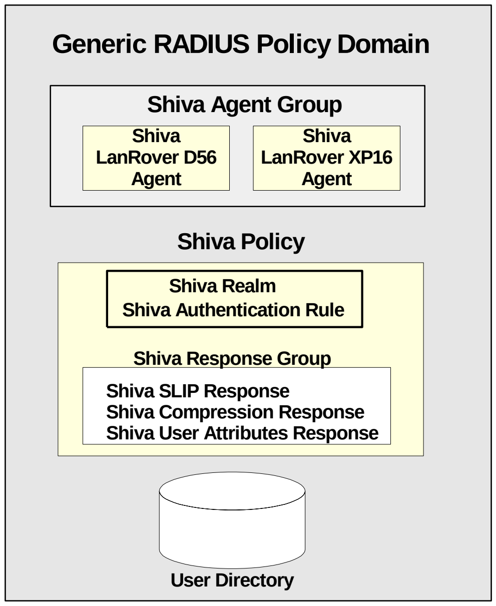
The procedure for setting up a RADIUS Agent Group is in Configure an Agent Group (see page 114).

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:


Generic RADIUS Policy Domain
Shiva Agent Group
Shiva LanRover D56 Agent
Shiva LanRover XP16 Agent
Shiva Policy
Shiva Realm
Shiva Authentication Rule
Shiva Response Group
Shiva SLIP Response
Shiva Compression Response
Shiva User Attributes Response
User Directory

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, for example:

```
log.txt.08Dec1999_13_30_57
```

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 706).
3. Configure the SiteMinder Test Tool to test RADIUS policies, as explained in Test RADIUS Policies (see page 706).

Configure the Policy Server Management Console

To configure the Policy Server Management console

1. Start the Policy Server Management Console.
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.
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'
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 755)
LOOP Function--Call a Virtual Attribute in a Loop (see page 771)

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 716)

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 716)

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 718)

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 717)

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 719)

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 718)

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 720)
Containment Operators (see page 721)

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 720)
Containment Operators (see page 721)

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 720)
Ends-with Operators (see page 720)

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 726)
Set Union Operators (see page 726)

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 721)
Set Union Operators (see page 726)

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 721)
Set Intersection Operators (see page 726)

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 727)
OR Operator (see page 728)
Exclusive OR Operator (see page 728)

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 727)
OR Operator (see page 728)
Exclusive OR Operator (see page 728)

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 727)
AND Operator (see page 727)
Exclusive OR Operator (see page 728)

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 727)
AND Operator (see page 727)
OR Operator (see page 728)

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 730)
Arithmetic Multiplication Operator (see page 730)
Arithmetic Division Operator (see page 731)

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 729)
Arithmetic Multiplication Operator (see page 730)
Arithmetic Division Operator (see page 731)

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 729)
Arithmetic Subtraction Operator (see page 730)
Arithmetic Division Operator (see page 731)

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 729)
Arithmetic Subtraction Operator (see page 730)
Arithmetic Multiplication Operator (see page 730)

Conditional Operator

The conditional operator evaluates a Boolean expression (the first operand) and based on the result returns one of two other operands.

If the Boolean operand is TRUE, the result of the operator is the second operand (the THEN clause), otherwise the third operand is returned (the ELSE clause). The second and the third operand must be the same type.

Only one clause of this operator is ever evaluated. In other words, if the THEN clause (operand) is evaluated, the ELSE clause (operand) is not evaluated. (And the opposite is also true.)

Examples:

"JUAN" = "juan" ? "YES" : "NO"
Result = "NO"

"JUAN" ~= "juan" ? "YES" : "NO"
Result = "YES"

Indexing Operator

The indexing operator takes two arguments: a set and a number (the index). The set is broken into components and the component corresponding to the specified index is returned (like a traditional array). If the index is out of range, a blank string is returned.

The first element in the set is at index zero (0).

Examples:

"Sun^Mon^Tue^Wed^Thu^Fri^Sat"[2]
Result = "Tue"

"Sun^Mon^Tue^Wed^Thu^Fri^Sat"[0]
Result = "Sun"

Functions Available within Expressions

This topic lists all of the functions by area of utility.

Numeric Functions

- ABS
- ALL
- ANDBITS
- ANY
- HEX
- MAX
- MIN
- MOD

- NOTBITS
- ORBITS
- SIGN
- XORBITS

String Functions

- AFTER
- BEFORE
- CENTER
- CHAR
- FIND
- LCASE
- LEFT
- LEN
- LPAD
- LTRIM
- MID
- PCASE
- RIGHT
- RPAD
- RPT
- RTRIM
- SPACE
- TRANSLATE
- UCASE

Set Functions

- COUNT
- ENUMERATE
- FILTER

- LOOP
- SORT

Date Functions

- DATE (form 1)
- DATE (form 2)
- DATEFROMSTRING
- DATETOSTRING
- DAY
- DOW
- DOY
- HOUR
- HOUR24
- MINUTE
- MONTH
- NOW
- NOWGMT
- SECOND
- YEAR
- YEAR4

Conversion Functions

- BOOLEAN
- NUMBER
- STRING

Generic User Directory I/O Functions

- GET
- SET

LDAP Functions

- ABOVE
- AT
- BELOW
- COMMONDN

- EXPLODEDN
- PARENTDN
- RDN
- RELATIONDN

URL/Path Handling Functions

- QS
- URL
- URLDECODE
- URLENCODE

Logging Functions

- ERROR
- INFO
- TRACE
- WARNING

File I/O Functions

- EXISTS
- KEY
- LOG

Error Handling Functions

- MAYBE
- THROW
- VEXIST

User Context Management Functions

- WITH

Miscellaneous Functions

- EVALUATE

ABOVE Function--Is User Above Specified LDAP DN

The ABOVE function returns a TRUE if the current user exists within a container that is *above* the container specified by the LDAP Distinguished Name (DN) that is passed to the function.

Note: The user's DN and the specified DN are assumed to be in the same directory.

Syntax

The ABOVE function has the following format:

```
ABOVE(LDAP_Distinguished_Name)
```

Parameters

The ABOVE function accepts the following parameter:

LDAP_Distinguished_Name (string)

Return Value

The ABOVE function returns a Boolean value.

Remarks

LDAP Only: Yes

ABS Function--Find the Absolute Value

The ABS function finds the absolute value of a number.

Syntax

The ABS function has the following format:

```
ABS(number)
```

Parameters

The ABS function accepts the following parameter:

number (number)

Return Value

The ABS function returns a number.

Example

```
Return_value=ABS(3)
Return_value=3

Return_value=ABS(-2)
Return_value=2
```

AFTER Function--Find a String

The AFTER function finds the specified instance of a search string in a source string and returns that part of the source string that follows the search string. If the search string is not found, then the AFTER function returns a blank string.

Syntax

The AFTER function has the following format:

AFTER(source_string, search_string[, n][, not_case_sensitive])

Parameters

The AFTER function accepts the following parameters:

source_string (string)

search_string (string)

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.

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.

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', -1, TRUE)
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', 2, TRUE)
Return_value='ic'
```

More information:

BEFORE Function--Find a String (see page 741)
FIND Function--Return Position in String (see page 761)

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 780)
ORBITS Function--Perform a Bitwise OR Operation (see page 783)
XORBITS Function--Perform a Bitwise XOR Operation (see page 806)

ANY Function--Any Bits Set

The ANY function accepts two numbers and returns a TRUE if *any* of the bits that are set in the second number are also set in the first number.

Syntax

The ANY function has the following format:

```
ANY(number1, number2)
```

Parameters

The ANY function accepts the following parameters:

number1 (number)

number2 (number)

Return Value

The ANY function returns a Boolean value.

Example

```
Return_value=ANY(7, 2)
Return_value=TRUE

Return_value=ANY(7, 15)
Return_value=TRUE
```

AT Function--Is User at Specified LDAP DN

The AT function returns a TRUE if the current user exists within the container specified by the LDAP Distinguished Name (DN) that is passed to the function.

Note: The user's DN and the specified DN are assumed to be in the same directory.

Syntax

The AT function has the following format:

```
AT(LDAP_Distinguished_Name)
```

Parameters

The AT function accepts the following parameter:

LDAP_Distinguished_Name (string)

Return Value

The AT function returns a Boolean value.

Remarks

LDAP Only: Yes

BEFORE Function--Find a String

The BEFORE function finds the specified instance of a search string in a source string and returns that part of the source string that precedes the search string. If the search string is not found, then the BEFORE function returns the entire source string.

Syntax

The BEFORE function has the following format:

```
BEFORE(source_string, search_string[, n][, not_case_sensitive])
```

Parameters

The BEFORE function accepts the following parameters:

source_string (string)

search_string (string)

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.

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.

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', -1, TRUE)
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', 2, TRUE)
Return_value='EricE'
```

More information:

AFTER Function--Find a String (see page 737)
FIND Function--Return Position in String (see page 761)

BELOW Function--Is User Below Specified LDAP DN

The BELOW function returns a TRUE if the current user exists within a container that is *below* the container specified by the LDAP Distinguished Name (DN) that is passed to the function.

Note: The user's DN and the specified DN are assumed to be in the same directory.

Syntax

The BELOW function has the following format:

```
BELOW(LDAP_Distinguished_Name)
```

Parameters

The BELOW function accepts the following parameter:

LDAP_Distinguished_Name (string)

Return Value

The BELOW function returns a Boolean value.

Remarks

LDAP Only: Yes

BOOLEAN Function--Convert to a Boolean Value

The BOOLEAN function converts a number or string value to a Boolean 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 a value of TRUE or 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 782)
STRING Function--Convert to a String (see page 796)

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, [padding, ]length)
```

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.

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.

length (number)

Specifies the length of the resulting string.

Return Value

The CENTER function returns a string.

Example

```
Return_value=CENTER('Robin', '*', 9)
Return_value='**Robin**'

Return_value=CENTER('Robin', 'ooo', 7)
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 772)
RPAD Function--Pad a String on the Right (see page 790)

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 759)
PARENTDN Function--Retrieve Parent in LDAP Tree (see page 784)
RDN Function--Retrieve First Component of LDAP DN (see page 787)
RELATIONDN Function--Compare Two Distinguished Names (see page 788)

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 755)
FILTER Function--Test Set Elements (see page 760)
SORT Function--Sort a Set (see page 795)

DATE Function--Set to Midnight (form 1)

The DATE function (form 1) accepts a numeric representation of date and time and sets the time to midnight on the specified date.

Syntax

The DATE function (form 1) has the following format:

```
DATE(date_time)
```

Parameters

The DATE function (form1) accepts the following parameter:

date_time (number)

Specifies the date and time as the number of seconds that have passed since January 1, 1970. If *date_time* is not a valid representation of date and time, the function returns -1.

Return Value

The DATE function (form1) returns a number.

DATE Function--Convert Year, Month, Day, Hours, Minutes, and Seconds (form 2)

The DATE function (form 2) accepts six numbers that represent the year, month, day, hours, minutes, and seconds and converts them to a numeric representation of date and time. Date and time are represented numerically as the number of seconds that have passed since January 1, 1970. If the three time parameters are omitted, the function sets the time to midnight on the specified date.

Syntax

The DATE function (form 2) has the following format:

```
DATE(year, month, day[, hours, minutes, seconds])
```

Parameters

The DATE function (form 2) accepts the following parameters:

year (number)

Specifies a four-digit representation of the year.

Example: 2007

month (number)

Specifies the month.

Range: 1-12

day (number)

Specifies the day.

Range: 1-31

hours (number)

(Optional) Specifies the number of hours.

Range: 0-23

minutes (number)

(Optional) Specifies the number of minutes.

Range: 0-59

seconds (number)

(Optional) Specifies the number of seconds.

Range: 0-59

Return Value

The DATE function (form 2) returns a number.

DATEFROMSTRING Function--Convert String to Number

The DATEFROMSTRING function accepts a string representation of a date, a time, or both and converts the string to a numeric representation. Date and time are represented numerically as the number of seconds that have passed since January 1, 1970.

Syntax

The DATEFROMSTRING function has the following format:

```
DATEFROMSTRING(date_time[, format_string])
```

Parameters

The DATEFROMSTRING function accepts the following parameters:

date_time (string)

format_string (string)

> (Optional) Specifies the format of *date_time*. For example, the *format_string* "YYYYMMDD" specifies the format of "20020223." If the *format_string* is omitted, the function uses the default format of "YYYYMMDDhhmmss." If *date_time* is invalid, the function returns -1.

Return Value

The DATEFROMSTRING function returns a number.

Example

```
Return_value=DATEFROMSTRING('20020223')
Return_value=1024804800
```

DATETOSTRING Function--Convert Number to String

The DATETOSTRING function accepts a numeric representation of a date, a time, or both and converts the number to a string representation.

Syntax

The DATETOSTRING function has the following format:

```
DATETOSTRING(date_time[, format_string])
```

Parameters

The DATETOSTRING function accepts the following parameters:

date_time (number)

> Specifies a date, a time, or both as the number of seconds that have passed since January 1, 1970. If the *date_time* is invalid, the function returns the value "INVALID DATE".

format_string (string)

(Optional) Specifies the format of a date, a time, or both in the resulting string using the format codes listed in the table. Any characters or character combinations not listed in the table are inserted in the resulting string as is. If the *format_string* is omitted, the function formats the resulting string using the date and time representation appropriate for the locale that is currently stored in the Policy Server. The format codes and resulting formats are:

Code	**Resulting Format**
%a	Abbreviated weekday name (English)
%A	Full weekday name (English)
%b	Abbreviated month name (English)
%B	Full month name (English)
%c	Date and time representation appropriate for current locale
%#c	Long date and time representation appropriate for current locale
%d	Day of month as decimal number (01-31)
%H	Hour in 24-hour format (00-23)
%I	Hour in 12-hour format (01-12)
%j	Day of year as decimal number (001-366)
%m	Month as decimal number (01-12)
%M	Minute as decimal number (00-59)
%P	Local am and pm indicator for 12-hour clock
%S	Second as decimal number (00-59)
%U	Week of year as decimal number with Sunday as first day of week (00-53)
%w	Weekday as decimal number (0-6 with Sunday as 0)
%W	Week of year as decimal number with Monday as first day of week (00-53)
%x	Date representation appropriate for current locale
%#x	Long date representation appropriate for current locale
%X	Time representation appropriate for current locale
%y	Year without century as decimal number (00-99)
%Y	Year with century as decimal number
%z, %Z	Time-zone name or abbreviation if known
%%	Percent sign

Note: The pound sign modifies the meaning of the format codes, as follows.

- The format code %#c specifies the long date and time representation appropriate for the current locale.
- The format code %#x specifies the long date representation appropriate for the current locale.
- In all other cases, the pound sign, which is inserted in the format code after the percent sign, results in the removal of leading zeros, if any.
- The pound sign has no effect on codes that format alpha characters, such as the names of months.

Return Value

The DATETOSTRING function returns a string.

Example

```
Return_value=DATETOSTRING(1024804800, '%Y%m%d')
Return_value='20020223'

Return_value=DATETOSTRING(1024804800, '%Y%#m%#d')
Return_value='2002223'
```

DAY Function--Return Day of Month

The DAY function accepts a numeric representation of date and time and returns the number corresponding to the day of the month.

Syntax

The DAY function has the following format:

```
DAY(date_time)
```

Parameters

The DAY function accepts the following parameter:

date_time (number)

Specifies the date and time as the number of seconds that have passed since January 1, 1970. If the date is invalid, the function returns -1.

Return Value

The DAY function returns a number from 1 to 31.

Example

```
Return_value=DAY(1024804800)
Return_value=23
```

More information:

DOW Function--Return Day of Week (see page 754)
DOY Function--Return Day of Year (see page 755)

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 753)
DOY Function--Return Day of Year (see page 755)

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 753)
DOW Function--Return Day of Week (see page 754)

ENUMERATE Function--Test Set Elements

The ENUMERATE function passes each element in the specified set to the named expression specified by the parameter #*virtual_user_attribute* or @*user_class* using pasting. If the named expression returns a value, the element is included in the resulting string.

Syntax

The ENUMERATE function has the following format:

```
ENUMERATE(set, #virtual_user_attribute | @user_class)
```

Parameters

The ENUMERATE function accepts the following parameters:

set (string)

Specifies a string of elements that are separated by the caret character: 'element1^element2'. Each element is a string.

#virtual_user_attribute (named expression)

Specifies a named expression that calculates a user attribute.

@user_class (named expression)

Specifies a named expression that tests for membership in a user directory or group.

Return Value

The ENUMERATE function returns a string.

Example

Assume that the following statements are true:

- The *#virtual_user_attribute* is #RTrimset.
- The named expression specified by #RTrimset is the built-in function RTRIM.
- RTRIM accepts one parameter.
- The parameter is represented by the placeholder %0.
- Therefore, #RTrimset := RTRIM(%0).

Return_value=ENUMERATE('First   ^Middle   ^Last   ', #RTrimset)
Return_value='First^Middle^Last'

More information:

COUNT Function--Count the Elements in a Set (see page 748)
FILTER Function--Test Set Elements (see page 760)
LOOP Function--Call a Virtual Attribute in a Loop (see page 771)
SORT Function--Sort a Set (see page 795)
Pasting (see page 713)

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.

Example

Return_value=ERROR('Invalid Access')
Return_value=''

More information:

INFO Function--Write INFO Message to Console Log (see page 766)
TRACE Function--Write Trace Entry to Console Log (see page 798)
TRACE Function--Write Trace Entry to Console Log (see page 805)

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 766)

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 747)
PARENTDN Function--Retrieve Parent in LDAP Tree (see page 784)
RDN Function--Retrieve First Component of LDAP DN (see page 787)
RELATIONDN Function--Compare Two Distinguished Names (see page 788)

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 748)
ENUMERATE Function--Test Set Elements (see page 755)
SORT Function--Sort a Set (see page 795)

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[, n][, not_case_sensitive])
```

Parameters

The FIND function accepts the following parameters:

source_string (string)

search_string (string)

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.

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.

Return Value

The FIND function returns a number.

Example

```
Return_value=FIND('PhoebePhoebe', 'oe', 1, FALSE)
Return_value=3

Return_value=FIND('PhoebePhoebe', 'OE', 1, FALSE)
Return_value=0

Return_value=FIND('PhoebePhoebe', 'OE', 1, TRUE)
Return_value=3

Return_value=FIND('PhoebePhoebe', 'oe', -1, FALSE)
Return_value=9

Return_value=FIND('PhoebePhoebe', 'OE', -1, FALSE)
Return_value=0

Return_value=FIND('PhoebePhoebe', 'OE', -1, TRUE)
Return_value=9
```

Return_value=FIND('PhoebePhoebe', 'oe', 2, FALSE)
Return_value=9

Return_value=FIND('PhoebePhoebe', 'OE', 2, FALSE)
Return_value=0

Return_value=FIND('PhoebePhoebe', 'OE', 2, TRUE)
Return_value=9

More information:

AFTER Function--Find a String (see page 737)
BEFORE Function--Find a String (see page 741)

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 765)

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 764)

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 758)

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 784)
UCASE Function--Convert to Upper Case (see page 799)

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 789)
MID Function--Return Part of a String (see page 776)

LEN Function--Return the Length of a String

The LEN function returns the length of a string.

Syntax

The LEN function has the following format:

```
LEN(source_string)
```

Parameters

The LEN function accepts the following parameter:

source_string (string)

Return Value

The function returns a number.

Example

```
Return_value=LEN("JuanJuan")
Return_value=8
```

LOG Function--Write a String to a File

The LOG function writes the string argument to the specified file.

Syntax

The LOG function has the following format:

```
LOG(filename, source_string)
```

Parameters

The LOG function accepts the following parameters:

filename (string)

source_string (string)

Return Value

The LOG function returns a Boolean, which is always TRUE.

Remarks

Privileged: Yes

Example

```
Return_value=LOG("auditlog.txt", "Accessing Realm XXX")
Return_value=TRUE
```

LOOP Function--Call a Virtual Attribute in a Loop

The LOOP function calls the virtual attribute once for each possible number in the loop, starting and ending at the specified values. If the step value is specified, the numbers are incremented by the step value. If the virtual attribute returns a non-blank value, the value is included in the resulting set.

Syntax

The LOOP function has the following format:

```
LOOP(start_value, end_value, [step,] #virtual_user_attribute)
```

Parameters

The LOOP function accepts the following parameters:

start_value (number)

end_value (number)

step (number)

(Optional) The default is 1. Negative values are allowed.

#virtual_user_attribute (Named Expression)

Name of a defined virtual attribute. The virtual attribute can access the current loop counter by using a reference to %0.

Return Value

The LOOP function returns a set.

Examples

For these examples assume the virtual user attribute is #Padset := LPAD(%0, 2)

```
Return_set=LOOP(1, 5, #Padset)
Return_set="001^002^003^004^005"

Return_set=LOOP(1, 5, 2, #Padset)
Return_set="001^003^005"

Return_set=LOOP(5, 1, -1, #Padset)
Return_set="005^004^003^002^001"
```

More information:

Pasting (see page 713)
ENUMERATE Function--Test Set Elements (see page 755)
FILTER Function--Test Set Elements (see page 760)

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, [padding,]length)
```

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.

padding

(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.

length

The number of characters of the final string.

Return Value

The LPAD function returns a string.

Examples

Result_value=LPAD('Juan', 5)
Result_value=' Juan'

Result_value=LPAD('Juan', 'X', 5)
Result_value= 'XJuan'

Result_value=LPAD('Juan', 'XY', 6)
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 745)
RPAD Function--Pad a String on the Right (see page 790)

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 792)

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 777)

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 768)
RIGHT Function--Retrieve Characters from a String (see page 789)

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 774)

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 764)
HOUR24 Function--Convert to Hour (see page 765)
SECOND Function--Return the Number of Seconds in a Date (see page 792)

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 753)
DOW Function--Return Day of Week (see page 754)
DOY Function--Return Day of Year (see page 755)
YEAR Function--Return the Year Component of a Numeric Date (see page 806)
YEAR4 Function--Return the Year Component of a Date (4 digits) (see page 807)

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 739)
ORBITS Function--Perform a Bitwise OR Operation (see page 783)
XORBITS Function--Perform a Bitwise XOR Operation (see page 806)

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 781)

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 781)

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 a Boolean Value (see page 743)
STRING Function--Convert to a String (see page 796)

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 739)
NOTBITS Function--Perform a Bitwise NOT (see page 780)
XORBITS Function--Perform a Bitwise XOR Operation (see page 806)

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 800)

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 747)
EXPLODEDN Function--Convert LDAP DN to Set (see page 759)
PARENTDN Function--Retrieve Parent in LDAP Tree (see page 784)
RELATIONDN Function--Compare Two Distinguished Names (see page 788)

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 768)
MID Function--Return Part of a String (see page 776)

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,[padding,]length)
```

Parameters

The RPAD function accepts the following parameters:

source_string (string)

This parameter can be a number; it is converted to a string.

padding (string)

(Optional)

length (number)

Return Value

The RPAD function returns a string.

Example

```
Return_value=RPAD('Juan', 5)
Return_value='Juan '

Return_value=RPAD('Juan', 'X', 5)
Return_value='JuanX'

Return_value=RPAD('Juan', 'XY', 6)
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 773)

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 778)

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 763)

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 a Boolean Value (see page 743)
NUMBER Function--Convert to a Numeric Value (see page 782)

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.

Example

Return_value=TRACE("Executing Code")
Return_value=TRUE

More information:

ERROR Function--Write Error Message to Console Log (see page 757)
INFO Function--Write INFO Message to Console Log (see page 766)
TRACE Function--Write Trace Entry to Console Log (see page 805)

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 768)
PCASE Function--Convert a String to Proper Case (see page 784)

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 803)

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 803)

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
```

TRACE Function--Write Trace Entry 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 739)
NOTBITS Function--Perform a Bitwise NOT (see page 780)
ORBITS Function--Perform a Bitwise OR Operation (see page 783)

YEAR Function--Return the Year Component of a Numeric Date

The YEAR function returns a number that indicates the year component of a date in seconds since January 1, 1970.

Syntax

The YEAR function has the following format:

```
YEAR(date_time)
```

Parameters

The YEAR function accepts the following parameter:

date_time (number)

The date represented in seconds.

Return Value

The YEAR function returns a number between 70 and 138.

YEAR4 Function--Return the Year Component of a Date (4 digits)

The YEAR4 function returns the four-digit year component of a date expressed in seconds since January 1, 1970.

Syntax

The YEAR4 function has the following format:

```
YEAR4(date_time)
```

Parameters

The YEAR4 function accepts the following parameters:

date_time (number)

Return Value

The YEAR4 function returns a returns a number between 1970 and 2038.

Index