CA CA SiteMinder®

Programming Guide for Perl

r12.5

CA Technologies Product References

This document references the following CA Technologies products:

CA CA SiteMinder®

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Contents

Chapter 1: Perl Scripting Overview

This section contains the following topics:

About the SiteMinder Command Line Interface

The SiteMinder Command Line Interface (CLI) lets you perform SiteMinder tasks by running custom Perl scripts from the command line.

The scripting interface contains the following APIs:

- Agent API—Lets you test whether access control of protected resources is behaving according to policy definitions.
- Policy Management API—Lets you design and administer policy stores.

The Command Line Interface provides:

- A quick and light-weight alternative to the Administrative UI.
- An efficient way to perform Policy Server design and administrative tasks over multiple policy stores.
- The ability to migrate individual objects between policy stores.

The Command Line Interface lets you perform most, but not all, of the policy store operations you can perform through the Administrative UI.

More Information:

Command Line Interface Restrictions (see page 579)

Installation Path

By default, the SiteMinder Command Line Interface is installed in the following location:

<sm-ps-root>/CLI

<sm-ps-root> is the root directory where you installed your Policy Server software.

Perl Location

A complete version of Perl is installed along with the Policy Server. When you run scripts against the Command Line Interface, you should use the Perl interpreter that is installed with the Policy Server rather than any other Perl interpreter that might be on your system.

The installation program installs Perl in the following default location:

<sm-ps-root>/CLI/bin

If you have another version of Perl installed on your system, make sure that the Perl location shown above comes before any other Perl location in your system's PATH environment variable.

Where to Run Your Scripts

The Perl Agent and Policy Management APIs can be used on the following machines:

- Agent API—On the machine where a Policy Server is installed. The Agent API can access the local Policy Server or a remote Policy Server.
- Policy Management API—On the machine where a Policy Server is installed.

To run a script against these APIs, use the following command line syntax:

```
perl scriptname
```

Note: A script built with the Policy Management API must run as the same user who installed the Policy Server (for example, smuser on UNIX platforms).

CLI Example: Create a Policy Store Object

Suppose you are an administrator for the domain engineering. You want to create the realm documentation in that domain. Using the Administrative UI, you might take the following steps:

1. Log into the SiteMinder Administration software.
2. Right-click the domain engineering where you are adding the realm.
3. Click Create Realm and provide the following configuration information for the fields on the Resource tab:
   - Name: documentation
   - Description: Source files for manuals
   - Agent: agent1
   - Resource Filter: /mysite/docs/*
   - Authentication Scheme: Basic

   You are accepting all other defaults for the realm (including resource protection, which is enabled by default).
4. Click OK to confirm the creation of the new realm.

If you write a script to perform the same operation, it might look like this:

```
#Initialize the Policy Management API
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();

print "Step 1. Log in the admin and create an API session.\n";
$session = $policyapi->CreateSession("adminid", "adminpwd");

print "Step 2. Select the domain for the new realm.\n";
$domain=$session->GetDomain("engineering");

#Get the realm’s agent and authorization scheme info.\n";
$agent=$session->GetAgent("agent1");
$authscheme=$session->GetAuthScheme("Basic");
```

```
print "Step 3. Create and configure the realm.\n";
$realm=$domain->CreateRealm("documentation",
                            $agent,
                            $authscheme,
                            "Source files for manuals",
                            "/mysite/docs/*" );

print "Step 4. Confirm the creation of the realm.\n";
if ($realm == undef) {
         print "Realm creation failed.\n";
   }
   else {
         print "Realm creation succeeded.\n";
}
```

Note: Generally, policy store object names are case-sensitive. In the above example, the Basic authentication scheme and the engineering domain are case-sensitive. Further, agent names are always written to the policy store in lowercase. Existing agents must be referenced in lowercase in your scripts.

CLI Example: View and Set Individual Properties

Policy Management API objects (such as PolicyMgtRealm) provide a number of get/set methods that let you view and modify individual properties of objects in the policy store. You use these get/set methods to view and edit an object's properties just as you would use the property fields in the Administrative UI.

The following script modifies the resource filter property:

```
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");
$domain=$session->GetDomain("engineering");

$realm=$domain->GetRealm("documentation");
if($realm->ResourceFilter() eq "/mysite/docs/*") {
      $filter=$realm->ResourceFilter("/mysite/docs/*.doc");
}
```

```
if ($filter eq undef) {
      print "Error changing resource filter.\n";
   }
   else {
      print "Resource filter changed to: " . $filter . "\n";
}
```

Note the following general rules:

- When you pass an argument into a method, the method behaves as a setter method—for example:

  ```
  $realm->ResourceFilter("/mysite/docs/*.doc");
  ```

- When no argument is given, the method behaves as a getter method—for example:

  ```
  $filter=$realm->ResourceFilter();
  ```

- With get and set methods, the existing or new property value is returned.

Location of Sample Scripts

Sample scripts are installed in the following default location:

<sm-ps-root>/CA/siteminder/CLI/examples

Before using a sample script, be sure to change the values of the site-specific variables (such as administrator’s credentials and user-store location) that are defined at the beginning of the script.

Related Documentation

You can find additional information about Policy Server and agent operations in the following SiteMinder documents:

- *CA SiteMinder Policy Server Configuration Guide*
- *CA SiteMinder Policy Server Administration Guide*
- *CA SiteMinder Web Agent Configuration Guide*

Object Dependencies Poster

The poster *Scripting Interface for Perl: Object Dependencies* is included with SiteMinder. The poster illustrates the Perl objects that you need to create or retrieve before you can manipulate dependent objects. Each object is shown with all of its methods.

Chapter 2: Agent API

This section contains the following topics:

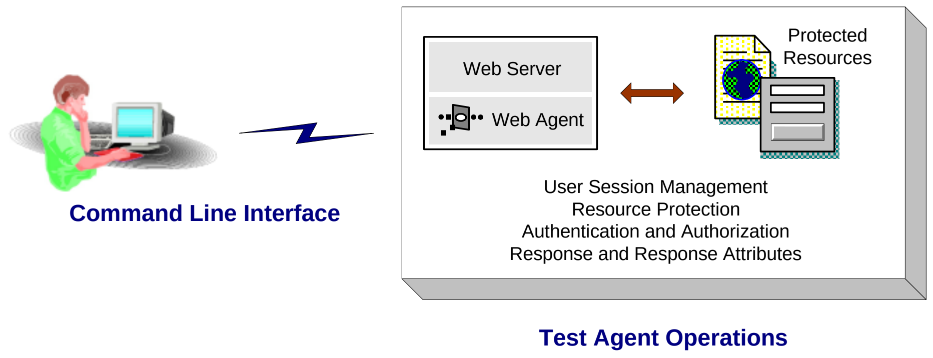
About Agents and the Agent API

A SiteMinder agent enforces access control policies provided by the Policy Server.

Agents act as gate keepers that protect resources from unauthorized users. When a user issues a request for a resource, the request is routed through the agent. The agent determines whether to accept or reject the request based on the information it receives from the Policy Server.

The Agent API (module Netegrity::AgentAPI) lets you write scripts that test the operation of your agents. The tasks you can perform with the Agent API include many of the tasks you can perform with the SiteMinder Test Tool. The Agent API also lets you view information about the user's session.

The following illustration shows some of the operations you can perform with the Agent API:



Write a Script against the Agent API

All scripts require certain common basic steps.

To write a script against the Agent API

1. Reference the Agent API at the beginning of your script:
2. Use the New() method to create an Agent API object:

   ```
   $agentapi=Netegrity::AgentAPI->New("agtName","shrdScrt");
   ```

   Note: Specifying the shared secret in the second argument creates a v4.x agent. Omitting the shared secret creates a v5.x or 6.x agent.

3. Provide configuration information about the Policy Server. The IP address is required, but you can accept the defaults for other arguments:

   ```
   $serverconfig = $agentapi->AddServerConfig($ipAddr);
   ```

4. Connect to the Policy Server:
   - With v4.x agents:

   ```
   $agentapi->Connect();
   ```

   - With v5.x and later agents:

   ```
   $agentapi->Connect("../../../../Program
   Files/Netegrity/webagent/config/smhost.conf");
   ```

Example: Verify the protection on a resource

After you have completed these basic steps, you can perform various agent operations. For example, you can see whether a specified resource is protected by the agent, and if so, you can find out the credentials that are required to access the resource:

```
$resource = $agentapi->GetResource("/companyXYZ/private/");
if($resource->IsProtected() == SM_AGENTAPI_YES) {
   print "\nAuthentication Type: ".$resource->GetAuthType();
}
else {
   print "The resource is not protected.";
}
```

Single Sign-on and the Agent API

In a single sign-on environment, a user who successfully authenticates through a given agent does not need to re-authenticate when accessing a realm protected by a different agent.

When a custom agent is involved in a single sign-on environment, the two agents must be in the same cookie domain—for example, xxx.domainname.com.

Single sign-on is made possible through a single sign-on cookie named SMSESSION. This cookie is created and written to the user's browser either by SiteMinder or by the custom agent.

The cookie's contents are retrieved from and written to the cookie in encrypted string form. The encrypted string is called a token.

The Agent API contains the following methods that allow custom agent scripts to share token information with standard SiteMinder Web Agents:

- CreateSSOToken(). After the user successfully logs in, the custom agent script passes information about the user to this method. The method creates a single sign-on token object from this user information and from session information returned from the login call.
- GetString(). After creating a token object, the custom agent script calls GetString() to retrieve the token as an encrypted string. The script then writes the token string to the SMSESSION cookie.
- Decode(). This method decrypts the token for the current token object and extracts the specified information. This method can also be used to update the last-access timestamp in the token.
- GetVersion(). Retrieves the SiteMinder version of the current token object.
- IsThirdParty(). Specifies whether the current token object was created by a custom (third-party) agent or a standard SiteMinder agent.

Single Sign-on Support for Custom Agents

You can create a single sign-on object when logging in through a custom agent, not when logging in through a standard SiteMinder Web Agent.

Here is the typical sequence of events in a single sign-on environment when the initial login is through the custom agent script:

- User logs in through the custom agent script.
- Custom agent script calls Login() to authenticate the user. The user is challenged for credentials.
- Custom agent script calls CreateSSOToken() and passes to it information about the user (user name, user DN, IP address of the requesting client). SiteMinder adds this information to a token object along with session information returned from the login call. SiteMinder also encrypts the information in the token.
- Custom agent script calls GetString() to retrieve the token information in encrypted string form.
- Custom agent script creates the SMSESSION cookie in the user's browser and writes the token string to the cookie.
- User requests a resource protected by a standard SiteMinder agent.
- The standard agent performs a login operation, which validates the user based on the information in the single sign-on cookie. The user is not challenged for credentials.

Single Sign-on Support for Standard Agents

Custom agent scripts created with the Perl Agent API can accept SMSESSION cookies created by a standard SiteMinder Web Agent.

However, standard SiteMinder Web Agents can only accept cookies created by a custom agent if the standard agent has been upgraded with the appropriate Siteminder Agent Quarterly Maintenance Release (QMR). For information about the QMR version required for each standard agent version, see the *SDK Release Notes*.

To enable a SiteMinder agent with the appropriate QMR upgrade to accept SMSESSION cookies created by a custom agent, the agent's Agent configuration file (LocalConfig.conf with IIS servers or WebAgent.conf with other servers) or central configuration object (for v5.x or later) must contain the following entry:

```
AcceptTPCookie="yes"
```

Session Information

Session information can consist of more than the session specification. Session information can include any information that the client application wants to associate with the user's session.

Application-defined session information consists of name-value pairs called *session variables*. For example, business logic, certificate information, and SAML assertions for affiliate operations can all be stored as session variables and bound to the session ID.

The package AgentSession provides the following methods for setting, retrieving, and deleting session variables, and for defining the parameters list:

- SetVariables()
- GetVariables()
- DelVariables()
- AddParameter()

Session variables are stored in a server-side database called the *session store*. The session store is managed by the Policy Server.

Advantages of Session Variables

When a client application uses session variables:

- Up to 4 KB of data can be stored for each session variable value.
- The session information persists across multiple Policy Servers. Centralizing session information about the server allows features such as cross-domain session management, including enforcing logout and idle timeout across different domains.

SiteMinder Support

The Command Line Interface supports:

- SiteMinder v4.x and v5.x agents running against SiteMinder v5.x Policy Servers
- SiteMinder v4.x and later agents agents running against SiteMinder v6.x Policy Servers.

The Agent API can be accessed on the machine where a SiteMinder Policy Server is installed. The Agent API can access the local Policy Server or a remote Policy Server.

Requirements for Using Session Variables

For a client application to use session variables, both of the following prerequisites must be met:

- The session store must be enabled in the Policy Server Management Console.
- During realm configuration in the Policy Server UI, Persistent Session must be selected for at least one of the realms to be accessed during the session. As soon as the user accesses a realm configured for persistent sessions, session variables can be used throughout the remainder of the session.

End of Session Cleanup

When the user logs out, the session ends. In the case of a persistent session, SiteMinder removes all session information, including any session variables, from the session store.

Objects and the Object Hierarchy

The Agent API contains a set of objects that let you perform agent operations.

The objects in the Agent API are hierarchical. Before an object can be used, it must be created or retrieved by a method in another object.

The following hierarchy shows the objects in the Agent API. Use it to find the methods in higher-level objects that create or retrieve other objects. For example, if you need an AgentResource object, refer to the object directly above it (AgentAPI) and choose the appropriate method.

- AgentAPI
  - AgentResource
  - AgentResponseAttr
  - AgentSvrCfg
  - AgentUser
    - AgentResponse
      - AgentResponseAttr
      - AgentSession
        - AgentResponseAttr
    - SSOToken

Chapter 3: CLI Agent API Methods

This section contains the following topics:

Agent Administration Methods

The following methods act on AgentAPI objects:

- AddServerConfig Method—Adds Policy Server Configurations to Agent API Object
- Connect Method—Establishes Connection between Agent API and Policy Server
- CreateBootstrapFile Method—Generates Bootstrap File for Connecting to Agent
- CreateUser Method—Creates a User Object
- Disconnect Method—Closes Connection between Agent and Policy Server
- DoManagement Method—Requests Agent Commands from Policy Server
- GetResource Method—Retrieves the Specified Resource
- IncrementRefCount Method—Increment the Reference Count
- New Method—Constructs the Agent API
- PrintDebugTrace Method—Outputs Trace Information to Console
- SetErrorCallback Method—Registers Subroutine that Processes Error Messages
- SetTraceCallback Method—Registers Subroutine that Processes Trace Messages

AddServerConfig Method—Adds Policy Server Configurations to Agent API Object

The AddServerConfig method adds one or more Policy Server configurations to the Agent API object. Before you can establish a connection between the Policy Server and the Agent API, you must call this method at least once.

Syntax

The AddServerConfig method has the following format:

```
Netegrity::AgentAPI->AddServerConfig(IPAddress[, minConn]
[, maxConn][, stepConn][, timeout][, azPort][, auPort]
[, acPort])
```

Parameters

The AddServerConfig method accepts the following parameters:

IPAddress (string)

Specifies the IP address of the Policy Server.

minConn (int)

(Optional) Specifies the minimum number of connections allowed.

maxConn (int)

(Optional) Specifies the maximum number of connections allowed.

stepConn (int)

(Optional) Specifies the number of connections to add when all existing connections are being used.

timeout (int)

(Optional) Specifies the time in seconds before the Agent API stops trying to connect to the Policy Server.

azPort (int)

(Optional) Specifies the port number for the authorization service.

auPort (int)

(Optional) Specifies the port number for the authentication service.

acPort (int)

(Optional) Specifies the port number for the accounting service.

Return Value

The AddServerConfig method returns the following value:

- AgentSvrCfg (object)

Remarks

The single-process Policy Server introduced in SiteMinder v6.0 combines the previously separate authentication, authorization, and accounting processes into one combined process whose requests go through one TCP port. As a result, the parameters azPort, auPort, and acPort all reference the same port number. The three arguments are maintained for backward compatibility.

Connect Method—Establishes Connection between Agent API and Policy Server

The Connect method establishes a connection between the Agent API and the Policy Server.

Syntax

The Connect method has the following format:

```
Netegrity::AgentAPI->Connect([bootFile])
```

Parameters

The Connect method accepts the following parameter:

bootFile (string)

(Optional) Specifies the name of the configuration file to be used for connecting to a v5.x agent.

Note: You can use an existing file such as SmHost.conf or create a new file. For more information, see the method AgentAPI->CreateBootstrapFile (see page 42).

Return Value

The Connect method returns one of the following values:

- SM_AGENTAPI_YES (value = 1)

  Specifies that the connection was successful.

- SM_AGENTAPI_FAILURE (value = -1)

  Specifies that the Agent API could not reach the Policy Server.

- SM_AGENTAPI_TIMEOUT (value = -2)

  Specifies that the method timed out.

- SM_AGENTAPI_NOCONNECTION (value = -3)

  Specifies that initialization failed.

CreateBootstrapFile Method—Generates Bootstrap File for Connecting to Agent

The CreateBootstrapFile method generates a bootstrap file that the Policy Server can use for connecting to a v5.x agent. The bootstrap file contains the settings specified by the method's parameters. Once the connection is made, the Policy Server can provide the remaining settings. SmHost.conf is an example of a bootstrap file.

Syntax

The CreateBootstrapFile method has the following format:

```
Netegrity::AgentAPI->CreateBootstrapFile(trustedHostName, ipAddress,
hostConfigName, sharedSecret, registrationDataFileName)
```

Parameters

The CreateBootstrapFile method accepts the following parameters:

trustedHostName (string)

Specifies the name of the trusted host.

ipAddress (string)

Specifies the IP address of the Policy Server.

hostConfigName (string)

Specifies the name of the host configuration object.

sharedSecret (string)

Specifies the value of the shared secret used by the host.

registrationDataFileName (string)

Specifies the name of the bootstrap file generated by the method.

Return Value

The CreateBootstrapFile method returns one of the following values:

- file_name (string)

  Specifies the name of the newly-created file.

- empty_string (string)

  Specifies that the method failed.

Example

The following code fragment is an example of how to use the CreateBootstrapFile method:

```
# A shared secret is not specified in the context of a v5.x agent.
# A v5.x agent by the specified name needs to exist in the policy
# store. Use Netegrity::AgentAPI.

$agentapi = Netegrity::AgentAPI->New($agentname);

# Create the bootstrap file.

$agentapi->CreateBootstrapFile($thostname, $ipaddr, $hconfname, $secret, $fname);

# Notice that in the v5.x context, the connect() method takes a filename.

$agentapi->Connect($fname);
```

CreateUser Method—Creates a User Object

The CreateUser method creates a user object containing the information passed to the method in the parameters. Perl uses this object to perform user operations, such as login.

Note: This method does not create a new user in the user directory.

Syntax

The CreateUser method has the following format:

```
Netegrity::AgentAPI->CreateUser(Username, Password[, cert][, certLength])
```

Parameters

The CreateUser method accepts the following parameters:

Username (string)

Specifies the user's ID.

Password (string)

Specifies the user's password.

cert (string)

(Optional) Specifies an X.509 certificate for user authentication.

certLength (int)

(Optional) Specifies the length of the certificate.

Return Value

The CreateUser method returns one of the following values:

- AgentUser (object)
- undef

  Specifies that the method failed.

Disconnect Method—Closes Connection between Agent and Policy Server

The Disconnect method closes the connection between the Agent and the Policy Server.

Syntax

The Disconnect method has the following format:

```
Netegrity::AgentAPI->Disconnect()
```

Parameters

The Disconnect method accepts no parameters.

Return Value

The Disconnect method returns one of the following values:

- SM_AGENT_SUCCESS (value = 0)

  Specifies that the connection was closed successfully.
- SM_AGENT_FAILURE (value = 1)

  Specifies that the connection was not successfully closed.

DoManagement Method—Requests Agent Commands from Policy Server

The DoManagement method requests the list of Agent commands pending from the Policy Server and returns them in an array. To access the commands in the array, call the AgentResponseAttr->GetID method.

Syntax

The DoManagement method has the following format:

```
Netegrity::AgentAPI->DoManagement()
```

Parameters

The DoManagement method accepts no parameters.

Return Value

The DoManagement method returns one of the following values:

- AgentResponseAttr (array)
- undef

  Specifies that there are no pending Agent commands.

GetResource Method—Retrieves the Specified Resource

The GetResource method retrieves the specified resource.

Syntax

The GetResource method has the following format:

```
Netegrity::AgentAPI->GetResource(resName[, action][, clientIP])
```

Parameters

The GetResource method accepts the following parameters:

resName (string)

Specifies the name of the resource to retrieve.

action (string)

(Optional) Specifies the HTTP action to perform (for example, GET, POST, or PUT).

clientIP (string)

(Optional) Specifies the client's IP address.

Return Value

The GetResource method returns the following value:

- AgentResource (object)

IncrementRefCount Method—Increment the Reference Count

You can call the IncrementRefCount method when the Agent API is operating in multi-threaded mode. After calling the Perl function fork, for example, you can call this method within the child process.

Syntax

The IncrementRefCount method has the following format:

```
Netegrity::AgentAPI->IncrementRefCount()
```

Parameters

The IncrementRefCount method accepts no parameters.

Return Value

The IncrementRefCount method does not return a value.

New Method—Constructs the Agent API

The New method constructs the Agent API object and must be called before the Agent API object can be used.

Syntax

The New method has the following format:

```
Netegrity::AgentAPI->New(agentName[, sharedSecret][, failover])
```

Parameters

The New method accepts the following parameters:

pagentName (string)

Specifies the name of the agent as it appears in the policy store.

Note: The agent name is not case-sensitive.

psharedSecret (string)

(Optional) Specifies the shared secret as it appears in the policy store.

Note: If you provide a value for the shared secret, a v4.x agent object is created. If you do not provide a value for the shared secret, a v5.x agent object is created. The shared secret *is* case-sensitive.

failover (int)

(Optional) Specifies whether to enable or disable failover operation:

- value = -1

  Specifies enabling failover operation.

- value = 0

  Specifies disabling failover operation.

Return Value

The New method returns the following value:

- AgentAPI (object)

PrintDebugTrace Method—Outputs Trace Information to Console

The PrintDebugTrace method enables or disables the output of error or trace information to the console and returns the value of 1 or 0, respectively.

Syntax

The PrintDebugTrace method has the following format:

```
Netegrity::AgentAPI->PrintDebugTrace([debugFlag])
```

Parameters

The PrintDebugTrace method accepts the following parameter:

debugFlag (int)

(Optional) Specifies enabling or disabling output of error or trace information to the console:

- value = 1

  Specifies enabling output.

- value = 0

  Specifies disabling output.

Return Value

The PrintDebugTrace method returns one of the following integer values:

- value = 1

  Specifies that output is enabled.

- value = 0

  Specifies that output is disabled.

ProcessAuthResponse Method--Extracts Information from a Login of Change Password

The ProcessAuthResponse method extracts pertinent information returned from a login or change password.

Syntax

The ProcessAuthResponse method has the following format:

```
Netegrity::AgentAPI->GetResource(UserCreds)
```

Parameters

The ProcessAuthResponse method accepts the following parameter:

UserCreds (UserCredentials)

Specifies the user credentials from a login or change password call

Return Value

The ProcessAuthResponse method returns the pertinent information as a string.

SetErrorCallback Method—Registers Subroutine that Processes Error Messages

The SetErrorCallback method registers a user-defined Perl subroutine that processes SiteMinder error messages. If multiple subroutines are registered, SiteMinder calls the most recently registered subroutine.

Syntax

The SetErrorCallback method has the following format:

```
Netegrity::AgentAPI->SetErrorCallback(subref)
```

Parameters

The SetErrorCallback method accepts the following parameter:

subref (string)

Specifies the name of the Perl subroutine or a reference to the subroutine:

- name

  Example:

  ```
  $agentapi->SetErrorCallback("CustomErrorHandler");
  ```

- reference

  Example:

  ```
  $agentapi->SetErrorCallback(\&CustomErrorHandler);
  ```

Return Value

The SetErrorCallback method returns one of the following integer values:

- value = 1

  Specifies that registration was successful.

- value = 0

  Specifies that registration failed.

SetTraceCallback Method—Registers Subroutine that Processes Trace Messages

The SetTraceCallback method registers a user-defined Perl subroutine that processes SiteMinder trace messages. If multiple subroutines are registered, SiteMinder calls the most recently registered subroutine.

Syntax

The SetTraceCallback method has the following format:

```
Netegrity::AgentAPI->SetTraceCallback(subref[, mode][, delim]
[, configFileName])
```

Parameters

The SetTraceCallback method accepts the following parameters:

subref (string)

Specifies the name of the Perl subroutine or a reference to the subroutine:

- name

  Example:

  ```
  $agentapi->SetErrorCallback("CustomTraceHandler");
  ```

- reference

  Example:

  ```
  $agentapi->SetErrorCallback(\&CustomTraceHandler);
  ```

mode (string)

(Optional) Specifies an output format for the trace messages:

- default

  Specifies fields enclosed by square brackets.

- fixed

  Specifies fixed-width fields.

- delim

  Specifies using a character to delimit the fields.

- xml

  Specifies fields enclosed by XML-like tags.

delim (string)

(Optional) Specifies the character to use as a delimiter when mode is set to delim.

configFileName (string)

(Optional) Specifies the name of the configuration file that specifies the data to be included in the trace.

Default: If no filename is specified, default settings are used.

Return Value

The SetTraceCallback method returns one of the following integer values:

- value = 1

  Specifies that registration was successful.

- value = 0

  Specifies that registration failed.

Resource Methods

The following methods act on AgentResource objects:

- GetAuthType Method—Retrieves the Type of Credentials Required
- IsProtected Method—Checks whether SiteMinder Is Protecting Resource

GetAuthType Method—Retrieves the Type of Credentials Required

The GetAuthType method retrieves the type of credentials required for successful login to the resource or realm.

Syntax

The GetAuthType method has the following format:

```
Netegrity::AgentResource->GetAuthType()
```

Parameters

The GetAuthType method accepts no parameters.

Return Value

The GetAuthType method returns one of the following values:

- Sm_AuthApi_Cred_None (value = 0x00)

  Specifies that no credentials are required.

  Note: This authorization type is used for anonymous realms.

- Sm_AuthApi_Cred_Basic (value = 0x01)

  Specifies that a username and password are required.

- Sm_AuthApi_Cred_Digest (value = 0x02)

  Specifies that the username and password must be exchanged using the digest protocol.

- Sm_AuthApi_Cred_X509Cert (value = 0x04)

  Specifies that a full X.509 client certificate is required.

- Sm_AuthApi_Cred_X509CertUserDN (value = 0x08)

  Specifies that the user DN from an X.509 client certificate is required.

- Sm_AuthApi_Cred_X509CertIssuerDN (value = 0x10)

  Specifies that the issuer DN from an X.509 client certificate is required.

- Sm_AuthApi_Cred_CertOrBasic (value = 0x20)

  Specifies that a certificate is required, if available.

  Note: If a certificate is not available, a username and password are required.

- Sm_AuthApi_Cred_NTChalResp (value = 0x40)

  Specifies that the username and password must be exchanged using the NT challenge/response protocol.

- Sm_AuthApi_Cred_CertOrForm (value = 0x80)

  Specifies that either an X.509 certificate or a forms-based authentication scheme is required.

- Sm_AuthApi_Cred_SSLRequired (value = 0x01000000)

  Specifies that an SSL connection is required.

- Sm_AuthApi_Cred_FormRequired (value = 0x02000000)

  Specifies that the user must be redirected to an HTML form.

- Sm_AuthApi_Cred_AllowSaveCreds (value = 0x04000000)

  Specifies that the credentials can be saved for 30 days

  Note: When credentials are saved, users are not required to re-enter them each time they access a protected resource.

- Sm_AuthApi_Cred_PreserveSessionID (value = 0x08000000)

  Specifies that the Session ID be preserved for as long as the current session is valid.

- Sm_AuthApi_Cred_DoNotChallenge (value = 0x10000000)

  Specifies that the user not be challenged for credentials.

- undef

  Specifies that the resource is not protected.

IsProtected Method—Checks whether SiteMinder Is Protecting Resource

The IsProtected method checks whether SiteMinder is protecting the defined resource.

Syntax

The IsProtected method has the following format:

```
Netegrity::AgentResource->IsProtected()
```

Parameters

The IsProtected method accepts no parameters.

Return Value

The IsProtected method returns one of the following values:

- SM_AGENTAPI_YES (value = 1)

  Specifies that the resource is protected.

- SM_AGENTAPI_NO (value = 2)

  Specifies that the resource is not protected.

- SM_AGENTAPI_FAILURE (value = -1)

  Specifies that the Policy Server could not be reached.

- SM_AGENTAPI_TIMEOUT (value = -2)

  Specifies that the method timed out.

- SM_AGENTAPI_NOCONNECTION (value = -3)

  Specifies that initialization failed.

Response Methods

The following methods act on AgentResponse objects:

- GetAttributes Method—Retrieves List of Available Response Attributes
- GetSession Method—Retrieves the Session from the Response

GetAttributes Method—Retrieves List of Available Response Attributes

The GetAttributes method retrieves the list of available response attributes and returns them in an array. Use this method after successful calls to AgentUser->Login or AgentUser->GetResponse. To identify the types of response attributes in the array, call the AgentResponseAttr->GetID method.

Syntax

The GetAttributes method has the following format:

```
Netegrity::AgentResponse->GetAttributes()
```

Parameters

The GetAttributes method accepts no parameters.

Return Value

The GetAttributes method returns one of the following values:

- AgentResponseAttr (array)
- undef

  Specifies that the method failed.

GetSession Method—Retrieves the Session from the Response

The GetSession method retrieves the session object from the response.

Syntax

The GetSession method has the following format:

```
Netegrity::AgentResponse->GetSession()
```

Parameters

The GetSession method accepts no parameters.

Return Value

The GetSession method returns one of the following values:

- AgentSession (object)
- undef

  Specifies that the method failed.

Response Attribute Methods

The following methods act on AgentResponseAttr objects:

- GetFlags Method—Retrieves Response Attribute's Flags
- GetID Method—Retrieves Response Attribute's ID or Agent Command's ID
- GetName Method—Retrieves Response Attribute's Name
- GetTTL Method—Retrieves Response Attribute's TTL Value
- GetValue Method—Retrieves Response Attribute's Value

GetFlags Method—Retrieves Response Attribute's Flags

The GetFlags method retrieves the response attribute's flags.

Syntax

The GetFlags method has the following format:

```
Netegrity::AgentResponseAttr->GetFlags()
```

Parameters

The GetFlags method accepts no parameters.

Return Value

The GetFlags method returns the following value:

- existing_response_attribute_flags

GetID Method—Retrieves Response Attribute's ID or Agent Command's ID

The GetID method retrieves a response attribute ID or an agent command ID.

Syntax

The GetID method has the following format:

```
Netegrity::AgentResponseAttr->GetID()
```

Parameters

The GetID method accepts no parameters.

Return Value

The GetID method returns a response attribute ID after AgentResponse->GetAttributes is called or an agent command ID after AgentAPI->DoManagement is called. In either case, the return value's type is long.

- AgentResponse->GetAttributes retrieves a list of response attributes and returns them in an array. When GetID is called after AgentResponse->GetAttributes is called, GetID returns one of the following response attribute IDs (long):
  - SM_AGENTAPI_ATTR_AUTH_DIR_OID (value = 151)

    Specifies the internal SiteMinder object ID for the user directory where the user was authenticated.
  - SM_AGENTAPI_ATTR_USERUNIVERSALID (value = 152)

    Specifies the user's universal ID.
  - SM_AGENTAPI_ATTR_IDENTITYSPEC (value = 156)

    Specifies the user's identity ticket.

    Note: This value is returned if user tracking is enabled.
  - SM_AGENTAPI_ATTR_SESSIONDRIFT (value = 167)

    Specifies the maximum time in minutes, during which actual session data is validated against session data stored in a cookie.
  - SM_AGENTAPI_ATTR_AUTH_DIR_NAME (value = 213)

    Specifies the directory name as it appears in the Name field of the SiteMinder User Directory dialog.
  - SM_AGENTAPI_ATTR_AUTH_DIR_SERVER (value = 214)

    Specifies the server specification as it appears in the Server field of the SiteMinder User Directory dialog.
  - SM_AGENTAPI_ATTR_AUTH_DIR_NAMESPACE (value = 215)

    Specifies the user directory's namespace: LDAP, AD, WinNT, or ODBC.
  - SM_AGENTAPI_ATTR_USERMSG (value = 216)

    Specifies the text presented to the user following an authentication attempt.

    Note: This text could be an authentication challenge or a reason why authentication failed.
  - SM_AGENTAPI_ATTR_USERDN (value = 218)

    Specifies the user's distinguished name.

- AgentAPI->DoManagement retrieves a list of agent commands pending from the Policy Server and returns them in an array. When GetID is called after AgentAPI->DoManagement is called, GetID returns one of the following agent command IDs (long):
  - SM_AGENTAPI_AFFILIATE_KEY_UPDATE (value = 189)

    Instructs the agent to update the name of the affiliate agent.
  - SM_AGENTAPI_AGENT_KEY_UPDATE_NEXT (value = 190)

    Instructs the agent to update its "next" agent key.

    Note: The encrypted value contains 24 bytes of binary data.
  - SM_AGENTAPI_AGENT_KEY_UPDATE_LAST (value = 191)

    Instructs the agent to update its "last" agent key.

    Note: The encrypted value contains 24 bytes of binary data.
  - SM_AGENTAPI_AGENT_KEY_UPDATE_CURRENT (value = 192)

    Instructs the agent to update its "current" agent key.

    Note: The encrypted value contains 24 bytes of binary data.
  - SM_AGENTAPI_AGENT_KEY_UPDATE_PERSISTENT (value = 193)

    Instructs the agent to update its static (persistent) agent key.

    Note: The encrypted value contains 24 bytes of binary data.
  - SM_AGENTAPI_CACHE_FLUSH_ALL (value = 194)

    Instructs the agent to flush all information in its caches.
  - SM_AGENTAPI_CACHE_FLUSH_ALL_USERS (value = 195)

    Instructs the agent to flush all user information stored in its caches.
  - SM_AGENTAPI_CACHE_FLUSH_THIS_USER (value = 196)

    Instructs the agent to flush all cache information for a given user.
  - SM_AGENTAPI_CACHE_FLUSH_ALL_REALMS (value = 197)

    Instructs the agent to flush all resource information stored in its caches.
  - SM_AGENTAPI_CACHE_FLUSH_THIS_REALM (value = 198)

    Instructs the agent to flush all resource information for a given realm.

GetName Method—Retrieves Response Attribute's Name

The GetName method retrieves the name of the response attribute.

Syntax

The GetName method has the following format:

```
Netegrity::AgentResponseAttr->GetName()
```

Parameters

The GetName method accepts no parameters.

Return Value

The GetName method returns the following value:

- response_attribute_name (string)

GetTTL Method—Retrieves Response Attribute's TTL Value

The GetTTL method retrieves the response attribute's Time To Live (TTL) value.

Syntax

The GetTTL method has the following format:

```
Netegrity::AgentResponseAttr->GetTTL()
```

Parameters

The GetTTL method accepts no parameters.

Return Value

The GetTTL method returns the following value:

- Time_To_Live_value (long)

GetValue Method—Retrieves Response Attribute's Value

The GetValue method retrieves the response attribute's value.

Syntax

The GetValue method has the following format:

```
Netegrity::AgentResponseAttr->GetValue()
```

Parameters

The GetValue method accepts no parameters.

Return Value

The GetValue method returns the following value:

- response_attribute_value (string)

Server Configuration Method

The following method acts on AgentSvrCfg objects:

- IPAddress Method—Sets or Retrieves Policy Server's IP Address

IPAddress Method—Sets or Retrieves Policy Server's IP Address

The IPAddress method sets or retrieves the IP address of the Policy Server.

Syntax

The IPAddress method has the following format:

```
Netegrity::AgentSvrCfg->IPAddress([IPAddress])
```

Parameters

The IPAddress method accepts the following parameter:

IPAddress (string)

(Optional) Specifies the IP address of the Policy Server.

Return Value

The IPAddress method returns the following value:

- IP_address (string)

  Specifies the IP address of the Policy Server.

Session Methods

The following methods act on AgentSession objects:

- AddParameter Method—Adds Session Variable Name-Value Pair to Parameters List
- DelVariables Method—Deletes Session Variables from Session Store
- GetID Method—Retrieves the Session ID
- GetReason Method—Retrieves the Session's Reason ID
- GetSpec Method—Retrieves the Encrypted Session Specification
- GetVariables Method—Retrieves Session Variables from Session Store
- IdleTimeout Method—Retrieves Session's Idle Timeout Value
- MaxTimeout Method—Retrieves Session's Maximum Timeout Value
- SetVariables Method—Writes Session Variables to Session Store

AddParameter Method—Adds Session Variable Name-Value Pair to Parameters List

The AddParameter method adds a session variable name-value pair to a parameters list. Session variables contain data about the client application, such as business logic, certificate information, and SAML assertions for Affiliate applications. This data is bound to a user session.

Syntax

The AddParameter method has the following format:

```
Netegrity::AgentSession->AddParameter(varName [,varValue][, varFlag])
```

Parameters

The AddParameter method accepts the following parameters:

varName (string)

Specifies the name of the variable to add to the parameter list.

Limit: Maximum length is 255 characters.

varValue (string)

(Optional) Specifies the value of the variable to be added to the parameter list.

Limit: Maximum length is determined by the target data store.

varFlag (int)

(Optional) Specifies whether the GetVariables method deletes the variable name-value pair from the session store:

- 0 (zero)

  Specifies that GetVariables retrieves the variable name-value pair from the session store, but does not delete it.

- 1 (one)

  Specifies that GetVariables retrieves the variable name-value pair from the session store and then deletes it.

Return Value

The AddParameter method does not return a value.

Remarks

You can manage the name-value pairs in the parameters list by calling these session variable methods:

- AgentSession->DelVariables
- AgentSession->GetVariables
- AgentSession->SetVariables

To manage multiple variables, call AddParameter once for each variable before calling the AgentSession->DelVariables, AgentSession->GetVariables, and AgentSession->SetVariables methods. While AddParameter adds variables to the parameters list, AgentSession->DelVariables, AgentSession->GetVariables, and AgentSession->SetVariables clear the parameters list.

Before you can use the session variable methods, the following conditions must be met:

- A persistent session must be created in at least one realm.
- The SiteMinder session store must be enabled.

Note: For more information, see the *Policy Server Administration Guide* and the *Policy Server Configuration Guide*.

DelVariables Method—Deletes Session Variables from Session Store

The DelVariables method deletes the session variables in the parameters list from the session store. Call AgentSession->AddParameter first to specify the variables in the parameters list.

Syntax

The DelVariables method has the following format:

```
Netegrity::AgentSession->DelVariables()
```

Parameters

The DelVariables method accepts no parameters.

Return Value

The DelVariables method does not return a value.

Remarks

The DelVariables method clears the parameters list.

GetID Method—Retrieves the Session ID

The GetID method retrieves the session ID.

Syntax

The GetID method has the following format:

```
Netegrity::AgentSession->GetID()
```

Parameters

The GetID method accepts no parameters.

Return Value

The GetID method returns the following value:

- session_ID (string)

GetReason Method—Retrieves the Session's Reason ID

The GetReason method retrieves the session's reason ID. The reason ID specifies a reason for a failed authentication or authorization.

Syntax

The GetReason method has the following format:

```
Netegrity::AgentSession->GetReason()
```

Parameters

The GetReason method accepts no parameters.

Return Value

The GetReason method returns one of the following values (long):

- Sm_Api_Reason_None (value = 0)
- Sm_Api_Reason_PwMustChange (value = 1)
- Sm_Api_Reason_InvalidSession (value = 2)
- Sm_Api_Reason_RevokedSession (value = 3)
- Sm_Api_Reason_ExpiredSession (value = 4)

- Sm_Api_Reason_AuthLevelTooLow (value = 5)
- Sm_Api_Reason_UnknownUser (value = 6)
- Sm_Api_Reason_UserDisabled (value = 7)
- Sm_Api_Reason_InvalidSessionId (value = 8)
- Sm_Api_Reason_InvalidSessionIp (value = 9)
- Sm_Api_Reason_CertificateRevoked (value = 10)
- Sm_Api_Reason_CRLOutOfDate (value = 11)
- Sm_Api_Reason_CertRevokedKeyCompromised (value = 12)
- Sm_Api_Reason_CertRevokedAffiliationChange (value = 13)
- Sm_Api_Reason_CertOnHold (value = 14)
- Sm_Api_Reason_TokenCardChallenge (value = 15)
- Sm_Api_Reason_ImpersonatedUserNotInDir (value = 16)
- Sm_Api_Reason_Anonymous (value = 17)
- Sm_Api_Reason_PwWillExpire (value = 18)
- Sm_Api_Reason_PwExpired (value = 19)
- Sm_Api_Reason_ImmedPWChangeRequired (value = 20)
- Sm_Api_Reason_PWChangeFailed (value = 21)
- Sm_Api_Reason_BadPWChange (value = 22)
- Sm_Api_Reason_PWChangeAccepted (value = 23)
- Sm_Api_Reason_ExcessiveFailedLoginAttempts (value = 24)
- Sm_Api_Reason_AccountInactivity (value = 25)
- Sm_Api_Reason_NoRedirectConfigured (value = 26)
- Sm_Api_Reason_ErrorMessageIsRedirect (value = 27)
- Sm_Api_Reason_Next_Tokencode (value = 28)
- Sm_Api_Reason_New_PIN_Select (value = 29)
- Sm_Api_Reason_New_PIN_Sys_Tokencode (value = 30)
- Sm_Api_Reason_New_User_PIN_Tokencode (value = 31)
- Sm_Api_Reason_New_PIN_Accepted (value = 32)
- Sm_Api_Reason_Guest (value = 33)
- Sm_Api_Reason_PWSelfChange (value = 34)

- Sm_Api_Reason_ServerException (value = 35)
- Sm_Api_Reason_UnknownScheme (value = 36)
- Sm_Api_Reason_UnsupportedScheme (value = 37)
- Sm_Api_Reason_Misconfigured (value = 38)
- Sm_Api_Reason_BufferOverflow (value = 39)
- Sm_Api_Reason_SetPersistentSessionFailed (value = 40)

GetSpec Method—Retrieves the Encrypted Session Specification

The GetSpec method retrieves the encrypted session specification. The session specification includes information about the user session, such as the session start time, timeout parameters, and the user's distinguished name. You can use the session specification to identify a session across multiple Web sites, as you would for single sign-on applications.

Syntax

The GetSpec method has the following format:

```
Netegrity::AgentSession->GetSpec()
```

Parameters

The GetSpec method accepts no parameters.

Return Value

The GetSpec method returns the following value:

- encrypted_session_specification (string)

Remarks

Where the session specification is stored depends on the type of session:

- Non-persistent Sessions

  The session specification is stored in a cookie.
- Persistent Sessions

  The session specification is stored in a session store database and optionally in a cookie on the client.

Note: The session ticket is used as an index to the actual user session data stored in the Web agent's cache.

GetVariables Method—Retrieves Session Variables from Session Store

The GetVariables method retrieves the session variables in the parameters list from the session store. Call AgentSession->AddParameter first to specify the variables in the parameters list. To delete a variable from the session store after it is retrieved, call AgentSession->AddParameter with *varFlag* set to 1 (one).

Syntax

The GetVariables method has the following format:

```
Netegrity::AgentSession->GetVariables()
```

Parameters

The GetVariables method accepts no parameters.

Return Value

The GetVariables method returns the following value:

- AgentResponseAttr (array)

Note: Some session variables cannot be retrieved. To check for these, call AgentResponseAttr->GetFlags for the associated AgentResponseAttr object and test for the following return value:

SM_AGENTAPI_RESPATTR_FLAGS_UNRESOLVED (value = 2)

Remarks

The GetVariables method clears the parameters list.

IdleTimeout Method—Retrieves Session's Idle Timeout Value

The IdleTimeout method retrieves the session's idle timeout value in seconds. The idle timeout value is the maximum length of time a user session can be *inactive*. When this time is exceeded, the session is no longer valid and the user must re-authenticate.

Syntax

The IdleTimeout method has the following format:

```
Netegrity::AgentSession->IdleTimeout()
```

Parameters

The IdleTimeout method accepts no parameters.

Return Value

The IdleTimeout method returns the following value:

- idle_timeout_value (long)

MaxTimeout Method—Retrieves Session's Maximum Timeout Value

The MaxTimeout method retrieves the session's maximum timeout value in seconds. The maximum timeout value is the length of time a user session can be *active*. When this time is exceeded, the user must re-authenticate.

Syntax

The MaxTimeout method has the following format:

```
Netegrity::AgentSession->MaxTimeout()
```

Parameters

The MaxTimeout method accepts no parameters.

Return Value

The MaxTimeout method returns the following value:

- maximum_timeout_value (long)

SetVariables Method—Writes Session Variables to Session Store

The SetVariables method writes the session variables in the parameters list to the session store. Call AgentSession->AddParameter first to specify the variables in the list. If a variable exists in the session store, its value is updated to the new value. If the variable does not exist, a new variable is created.

Syntax

The SetVariables method has the following format:

```
Netegrity::AgentSession->SetVariables()
```

Parameters

The SetVariables method accepts no parameters.

Return Value

The SetVariables method does not return a value.

Remarks

The SetVariables method clears the parameters list.

Single Sign-on Token Methods

The following methods act on SSOToken objects:

- Decode Method—Decodes a Single Sign-on Token
- GetString Method—Retrieves String Representation of SSO Token Object
- GetVersion Method—Retrieves SiteMinder Version of SSO Token
- IsThirdParty Method—Determines Whether the Token Is Custom

Decode Method—Decodes a Single Sign-on Token

The Decode method decodes a single sign-on token and returns a subset of its attributes. In addition, you have the option of updating the token's last-accessed timestamp by passing a non-zero value to this method. To retrieve the updated token in string format, call SSOToken->GetString and write the token string to the SMSESSION cookie.

Syntax

The Decode method has the following format:

```
Netegrity::SSOToken->Decode([update])
```

Parameters

The Decode method accepts the following parameter:

update (int)

(Optional) Specifies whether an updated token is requested:

- value = non-zero

  Specifies that an updated token is requested.

- value = 0 (default)

  Specifies that an updated token is not requested.

Return Value

The Decode method returns one of the following values:

- an array of attributes containing a subset of the following:
  - ATTR_CLIENTIP

    Specifies the IP address of the machine where the user initiated a request for a protected resource.
  - ATTR_DEVICENAME

    Specifies the name of the agent that is decoding the token.
  - ATTR_IDLESESSIONTIMEOUT

    Specifies the maximum idle time for a session.
  - ATTR_LASTSESSIONTIME

    Specifies the time when the Policy Server was last accessed within the session.
  - ATTR_MAXSESSIONTIMEOUT

    Specifies the maximum time that a session can be active.
  - ATTR_SESSIONID

    Specifies the session ID returned from the login call.
  - ATTR_SESSIONSPEC

    Specifies the session specification returned from the login call.
  - ATTR_STARTSESSIONTIME

    Specifies when the session started after a successful login.
  - ATTR_USERDN

    Specifies the user's distinguished name.
  - ATTR_USERNAME

    Specifies the user's name.
- undef

  Specifies that the method failed.

Remarks

To create a single sign-on object, call AgentUser->CreateSSOToken.

GetString Method—Retrieves String Representation of SSO Token Object

The GetString method retrieves the string representation of a single sign-on token object. After calling GetString, you can write the token string to the SMSESSION cookie.

Syntax

The GetString method has the following format:

```
Netegrity::SSOToken->GetString()
```

Parameters

The GetString method accepts no parameters.

Return Value

The GetString method returns the following value:

- SSO_token (string)

Remarks

You can call GetString after creating a single sign-on token object with CreateSSOToken. You can also call GetString after updating the token's last-accessed timestamp with Decode.

GetVersion Method—Retrieves SiteMinder Version of SSO Token

The GetVersion method retrieves the SiteMinder version of the single sign-on token.

Syntax

The GetVersion method has the following format:

```
Netegrity::SSOToken->GetVersion()
```

Parameters

The GetVersion method accepts no parameters.

Return Value

The GetVersion method returns the following value:

- version (int)

  Specifies the SiteMinder version of the single sign-on token.

IsThirdParty Method—Determines Whether the Token Is Custom

The IsThirdParty method determines whether the token was originally produced by a custom (or third-party) agent and has not yet been updated by a standard SiteMinder agent.

Syntax

The IsThirdParty method has the following format:

```
Netegrity::SSOToken->IsThirdParty()
```

Parameters

The IsThirdParty method accepts no parameters.

Return Value

The IsThirdParty method returns one of the following integer values:

- value = non-zero

  Specifies that the token was originally produced by a custom agent and has not yet been updated by a standard SiteMinder agent.

- value = 0

  Specifies that the token was not produced by a custom agent or has been updated by a standard SiteMinder agent.

User Methods

The following methods act on AgentUser objects:

- Audit Method—Audits Authorizations Performed out of Agent Cache
- Certificate Method—Sets or Retrieves User's X.509 Certificate
- CertificateFile Method—Sets or Retrieves User's X.509 Certificate Using File
- CreateSSOToken Method—Creates Single Sign-on Token Object
- CustomData Method—Sets or Retrieves Custom Authentication Data
- FormData Method—Sets or Retrieves HTML Forms-based Authentication Data
- GetResponse Method—Returns Response After IsAuthorized or Login
- Impersonate Method—Allows One User to Impersonate Another
- IsAuthorized Method—Determines Whether User Is Authorized
- Login Method—Performs Session Login and Validation
- Logout Method—Logs the User out of the Session
- Name Method—Sets or Retrieves the User's Username
- Password Method—Sets or Retrieves the User's Password
- Validate Method—Validates a Session Specification

Audit Method—Audits Authorizations Performed out of Agent Cache

The Audit method audits authorizations performed out of the agent cache.

Syntax

The Audit method has the following format:

```
Netegrity::AgentUser->Audit()
```

Parameters

The Audit method accepts no parameters.

Return Value

The Audit method returns one of the following values:

- SM_AGENTAPI_YES (value = 1)

  Specifies that the audit was successful.

- SM_AGENTAPI_NO (value = 2)

  Specifies that the audit was not successful.

- SM_AGENTAPI_FAILURE (value = -1)

  Specifies that the Policy Server could not be reached.

- SM_AGENTAPI_TIMEOUT (value = -2)

  Specifies that the method timed out.

- SM_AGENTAPI_NOCONNECTION (value = -3)

  Specifies that initialization failed.

Certificate Method—Sets or Retrieves User's X.509 Cerficate

The Certificate method sets or retrieves the user's X.509 certificate. This method only affects the certificate data associated with the current instance of the user object.

Syntax

The Certificate method has the following format:

```
Netegrity::AgentUser->Certificate([cert, certBinaryLen])
```

Parameters

The Certificate method accepts the following parameters:

cert (string)

(Optional) Specifies the certificate data to set.

certBinaryLen (int)

(Optional) Specifies the length of the certificate.

Return Value

The Certificate method returns one of the following values:

- ($String, $Length)

  Specifies the new or existing certificate's data and length.

- undef

  Specifies that the method failed.

CertificateFile Method—Sets or Retrieves User's X.509 Certificate Using File

The CertificateFile method sets or retrieves the user's X.509 certificate using the specified certificate file.

Syntax

The CertificateFile method has the following format:

```
Netegrity::AgentUser->CertificateFile([certFile[, format]])
```

Parameters

The CertificateFile method accepts the following parameters:

certFile (string)

> (Optional) Specifies the full path and file name of the certificate file.

format (string)

> (Optional) Specifies the format of the certificate file.
>
> **Default:** base64 encoded X.509 (value = 1)
>
> **Note:** The default is the only supported file format.

Return Value

The CertificateFile method returns the following value:

- ($String, $Length)

  Specifies the new or existing certificate's data and length.

CreateSSOToken Method—Creates Single Sign-on Token Object

The CreateSSOToken method creates a single sign-on token object from a valid user session. The token contains encrypted session and other information that a custom agent can share with a standard SiteMinder Web agent. Creating single sign-on between standard and custom agents requires that the agents be in the same domain. To create the single sign-on object, the user must be logged in to the custom agent, not the SiteMinder agent.

Syntax

The CreateSSOToken method has the following format:

```
Netegrity::AgentUser->CreateSSOToken(szDn, szName, szIP)
```

Parameters

The CreateSSOToken method accepts the following parameters:

szDn (string)

Specifies the user's distinguished name.

szName (string)

Specifies the user's name.

szIP (string)

Specifies the IP address of the machine, where the user initiates the request for a protected resource.

Return Value

The CreateSSOToken method returns the following value:

- SSOToken (object)

Remarks

To retrieve the token object in string format, use the GetString method and write the token string to the SMSESSION cookie. To decode the token and retrieve a subset of its attributes, use the Decode method.

CustomData Method—Sets or Retrieves Custom Authentication Data

The CustomData method sets or retrieves custom authentication data. This method is used to test authentication schemes that are based on the Custom Template. The format and content of custom authentication data are customer-defined according to the requirements of each Web site.

Syntax

The CustomData method has the following format:

```
Netegrity::AgentUser->CustomData([customData, length])
```

Parameters

The CustomData method accepts the following parameters:

customData (string)

(Optional) Specifies the custom authentication data to set.

length (int)

(Optional) Specifies the length of the custom authentication data.

Return Value

The CustomData method returns one of the following values:

- ($String, $Length)

  Specifies the new or existing custom authentication data and length.

- undef

  Specifies that the method failed.

FormData Method—Sets or Retrieves HTML Forms-based Authentication Data

The FormData method sets or retrieves HTML forms-based authentication data. This method is used to test authentication schemes that are based on the HTML Forms Template. The formData string consists of attribute name-value pairs separated by the ampersand (&) character.

Example:

```
"PASSWORD=$password1&email=$username1@mycompany.com"
```

Syntax

The FormData method has the following format:

```
Netegrity::AgentUser->FormData([formData])
```

Parameters

The FormData method accepts the following parameter:

formData (string)

(Optional) Specifies the HTML forms-based authentication data to set.

Return Value

The FormData method returns one of the following values:

- authentication_data

  Specifies the new or existing HTML forms-based authentication data.

- undef

  Specifies that the method failed.

GetResponse Method—Returns Response After IsAuthorized or Login

The GetResponse method returns a response after AgentUser->IsAuthorized or AgentUser->Login is called regardless of whether the user is authorized.

Syntax

The GetResponse method has the following format:

```
Netegrity::AgentUser->GetResponse()
```

Parameters

The GetResponse method accepts no parameters.

Return Value

The GetResponse method returns one of the following values:

- AgentResponse (object)
- undef

  Specifies that the method failed, because neither AgentUser->IsAuthorized or AgentUser->Login was called before calling GetResponse.

Impersonate Method—Allows One User to Impersonate Another

The Impersonate method allows one user to impersonate another user by logging in as that user. For example, a customer service representative can impersonate a customer to better understand a software problem that the customer is having.

Syntax

The Impersonate method has the following format:

```
Netegrity::AgentUser->Impersonate(username, resource)
```

Parameters

The Impersonate method accepts the following parameters:

username (string)

Specifies the ID of the user to impersonate.

resource (AgentResource object)

Specifies the resource to log in to.

Return Value

The Impersonate method returns one of the following values:

- SM_AGENTAPI_YES (value = 1)

  Specifies that the impersonation was successful.
- SM_AGENTAPI_NO (value = 2)

  Specifies that impersonation failed.
- SM_AGENTAPI_FAILURE (value = -1)

  Specifies that the operation failed.
- SM_AGENTAPI_TIMEOUT (value = -2)

  Specifies that the method timed out.
- SM_AGENTAPI_NOCONNECTION (value = -3)

  Specifies that initialization failed.

Remarks

The Impersonate method creates a new session without destroying the impersonator's original session. To end the impersonation session and restore the impersonator's original session, call AgentUser->Logout.

Only one user at a time can be impersonated. You cannot chain impersonation sessions.

Impersonation begins in a realm that is protected by the Impersonation Authorization Scheme. The impersonator must be authorized to impersonate users in the realm, and the user must be allowed to be impersonated in the realm.

For more information about user impersonation, see the *Policy Server Configuration Guide*.

IsAuthorized Method—Determines Whether User Is Authorized

The IsAuthorized method determines whether the user is authorized to perform the specified action on the specified resource. This method calls AgentUser->Login, if AgentUser->Login has not been called. After calling this method, call AgentUser->GetResponse.

Syntax

The IsAuthorized method has the following format:

```
Netegrity::AgentUser->IsAuthorized(resource[, clientIP][, transID])
```

Parameters

The IsAuthorized method accepts the following parameters:

resource (AgentResource object)

Specifies the resource to check.

clientIP (string)

(Optional) Specifies the client's IP address.

transID (string)

(Optional) Specifies the user-defined transaction ID that the agent uses to associate application activity with security activity.

Return Value

The IsAuthorized method returns one of the following values:

- SM_AGENTAPI_YES (value = 1)

  Specifies that the user is authorized.

- SM_AGENTAPI_NO (value = 2)

  Specifies that the user is not authorized.

- SM_AGENTAPI_FAILURE (value = -1)

  Specifies that the Policy Server could not be reached.

- SM_AGENTAPI_TIMEOUT (value = -2)

  Specifies that the method timed out.

- SM_AGENTAPI_NOCONNECTION (value = -3)

  Specifies that initialization was not done.

IsAuthorizedEx Method--Determines Whether User Is Authorized

The IsAuthorizedEx method determines whether the user is authorized to perform the specified action on the specified resource. This method calls AgentUser->Login if AgentUser->Login has not been called. After calling this method, call AgentUser->GetResponse.

Syntax

The IsAuthorizedEx method has the following format:

```
Netegrity::AgentUser->IsAuthorizedEx(resource[, clientIP][, transID])
```

Parameters

The IsAuthorizedEx method accepts the following parameters:

resource (AgentResource object)

  Specifies the resource to check.

clientIP (string)

  (Optional) Specifies the client's IP address.

transID (string)

  (Optional) Specifies the user-defined transaction ID that the agent uses to associate application activity with security activity.

Return Value

The IsAuthorizedEx method returns one of the following values:

- SM_AGENTAPI_YES (value = 1)

  Specifies that the user is authorized.

- SM_AGENTAPI_NO (value = 2)

  Specifies that the user is not authorized.

- SM_AGENTAPI_FAILURE (value = -1)

  Specifies that the Policy Server could not be reached.

- SM_AGENTAPI_TIMEOUT (value = -2)

  Specifies that the method timed out.
- SM_AGENTAPI_NOCONNECTION (value = -3)

  Specifies that initialization was not done.

Login Method—Performs Session Login and Validation

The Login method performs session login and validation. Before calling this method, call AgentResource->IsProtected for the target resource.

Syntax

The Login method has the following format:

```
Netegrity::AgentUser->Login(resource[, clientIP])
```

Parameters

The Login method accepts the following parameters:

resource (AgentResource object)

Specifies the resource to log in to.

clientIP (string)

(Optional) Specifies the client's IP address.

Return Value

The Login method returns one of the following values:

- SM_AGENTAPI_YES (value = 1)

  Specifies that user login was successful.
- SM_AGENTAPI_NO (value = 2)

  Specifies that user login failed.
- SM_AGENTAPI_CHALLENGE (value = 3)

  Specifies that a challenge is required for authentication.
- SM_AGENTAPI_FAILURE (value = -1)

  Specifies that the operation failed.

- SM_AGENTAPI_TIMEOUT (value = -2)

  Specifies that the method timed out.

- SM_AGENTAPI_NOCONNECTION (value = -3)

  Specifies that the object was not connected.

Remarks

To allow one user, who is already logged in, to log in again as another user, call AgentUser->Impersonate.

Logout Method—Logs the User out of the Session

The Logout method logs the user out of the session. Calling this method is optional, because the user is automatically logged out when the user object exceeds its scope in the Perl script.

Syntax

The Logout method has the following format:

```
Netegrity::AgentUser->Logout()
```

Parameters

The Logout method accepts no parameters.

Return Value

The Logout method returns one of the following values:

- SM_AGENTAPI_YES (value = 1)

  Specifies that the user logged out successfully.

- SM_AGENTAPI_NO (value = 2)

  Specifies that user logout failed.

- SM_AGENTAPI_CHALLENGE (value = 3)

  Specifies that a challenge is required for authentication.

- SM_AGENTAPI_FAILURE (value = -1)

  Specifies that the operation failed.

- SM_AGENTAPI_TIMEOUT (value = -2)

  Specifies that the method timed out.
- SM_AGENTAPI_NOCONNECTION (value = -3)

  Specifies that the object was not connected.

Remarks

Calling Logout while one user is impersonating another user ends the impersonation session and restores the impersonator's original session. Calling AgentUser->Impersonate allows one user to impersonate or log in as another user.

Name Method—Sets or Retrieves the User's Username

The Name method sets or retrieves the user's username.

Syntax

The Name method has the following format:

```
Netegrity::AgentUser->Name([username])
```

Parameters

The Name method accepts the following parameter:

username (string)

(Optional) Specifies the username to set.

Return Value

The Name method returns the following value:

- username (string)

  Specifies the new or existing username.

Remarks

Setting the username only affects the current instance of the user object. It does not affect the user's entry in the directory.

Password Method—Sets or Retrieves the User's Password

The Password method sets or retrieves the user's password.

Syntax

The Password method has the following format:

```
Netegrity::AgentUser->Password([password])
```

Parameters

The Password method accepts the following parameter:

password (string)

(Optional) Specifies the password to set.

Return Value

The Password method returns the following value:

- password (string)

  Specifies the new or existing password.

Remarks

Setting the password only affects the current instance of the user object. It does not affect the user's entry in the directory.

Validate Method—Validates a Session Specification

The Validate method validates a session specification, checking that a user session has neither expired nor been terminated or revoked. This check can occur at any time during the life of a session.

Syntax

The Validate method has the following format:

```
Netegrity::AgentUser->Validate(resource[, clientIP][, transID])
```

Parameters

The Validate method accepts the following parameters:

resource (AgentResource object)

Specifies the resource to log in to.

clientIP (string)

(Optional) Specifies the client's IP address.

transID (string)

(Optional) Specifies a user-defined transaction ID.

Return Value

The Validate method returns one of the following values:

- SM_AGENTAPI_YES (value = 1)

  Specifies that the operation was successful.

- SM_AGENTAPI_NO (value = 2)

  Specifies that the user was not logged in.

- SM_AGENTAPI_FAILURE (value = -1)

  Specifies that the operation failed.

- SM_AGENTAPI_TIMEOUT (value = -2)

  Specifies that the method timed out.

- SM_AGENTAPI_NOCONNECTION (value = -3)

  Specifies that the object was not connected.

Remarks

The Policy Server validates a session specification or session ID, as follows:

- If the session ID is specified without a session specification, the Policy Server uses the session ID for validation.
- If the session ID is specified with a session specification, the Policy Server validates the session ID against the session specification.
- If the client's IP address is specified with a session specification, the Policy Server validates the IP address against the session specification.

Chapter 4: Agent Operations

This section contains the following topics:

Resource Protection

When a user attempts to log into a site and access a protected resource, the agent typically needs to answer the following questions:

- Is the requested resource protected?
- Can the user be authenticated for login?
- Is the user authorized to access the resource?

The following script illustrates how you can use the Agent API to address and respond to these basic agent questions:

```
use Netegrity::AgentAPI;

#Define script variables
$agent = "agent1";
$secret = "oursecret";
$ip = "127.0.0.1";
$respath = "/mysite/hr/payroll.htm";
$username = "userid";
$pwd = "userpwd";

print "\nStep 1. Connecting to Policy Server...\n";
$agentapi = Netegrity::AgentAPI->New($agent, $secret);
$serverconfig = $agentapi->AddServerConfig($ip);
$status=$agentapi->Connect();
```

```
die "FATAL: Connect() failed with error code " .
                         $status unless($status==SM_AGENTAPI_YES);

$resource = $agentapi->GetResource($respath);
print "\nStep 2. Is the resource protected?\n";
if ($resource->IsProtected == SM_AGENTAPI_YES) {
   print "Resource ".$respath." is protected.\n\n";

   print "\nStep 3. User login...\n";
   $user = $agentapi->CreateUser($username, $pwd);
   print "Logging in user ".$user->Name().".\n";
   $status = $user->Login($resource);
   if($status==SM_AGENTAPI_YES) {
      print $user->Name() . " logged in successfully!\n\n";

      print "\nStep 4. User authorized for the resource?\n";
      $status = $user->IsAuthorized($resource);
      if($status==SM_AGENTAPI_YES) {
         print $user->Name()." is authorized for " .
                                             $respath . "\n\n";
      }
      else {
         print $user->Name()." is not authorized for " .
                                             $respath . "\n\n";
      }
   }
   else {
      print "Couldn't log in user " . $username . ".\n\n";
   }
}
else {
      print "Resource ".$respath." is not protected.\n\n";
}
```

Responses and Response Attributes

After an agent issues a request through a call to Login() or IsAuthorized(), a response to the request is returned. The response is returned whether or not the user is authorized.

With the returned response object, you can retrieve the following Agent API objects:

- Session object. The session object lets you retrieve session information such as session ID and timeout values.
- Response attribute object. The Policy Server uses response attributes to send information to agents. Agents can then pass that information on to the client application.

Retrieve Response Attributes

This sequence of calls retrieves the well-known attributes such as the user's distinguished name, the name of the directory associated with the user, and any text associated with the user's authentication. The online Agent API reference has a list of the well-known attributes that GetAttributes() can retrieve.

To retrieve response attributes

1. Call the AgentUser methods Login() and GetResponse().
2. With the AgentResponse object returned from GetResponse(), call GetAttributes().

Example: Retrieve the attributes for a response

The following script retrieves the attributes for a response to a user login request. The script then calls methods in the AgentResponseAttr object to display the attribute IDs, any attribute values, and other attribute information:

```
use Netegrity::AgentAPI;

$agentname="agent1";
$ip="127.0.0.1";
$sharedsecret="oursecret";
$username="userid";
$userpwd="userpwd";
```

```
$agentapi=Netegrity::AgentAPI->New($agentname,$sharedsecret);
#Add Policy Server configuration info...
$serverconfig = $agentapi->AddServerConfig($ip);
$agentapi->Connect();

$resource=$agentapi->GetResource("/mysite/hr/payroll.htm");
# Test whether the resource is protected. If it is,
# log in the user and get the attributes of the response.
if($resource->IsProtected() == SM_AGENTAPI_YES) {
   $user = $agentapi->CreateUser($username,$userpwd);
   print "\nLogging in user ".$user->Name()."...\n";
   $status=$user->Login($resource);
   if($status==SM_AGENTAPI_YES) {
      $response=$user->GetResponse();
      @attr = $response->GetAttributes();
      foreach $attr(@attr) {
         print "\nAttribute ID = " . $attr->GetID()."\n";
         print "TTL = " . $attr->GetTTL()."\n";
         print "Value = " . $attr->GetValue()."\n";
         print "Name = " . $attr->GetName()."\n";
      }
   }
}
else {
   print "\nThe resource is not protected.\n\n";
}
```

Session Management

After you retrieve an AgentSession object, you can perform session management operations. You can retrieve information about a session, such as session timeout values, the session ID, and the session specification.

The *session specification* can be used to identify a session across multiple sites, such as for single sign-on operations. You can also retrieve a reason code for a failed authentication or authorization attempt by calling GetReason().

Example: Login in a user

The following example logs in a user, gets a response to the login attempt, retrieves a session object for the user's session, and prints out various details about the session:

```
use Netegrity::AgentAPI;

#Define script variables
$agent = "agent1";
$secret = "oursecret";
$ip = "127.0.0.1";
$respath = "/mysite/hr/payroll.htm";
$username = "userid";
$pwd = "userpwd";

#Establish the connection and create needed objects
$agentapi = Netegrity::AgentAPI->New($agent, $secret);
$serverconfig = $agentapi->AddServerConfig($ip);
$agentapi->Connect();
$user = $agentapi->CreateUser($username, $pwd);
print "Logging in user ".$user->Name().".\n";
$resource = $agentapi->GetResource($respath);
#Log in the user
$status = $user->Login($resource);
if($status==SM_AGENTAPI_YES) {
   print $user->Name() . " logged in successfully!\n\n";
   #Get the login response
   $response=$user->GetResponse();
   #Get the session object
   $session=$response->GetSession();
   if ($session != undef) {
      print "Printing session details:\n";
      print "Session reason=".$session->GetReason()."\n";
      print "Session IdleTimeout=".$session->IdleTimeout()."\n";
      print "Session Maxtimeout=".$session->MaxTimeout()."\n";
      print "Session ID=".$session->GetID()."\n";
      print "Session specification=".$session->GetSpec()."\n\n";
   }
}
else {
      print "Couldn't log in user " . $username . "\n\n";
}
```

Policy Server Commands

A management protocol exists between agents and the Policy Server. An agent uses this protocol to manage its caches and encryption keys in a manner consistent with policies and administrative changes on the Policy Server.

When Policy Server changes occur that require corresponding agent changes, the Policy Server issues commands for the agent to make the changes. You can retrieve these commands by calling DoManagement(). DoManagement() returns an array of AgentResponseAttr objects. Call GetID() in this object to retrieve any management commands that might be pending from the Policy Server.

The following script checks for pending management commands. If any commands are found, the script prints out each command ID and, if appropriate, the value associated with a command:

```
use Netegrity::AgentAPI;

$agentname="agent1";
$ip="127.0.0.1";
$secret="oursecret";

$agentapi = Netegrity::AgentAPI->New($agentname,$secret);
$serverconfig = $agentapi->AddServerConfig($ip);
$agentapi->Connect();

@attr = $agentapi->DoManagement();

if (@attr == undef) {
   print "No commands are pending from the Policy Server.";
}
else {
   print "IDs of commands pending from the Policy Server:";
   foreach $attr(@attr) {
      if ($attr->GetID()==SM_AGENTAPI_AFFILIATE_KEY_UPDATE ||
         $attr->GetID()==SM_AGENTAPI_CACHE_FLUSH_THIS_USER ||
         $attr->GetID()==SM_AGENTAPI_CACHE_FLUSH_THIS_REALM) {
         print "\n Command ID: " . $attr->GetID();
         print "\n \tValue: " . $attr->GetValue();
      }
      else {
         #No need to print values for the IDs below. Value
         #will contain either binary data or no data.
         print "\nCommand ID: " . $attr->GetID();
      }
   }
}
```

Chapter 5: Policy Management API

This section contains the following topics:

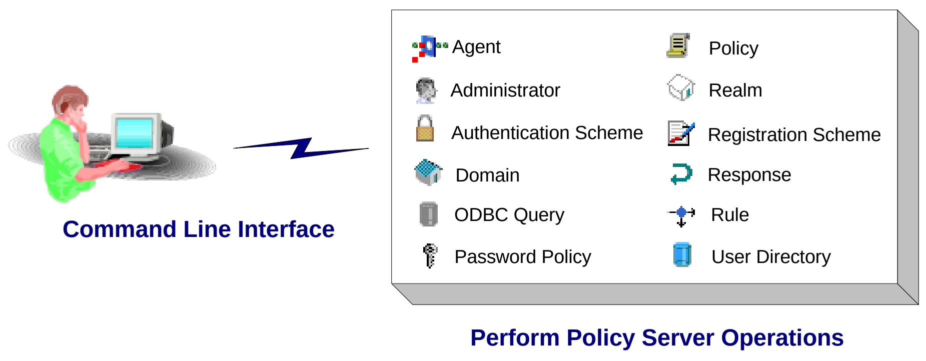
About the Policy Server and the Policy Management API

A SiteMinder Policy Server manages data that describes protected resources and the requirements for accessing those resources. A Policy Server also manages information about the administrators of the protected domains. This security data is located in a policy store such as LDAP.

The Policy Management API (module Netegrity::PolicyMgtAPI) lets you perform most of the Policy Server design and administration operations that you can perform with the Administrative UI. For example, you can:

- Create, modify, and delete policy store objects such as domains, realms, rules, and policies
- Manage administrators and user directories

The following illustration shows some of the policy store objects you can manage with the Policy Management API:



In addition, the Policy Management API data management object (PolicyMgtDataMgr) lets you copy specific objects from one policy store to another, rather than an entire policy store or domain as allowed by the SiteMinder smobjexport and smobjmport tools.

Location of the Policy Management API

The Policy Management API must be installed on the machine where the target Policy Server is located. The Policy Management API cannot access a remote Policy Server. However, the policy store can be on a remote machine as long as the Policy Server is configured to point to the remote policy store.

Write a Script against the Policy Management API

When you write a script against the Policy Management API, take the following basic steps:

1. Reference the Policy Management API at the beginning of your script:

   ```
   use Netegrity::PolicyMgtAPI;
   ```

2. Use the New() method to create a Policy Management API object:

   ```
   $policymgtapi = Netegrity::PolicyMgtAPI->New();
   ```

3. Optionally, set one or more Policy Server initialization flags through PolicyMgtAPI methods such as DisableValidation(). By default, all initialization flags are set to 0.

4. Create a session with the Policy Server:

   ```
   $session = $policymgtapi->CreateSession("userid",
                                           "password",
                                           "127.0.0.1" );
   ```

You can now perform operations against Policy Server objects. For example, you could retrieve and print out a list of configured agents in the Policy Server:

```
@agents = $session->GetAllAgents();
foreach $agent (@agents) {
   print "Agent Name = " . $agent->Name() . "\n";
}
```

Script Execution Performance Enhancement

You can reduce the time it takes for Policy Management scripts to make changes in the policy store. To do so, pass 0 in PreLoadCache() during initialization. By default, cache pre-loading is disabled.

Federation Manager: Legacy Federation

The CA SiteMinder federation programming interfaces allows businesses in a federated relationship across the Internet to securely share information about users visiting any of the federated sites. Users can access federated sites without having to supply credentials more than once.

Affiliated sites share user information through an industry-standard Security Assertion Markup Language (SAML) document called an *assertion*.

The CA SiteMinder federation product supports SAML 1.x and SAML 2.0 protocols.

SAML Assertions

A SAML assertion includes:

- Affiliate attributes, such as:
  - User profile information from a user directory, such as a user's email address or business title.
  - User entitlements, such as the user's credit limit at the affiliate site.
- Session information (SAML 1.x assertions)—for example, whether the assertion producer and the consumer can maintain separate sessions.

Note: You can modify the default assertion that the Policy Server generates. You do so through a custom Java class that you create with the SiteMinder Java SDK.

SAML 1.x

SAML 1.x support lets a user access a consumer site directly or from an assertion producer site without having to supply credentials more than once.

When a user requests access to a protected resource at an affiliate site, the Policy Server at the producer site is notified. After authenticating the user (if the user has not yet been authenticated), the Policy Server generates a SAML assertion from the affiliate object associated with the consumer site.

An application at the affiliate site then retrieves the SAML assertion from the Policy Server, and uses the information for authorization purposes and any other required purpose.

For example, suppose a user logs into a site for a bank (the producer site). The producer includes Policy Server software. The Policy Server contains an affiliate object that represents a site offering credit card services, and also other affiliate objects that represent other sites affiliated with the bank. When a user is authenticated at the producer, the user can click the link for the credit-card site and access the site without having to re-enter his credentials.

SAML 1.x Pseudo-code Example

The pseudo-code in this section illustrates the following operations:

1. Initialize the API.
2. Add an affiliate domain.
3. Add a user directory to an affiliate domain.
4. Create an affiliate in an affiliate domain.
5. Add users to an affiliate.
6. Add an attribute to an affiliate.
7. Get an existing affiliate domain.
8. Get all the affiliates in an affiliate domain.
9. Get all the attributes in an affiliate.
10. Remove an affiliate domain.

Note: Comments using <> notation represent code omitted for ease of understanding. Return code checking is omitted for ease of understanding.

```
# 1. Initialize the API
use Netegrity::PolicyMgtAPI;
$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");

# 2. Add an affiliate domain
$affdomain = $session->CreateAffDomain("name", "description");

# 3. Add a previously obtained user directory to the affiliate domain
# <Obtain $userdir via $session->GetAllUserDirs>
$affdomain->AddUserDir($userdir);
```

```
# 4. Create an affiliate in the affiliate domain
$affiliate = $affdomain->CreateAffiliate("affname", "password",
                                          http://authurl, 60, 30);

# 5. Add users from a previously obtained user table to the affiliate
# <Obtain $user via $userdir->GetContents>
$affdomain->AddUser($user);

# 6. Add an attribute for the affiliate
$affdomain->AddAttribute(1, "staticAttrName=StaticAttrValue");
# 7. Get an existing affiliate domain
$affiliate = $affdomain->GetAffiliate("affname");

# 8. Get all the affiliates in an affiliate domain
@affiliates = $affdomain->GetAllAffiliate();

# 9. Get all the attributes in an affiliate
@affiliateAttrs = $affiliate->GetAllAttributes();

# 10. Remove an affiliate domain
$session->DeleteAffDomain($affiliate);
```

SAML 2.0

With SAML 2.0, security assertions are shared between the following entities within a federation:

Identity Provider

An Identity Provider generates assertions for principals within a SAML 2.0 federation. The Identity Provider sends the SAML assertion to the Service Provider where the principal is attempting to access resources.

Service Provider

A Service Provider makes applications and other resources available to principals within a federation, using the identity information provided in an assertion. A principal is a user or another federation entity.

The Service Provider uses a SAML 2.0 authentication scheme to validate a user based on the information in a SAML 2.0 assertion.

Identity Providers and Service Providers can belong to a SAML affiliation. A SAML affiliation is a group of SAML entities that share a name identifier for a single principal.

Service Providers and Identity Providers can belong to an affiliation; however, an entity can belong to no more than one affiliation. Service Providers share the Name ID definition across the affiliation. Identity Providers share the user disambiguation properties across the affiliation.

Using affiliations reduces the configuration required at each Service Provider. Additionally, using one name ID for a principal saves storage space at the Identity Provider.

Single Sign-on Example

By sharing security assertions, a principal can log in at one site (the site acting as the Identity Provider), and then access resources at another site (the Service Provider) without explicitly supplying credentials at the second site.

For example:

1. The user is a home buyer who authenticates at a realtor’s web site.

   Any authentication scheme can be used to authenticate the user.

2. While viewing real estate listings, the user notices a link to a bank with an attractive mortgage rate.

3. The user clicks the link.

4. At the realtor’s site, an entity acting as the Identity Provider packages the user’s information in a SAML assertion, then transports the assertion to the bank’s site using the SAML 2.0 POST binding.

5. At the bank’s site, an entity acting as the Service Provider uses the SAML 2.0 Authentication scheme associated with the Identity Provider to validate the user for the resources on the bank’s site.

   This validation is transparent to the user.

6. If the user is successfully validated, the user is allowed on the bank’s site to view the rate information.

SAML 2.0 Pseudo-code Example

The pseudo-code in this section illustrates the following operations:

1. Initialize the API.
2. Retrieve the affiliate domain for the Service Provider.
3. Assign metadata constants to variables.
4. Assign values to the Service Provider metadata.
5. Create the Service Provider.
6. Retrieve users from the directory associated with the affiliate domain.
7. Add the users to the Service Provider.

8. Update the Service Provider's default skew time to 100.
9. Save the update.
10. Print the updated skew time.

```
# 1. Initialize the API
use Netegrity::PolicyMgtAPI;
$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");

# 2. Retrieve the affiliate domain for the Service Provider
$affDom=$session->GetAffDomain("AffiliateDomain");

# 3. Assign metadata constants to variables
$SAML_NAME=SAML_NAME;
$SAML_SP_AUTHENTICATION_URL=SAML_SP_AUTHENTICATION_URL;
$SAML_KEY_SPID=SAML_KEY_SPID;
$SAML_SP_IDPID=SAML_SP_IDPID;
$SAML_AUDIENCE=SAML_AUDIENCE;
$SAML_SP_ASSERTION_CONSUMER_DEFAULT_URL=
                           SAML_SP_ASSERTION_CONSUMER_DEFAULT_URL;
$SAML_SP_NAMEID_ATTRNAME=SAML_SP_NAMEID_ATTRNAME;
$SAML_SKEWTIME=SAML_SKEWTIME;

# 4. Assign values to the Service Provider metadata
%hsh=($SAML_NAME=>'My Service Provider',
   $SAML_SP_AUTHENTICATION_URL=>
                             'http://www.mysite.com/redirect.jsp',
   $SAML_KEY_SPID=>'http://www.spprovider.com',
   $SAML_SP_IDPID=>'http://www.idpprovider.com',
   $SAML_AUDIENCE=>'SSOAudience',
   $SAML_SP_ASSERTION_CONSUMER_DEFAULT_URL=>
                                 'http://www.defaultconsumer.com',
   $SAML_SP_NAMEID_ATTRNAME=>'attribute'
   );
# 5. Create the Service Provider
$sp=$affDom->CreateSAMLServiceProvider(\%hsh);

# 6. Retrieve users from the directory associated with the #    affiliate
domain—in this case, users in the group HR
$userDir=$session->GetUserDir("MyNtDirectory");
$usr=$userDir->LookupEntry("HR");

# 7. Add the users to the Service Provider
$sp->AddUser($usr);

# 8. Update the Service Provider's default skewtime to 100
$sp->Property($SAML_SKEWTIME,"100");
```

```
# 9. Save the update
$sp->Save();

# 10. Print the updated skewtime
print "\n";
print $sp->Property($SAML_SKEWTIME);
```

SAML 2.0 Attribute Authority

SiteMinder supports authorization for user access to a resource by requesting the values of predetermned user attributes from a remote site and then using those values as the basis for the authorization decision. The request contains no session information, because the user is not necessarily authenticated on the remote site.

For example, imagine a customer logs on to a car rental agency site to inquire about rates. The customer is authenticated by the agency, but to provide a competitive rate, the agency uses information from the customer's prefered airline. The car rental agency puts in a request to the airline's Web site to obtain the customer's quality code, which is based on the customer's accrued frequent flier miles. The airline returns the value of the quality code, for instance, 1A, and the car agency displays the appropriate rate sheet to the customer.

In this example, the car rental agency acts as what is know as the the SAML Requester, and the airline acts as what is known as a SAML Attribute Authority. Note that the customer is not authenticated by the Attribute Authority.

The Policy Server implements this kind of authorization decision by using variables within policy expressions. In the policy expressions, Federation attribute variables associate an attribute with a remote Attribute Authority. When the policy server attempts to resolve the Federation attribute variable, it determines the Attribute Authority from which to request the value of the attribute.

The Perl Policy Management API includes the following three methods in the PolicyMgtSession object to support authorization based on user attributes:

- AddAttributeToSAMLScheme()
- GetAllSAMLSchemeAttributes()
- RemoveAttributeFromSAMLScheme()

The PolicyMgtSAMLServiceProvider->AddAttribute method supports the addition of an attribute to the Service Provider (the Attribute Authority) that can be requested by a SAML Requester.

SAML 2.0 Indexed Endpoints

When configuring single sign-on at the Identity Provider, you can specify more than one endpoint for the Assertion Consumer Service, the service that enables a Service Provider to consume a SAML assertion. Each endpoint is assigned a unique index value, instead of a single, explicit reference to an Assertion Consumer Service URL. The assigned index can be used as part of a Service Provider's request for an assertion that it sends to the Identity Provider. This enables you to have a different Assertion Consumer Service at the Service Provider for different protocol bindings.

In a Perl script, you specify the index value, the protocol binding, and the URL of the indexed endpoint using the AddAssertionConsumerServiceToSAMLSP() method in the PolicyMgtSAMLServiceProvider object as follows:

```
$res = $sp->AddAssertionConsumerServiceToSAMLSP(index, protocolBinding, URL);
```

This method returns a PolicyMgtSAMLSPACS object. There are also methods in PolicyMgtSAMLServiceProvider for retrieving all Assertion Consumer Service objects and for removing an Assertion Consumer Service.

The PolicyMgtSAMLSPACS object includes methods to retrieve values for the index, protocol binding, and Assertion Consumer URL.

WS-Federation

The WS-Federation specification provides a protocol for how passive clients (such as Web browsers) implement the federation framework. ADFS is Microsoft's implementation of the WS-Federation Passive Requestor Profile.

Web SSO and sign-out in this environment are implemented using Account Partners and Resource Partners. An Account Partner authenticates users, provides WS-Federation security tokens, and passes them to a Resource Partner. The Resource Partner consumes security tokens and establishes a session based on the contents of the WS-Federation security token.

For SiteMinder to act as an Account Partner, an administrator must define the Resource Partner that will be consuming security tokens. This is done by defining a Resource Partner in an Affiliate domain. For SiteMinder to act as a Resource Partner, an administrator must define the Account Partner that is going to supply security tokens. This is done by defining a WS-Federation authentication scheme.

In a Perl script, you define a Resource Partner by calling the PolicyMgtAffDomain->CreateWSFEDResourcePartner method as follows:

```
$aff = $affDomain->CreateWSFEDResourcePartner(propsHash_ref);
```

propsHash_ref is a reference to a hash table of metadata properties defined for the Resource Partner.

This method returns a PolicyMgtWSFEDResourcePartner object. The PolicyMgtWSFEDResourcePartner object includes methods for managing users in the Resource Partner (AddUser, GetAllUsers, and RemoveUser). Note that the PolicyMgtWSFEDResourcePartner->Property() method does not submit changes to the data store. You must call the PolicyMgtWSFEResourcePartner->Save() method.

To define an Account Partner in a Perl script you create an instance of a WS-Federation authentication scheme by calling PolicyMgtSession->CreateWSFEDAuthScheme(). You can set or retrieve metadata properties for this authentication scheme by calling PolicyMgtSession->WSFEDAuthSchemeProperties().

There are no methods for deleting or retrieving a WS-Federation authentication scheme specifically. You use the DeleteAuthScheme, GetAuthScheme, and GetAllAuthSchemes as you would for any other type of authentication scheme. .

More Information:

WS-Federation Resource Partner Methods (see page 537)
WS-Federation Resource Partner Attribute Methods (see page 546)

Sample Scripts

The sample scripts AffiliateDemo.pl, SAMLServiceProvider.pl, WSFEDAccountPartner.pl, and WSFEDResourcePartner.pl are provided with legacy federation.

If not installed already Import ampolicy.smdif into the Policy Store before using the sample script. The default location of ampolicy.smdif is <*siteminder_install_dir*>\db\SMdif.

Affiliate Domains

Affiliate objects representing sites in a federated business network are contained within a Policy Server affiliate domain. An affiliate domain can contain SAML 1.x affiliates and SAML 2.0 Service Providers.

An affiliate domain also contains references to one or more user directories associated with the affiliates and Service Providers in the domain, and to the administrator accounts that can manage the domain.

You can use the Command Line Interface to configure affiliate objects and the affiliate domain where they reside.

Note: For more information about affiliate objects and affiliate domains, including prerequisites for using affiliate functionality, see the *Policy Server Configuration Guide*.

Authentication Scheme Configuration

When you configure an authentication scheme through a Perl script, you provide information that would otherwise be provided through the Authentication Scheme Properties dialog of the Administrative UI. This section describes the information you need to configure a given authentication scheme using the Policy Management API.

Note: When modifying an authentication scheme, be sure to call Save() after calling all the configuration methods.

Configuration Information

Typically, you configure an authentication scheme when you create the scheme with CreateAuthScheme() or when you modify the scheme with the methods in the PolicyMgtAuthScheme object.

Note: The exception to this rule is an authentication scheme based on the SAML 2.0 Template. You create and configure a SAML 2.0 authentication scheme with the method CreateSAMLAuthScheme().

You can provide the following kinds of configuration information for an authentication scheme. Not every authentication scheme template uses all categories of configuration information:

- Scheme type

  SiteMinder provides a number of standard authentication scheme types (also known as templates). Each authentication scheme type is configured differently.

- Description

  Brief description of the authentication scheme.

- Protection level

  Protection level values can range from 1 through 1000. The higher the number, the greater the degree of protection provided by the scheme.

- Library

  An authentication scheme library performs authentication processing for the associated authentication scheme type. Each predefined authentication scheme is shipped with a default library. Optionally, you can use a custom library instead of the default.

- Parameter

  Additional information that the authentication scheme requires, such as the URL of an HTML login page.

  With some authentication schemes, the parameter information is constructed from field values in the Scheme Type Setup tab of the Authentication Scheme Properties dialog. To see how a parameter string is constructed for a given scheme type, open this dialog, select the appropriate scheme type, provide values to the fields in the Scheme Type Setup tab, and view the constructed parameter in the Advanced tab.

- Shared Secret

  Information that is known to both the authentication scheme and the Policy Server. Different authentication schemes use different kinds of secrets. Most schemes use no secret.

- Is template?

  A flag that specifies whether the authentication scheme is a template.

  Note: Setting an authentication scheme as a template with the Perl Policy Management API is deprecated in SiteMinder v6.0 SP3.

- Is used by administrator?

  A flag that specifies whether the authentication scheme can be used to authenticate administrators.

- Save Credentials?

  A flag that specifies whether the user’s credentials are saved.

- Is RADIUS?

  A flag that specifies whether the scheme can be used with RADIUS agents.

- Ignore password check?

  A flag that specifies whether password policies for the scheme are enabled. If 1, password policies are disabled.

  Note: The Ignore password check flag must be set to True for anonymous authentication schemes.

Configuration Tables

The following tables will help you configure authentication schemes. Each table applies to a particular authentication scheme type and contains the following information:

- The PolicyMgtAuthScheme method that sets or retrieves a particular kind of configuration value.
- The default configuration value, if applicable, or a variable name, shown in italics, that represents the configuration value to be supplied.
- The name of the CreateAuthScheme() argument used to supply the configuration value when you are creating an authentication scheme.

The values in the Information Type column can be used for different purposes in different authentication schemes. For example, with TeleID authentication schemes, the shared secret is used to supply the encryption seed.

Anonymous Template

Use this table when configuring an authentication scheme based on the scheme type Anonymous.

Note: The Ignore password check flag must be set to True for anonymous authentication schemes.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type Anonymous.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(0) CreateAuthScheme() param: *protLevel* Set to 0. Not applicable to this scheme type.
Library	CustomLib("smauthanon") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the guest DN. Policies associated with the guest DN must apply to anonymous users.

Information Type	Value Assignment and Meaning
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme is not used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(1) CreateAuthScheme() param: *ignorePwd* Set to 1—ignore password checking.

Basic Over SSL Template

Use this table when configuring an authentication scheme based on the scheme type Basic over SSL.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type Basic over SSL.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.

Information Type	**Value Assignment and Meaning**
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 10.
Library	CustomLib("smauthcert") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the domain or IP address of the SSL server and the name of the SSL Credentials Collector (SCC). Format: https://*server*/*SCC*?basic The following example uses the default SCC: https://my.server.com/siteminderagent/ nocert/smgetcred.scc?basic
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0 for this scheme.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(*flag*) CreateAuthScheme() param: *ignorePwd* Set to 1 to ignore password checking, or 0 to check passwords. Default is 0.

Basic Template

Use this table when configuring an authentication scheme based on the scheme type Basic.

Information Type	**Value Assignment and Meaning**
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type Basic.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthdir") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam("") CreateAuthScheme() param: *schemeParam* Set to an empty string. Not applicable to this scheme.
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(1) CreateAuthScheme() param: *isUsedByAdmin* Set to 1—scheme can be used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.

Information Type	Value Assignment and Meaning
Is RADIUS?	IsRadius(1) CreateAuthScheme() param: *isRadius* Set to 1—scheme can be used with RADIUS agents.
Ignore password check?	IgnorePwd(*flag*) CreateAuthScheme() param: *ignorePwd* Set to 1 to ignore password checking, or 0 to check passwords. Default is 0.

Custom Template

Use this table when configuring an authentication scheme based on the scheme type Custom. You create custom schemes using the C-language Authentication API, which is available with the SiteMinder SDK.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type Custom.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 0 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib(*customLibName*) CreateAuthScheme() param: *schemeLib* The name of the custom shared library you created using the C Authentication API.
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* Any string of one or more parameters required by your custom authentication scheme. For a custom authentication scheme that uses SSL, you must supply a URL that points to a SiteMinder Web Agent library required for the SSL-based authentication.

Information Type	**Value Assignment and Meaning**
Shared secret	CustomSecret(*secret*) CreateAuthScheme() param: *secret* The shared secret, if any, that your custom authentication scheme uses for encryption of credentials.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(*flag*) CreateAuthScheme() param: *isUsedByAdmin* Set to true (1) to specify that the scheme can be used to authenticate administrators, or to false (0) to specify that the scheme cannot be used to authenticate administrators. Default is 0.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius*
Ignore password check?	IgnorePwd(*flag*) CreateAuthScheme() param: *ignorePwd* Set to 1 to ignore password checking, or 0 to check passwords. Default is 0.

HTML Form Template

Use this table when configuring an authentication scheme based on the scheme type HTML Form.

Information Type	**Value Assignment and Meaning**
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type HTML Form.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.

Information Type	**Value Assignment and Meaning**
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthhtml") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing a user attribute list plus the location of the forms credential collector (FCC). The attribute list must begin with AL= and use commas as the list delimiter character, and it must end with a semicolon—for example: AL=Password,SSN,age,zipcode; The complete parameter format is: *attr-list*;https:/*server*/*fcc* The following example uses the default FCC: AL=PASSWORD,SSN,age,zipcode; http://my.server.com/siteminderagent/ forms/login.fcc
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme is not used to authenticate administrators.
Save credentials?	SaveCredentials(*credFlag*) CreateAuthScheme() param: *saveCreds* Set to 1 to indicate that user credentials should be saved, or 0 to indicate that user credentials should not be saved. Default is 0.

Information Type	Value Assignment and Meaning
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(*flag*) CreateAuthScheme() param: *ignorePwd* Set to 1 to ignore password checking, or 0 to check passwords. Default is 0.

Impersonation Template

Use this table when configuring an authentication scheme based on the scheme type MS Passport.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type MS Passport.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 1.
Library	CustomLib("smauthmspp") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.

Information Type	**Value Assignment and Meaning**
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* The following information, separated by semicolons: ■ A DN for an anonymous user. Format: *anonuser=anonUserDN.* If you specify an anonymous user DN, the protection level is 0. ■ The search string for looking up a user in a user directory of the specified type. Format: *attribute=nameSpace*:*attrib=searchSpec* Valid namespaces are LDAP, AD, ODBC, WinNT, and Custom. ■ The registration URL. The URL can be a custom URL or a SiteMinder form. Formats: registrationurl=*URL* (custom URL) registrationurl=FORM=*URL* (SiteMinder form) Example using an LDAP attribute and a custom URL: attribute=LDAP:altSecurityIdentities= Kerberos:%s@company.local;registrationurl =http://passport.xanadu.local/registration/passportreg.asp
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme is not used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.

Information Type	**Value Assignment and Meaning**
Ignore password check?	IgnorePwd(1) CreateAuthScheme() param: *ignorePwd* Set to 1—ignore password checking.

RADIUS CHAP/PAP Template

Use this table when configuring an authentication scheme based on the scheme type RADIUS CHAP/PAP.

Information Type	**Value Assignment and Meaning**
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type RADIUS CHAP/PAP.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthchap") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the name of a user directory attribute. This attribute is used as the clear text password for authentication.
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.

Information Type	Value Assignment and Meaning
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme is not used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(1) CreateAuthScheme() param: *isRadius* Set to 1—scheme can be used with RADIUS agents.
Ignore password check?	setIgnorePwCheck(*flag*) Set to 1 to ignore password checking, or 0 to check passwords. Default is 0.

RADIUS Server Template

Use this table when configuring an authentication scheme based on the scheme type RADIUS Server.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type RADIUS Server.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthradius") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.

Information Type	**Value Assignment and Meaning**
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the IP address and port of the RADIUS server—for example: 123.123.12.12:1645 The default UDP port is 1645.
Shared secret	CustomSecret(*secret*) CreateAuthScheme() param: *secret* The user attribute that the RADIUS Server will use as the clear text password.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(1) CreateAuthScheme() param: *isUsedByAdmin* Set to 1—scheme can be used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(1) CreateAuthScheme() param: *isRadius* Set to 1—scheme can be used with RADIUS agents.
Ignore password check?	IgnorePwd(*flag*) CreateAuthScheme() param: *ignorePwd* Set to 1 to ignore password checking, or 0 to check passwords. Default is 0.

SafeWord HTML Form Template

Use this table when configuring an authentication scheme based on the scheme type SafeWord.

Information Type	**Value Assignment and Meaning**
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type SafeWord HTML Form.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 10.
Library	CustomLib("smauthenigmahtml") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the name and location of the forms credentials collector. This example shows the default credentials collector: http://my.server.com/ siteminderagent/forms/safeword.fcc
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(1) CreateAuthScheme() param: *isUsedByAdmin* Set to 1—scheme can be used to authenticate administrators.

Information Type	Value Assignment and Meaning
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(1) CreateAuthScheme() param: *isRadius* Set to 1—scheme can be used with RADIUS agents.
Ignore password check?	IgnorePwd(1) CreateAuthScheme() param: *ignorePwd* Set to 1—ignore password checking.

SafeWord Template

Use this table when configuring an authentication scheme based on the scheme type SafeWord.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type SafeWord.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 10.
Library	CustomLib("smauthenigma") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam("") CreateAuthScheme() param: *schemeParam* Set to an empty string. Not applicable to this scheme.
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.

Information Type	Value Assignment and Meaning
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(1) CreateAuthScheme() param: *isUsedByAdmin* Set to 1—scheme can be used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(1) CreateAuthScheme() param: *isRadius* Set to 1—scheme can be used with RADIUS agents.
Ignore password check?	IgnorePwd(1) CreateAuthScheme() param: *ignorePwd* Set to 1—ignore password checking.

SAML Artifact Template

Use this table when configuring a SAML authentication scheme based on the profile type *artifact* for communicating security assertions. With the artifact profile type, the URL for retrieving the SAML assertion is referenced within the *AssertionRetrievalURL* portion of the Parameter string.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type SAML Artifact.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.

Information Type	Value Assignment and Meaning
Library	CustomLib("smauthsaml") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* The following required parameters: ■ Name. The name of the affiliate. ■ RedirectMode. The way in which the SAML Credentials Collector redirects to the target resource. One of the following numeric values: 0. Meaning: 302 No Data. 1. Meaning: 302 Cookie Data. 2. Meaning: Server Redirect. 3. Meaning: Persist Attributes. ■ SRCID. The 20-byte source ID for the site that produces the SAML assertion. The ID is located at the SAML assertion producer’s site in the properties file AMAssertionGenerator.properties. ■ AssertionRetrievalURL. The URL for obtaining the assertion from the SAML assertion producer’s site. ■ Audience. The URI of the document that describes the agreement between the assertion producer site and the affiliate. This value is compared with the audience value specified in the SAML assertion. ■ Issuer. The SAML issuer specified in the assertion.

Information Type	**Value Assignment and Meaning**
Parameter (con't)	■ AttributeXPath. A standard XPath query run against the SAML assertion. The query obtains the data that is substituted in a search specification that looks up a user—for example: //saml:AttributeValue/SM:/SMContent /SM:Smlogin/SM:Username.text() This query gets the text of the Username element. ■ SAMLVersion. The SAML version in use: 1.0 or 1.1. ■ RetrievalMethod. One of these values: 0. Meaning: Basic authentication. 1. Meaning: Client certificate authentication. ■ attribute. The search string for looking up a user in a user directory of the specified type. Use a percent sign (%) to indicate where the value returned from the XPath query should be inserted. For example, if you specify attribute **LDAP:uid=%s**, and **user1** is returned from the query, the search string used for LDAP directories is **uid=user1**. At least one attribute must be specified. Format of the parameter string is as follows. Separate name-value pairs with semi-colons (;). The format example includes LDAP and ODBC attributes: Name=*name*;RedirectMode=0|1|2;SRCID=*srcid*; AssertionRetrievalURL=*url*;Audience=*audience*; Issuer=*issuer*;AttributeXpath=*XPathQuery*; SAMLVersion=1.0|1.1;RetrievalMethod=0|1; attribute=LDAP:*srch*Spc;attribute=ODBC:*srchSpc*
Shared secret	CustomSecret(*secret*) CreateAuthScheme() param: *secret* The password for the affiliate site. The password must match the password entered for the affiliate at the site where the SAML assertion is produced.
Is template?	IsTemplate(0) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme cannot be used to authenticate administrators.

Information Type	Value Assignment and Meaning
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(1) CreateAuthScheme() param: *ignorePwd* Set to 1—ignore password checking.

SAML POST Template

Use this table when configuring a SAML authentication scheme based on the profile type *POST* for communicating security assertions. With the POST profile type, the generated SAML assertion is POSTed to the URL specified in the *AssertionConsumerURL* portion of the Parameter string.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type SAML POST.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthsaml") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.

Information Type	**Value Assignment and Meaning**
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* The following required parameters: ■ Name. The name of the affiliate. ■ SAMLProfile. The profile type: POST. ■ SAMLVersion. The SAML version in use. The POST profile requires version 1.1. ■ RedirectMode. The way in which the SAML Credentials Collector redirects to the target resource. One of the following numeric values: 0. Meaning: 302 No Data. 1. Meaning: 302 Cookie Data. 2. Meaning: Server Redirect. 3. Meaning: Persist Attributes. ■ AssertionConsumerURL. The URL to be sent the generated assertion. ■ Audience. The URI of the document that describes the agreement between the assertion producer site and the affiliate. This value is compared with the audience value specified in the SAML assertion. ■ Issuer. The SAML issuer specified in the assertion.

Information Type	**Value Assignment and Meaning**
Parameter (con't)	■ AttributeXPath. A standard XPath query run against the SAML assertion. The query obtains the data that is substituted in a search specification that looks up a user—for example: //saml:AttributeValue/SM:/SMContent /SM:Smlogin/SM:Username.text() This query gets the text of the Username element. ■ attribute. The search string for looking up a user in a user directory of the specified type. Use a percent sign (%) to indicate where the value returned from the XPath query should be inserted. For example, if you specify attribute **LDAP:uid=%s**, and **user1** is returned from the query, the search string used for LDAP directories is **uid=user1**. At least one attribute must be specified. Format of the parameter string is as follows. Separate name-value pairs with semi-colons (;). The format example includes LDAP and ODBC attributes: Name=*name*;SAMLProfile=POST; SAMLVersion=1.1;RedirectMode=0|1|2; AssertionConsumerURL=*consumerUrl*; Audience=*audience*;Issuer=*issuer*; AttributeXpath=*XPathQuery*; attribute=LDAP:*srch*Spc;attribute=ODBC:*srchSpc*
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(0) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme cannot be used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.

Information Type	Value Assignment and Meaning
Ignore password check?	IgnorePwd(1) CreateAuthScheme() param: *ignorePwd* Set to 1—ignore password checking.

SAML 2.0 Template

This authentication scheme is based on the SAML 2.0 scheme type. It is configured by a SAML 2.0 Service Provider.

A Service Provider uses this authentication scheme to transparently validate a user based on the information in a SAML 2.0 assertion.

An authentication scheme based on the SAML 2.0 Template differs from other types of authentication schemes in two ways:

- SAML 2.0 authentication schemes are created with the method CreateSAMLAuthScheme(). This method creates a PolicyMgtAuthScheme object, just as CreateAuthScheme() does.
- SAML 2.0 authentication schemes have two sets of properties:
    - The properties listed in the table that follows. These properties are stored in the PolicyMgtAuthScheme object.

      Typically, the only properties in this set that you might choose to modify in an existing SAML 2.0 authentication scheme are name, description, and protection level. Modify these properties with the appropriate method in the PolicyMgtAuthScheme object.
    - Metadata properties for the associated Identity Provider. The associated Identity Provider is the Identity Provider that supplies the assertion to the Service Provider.

These properties are stored with the PolicyMgtAuthScheme object as a hashtable.

For information about the metadata properties you can assign to a SAML 2.0 authentication scheme, see the section SAML 2.0 Property Reference in this guide.

Where applicable, the method CreateSAMLAuthScheme() is referenced in place of CreateAuthScheme().

Information Type	**Value Assignment and Meaning**
Metadata properties	CreateSAMLAuthScheme() param: propsHash_ref The hashtable of SAML 2.0 metadata properties associated with the authentication scheme object. Call SAMLAuthSchemeProperties() to modify metadata properties associated with an existing SAML 2.0 authentication scheme.
Scheme type	Type(templateObject) The scheme type SAML 2.0.
Description	Description(schemeDesc) CreateSAMLAuthScheme() param: schemeDesc The description of the authentication scheme.
Protection level	ProtectionLevel(nLevel) CreateSAMLAuthScheme() param: protLevel A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthsaml") The default library for this scheme type.
Parameter	CustomParam(param) Set as an empty string.
Shared secret	CustomSecret("") Set as an empty string. Not applicable to this scheme.
Is template?	IsTemplate(0) Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) Set to 0—scheme cannot be used to authenticate administrators.
Save credentials?	SaveCredentials(0) Set to 0 to indicate that user credentials will not be saved.

Information Type	Value Assignment and Meaning
Is RADIUS?	IsRadius(0) Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(1) Set to 1—ignore password checking.

SecurID HTML Form Template

Use this table when configuring an authentication scheme based on the scheme type SecureID HTML Form.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type SecureID HTML Form.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 15.
Library	CustomLib("smauthacehtml") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.

Information Type	Value Assignment and Meaning
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the name of the attribute that contains the ACE IDs, the Web server where the forms credential collector (FCC) is installed, and the target executable file required for processing SecureID authentication with forms support. It also specifies whether an SSL connection is required. Format: *attr*;https://*server*/*target* The "s" in "https" is optional, depending on whether you want an SSL connection. The following example uses the default for processing SecureID authentication with forms support: ace_id;https://my.server.com/siteminderagent/pwcgi/smpwservicescgi.exe
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme is not used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(1) CreateAuthScheme() param: *ignorePwd* Set to 1—ignore password checking.

SecurID Template

Use this table when configuring an authentication scheme based on the scheme type SecureID.

Information Type	**Value Assignment and Meaning**
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type SecureID.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 15.
Library	CustomLib("smauthace") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the attribute in the authentication user directory that contains the ACE Server user ID.
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(1) CreateAuthScheme() param: *isUsedByAdmin* Set to 1—scheme can be used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.

Information Type	Value Assignment and Meaning
Is RADIUS?	IsRadius(1) CreateAuthScheme() param: *isRadius* Set to 1—scheme can be used with RADIUS agents.
Ignore password check?	IgnorePwd(1) CreateAuthScheme() param: *ignorePwd* Set to 1—ignore password checking.

smauthetsso Authentication Scheme

This authentication scheme is similar to the SiteMinder X.509 certification scheme, but with an eSSO cookie as the authentication credential instead of an X.509 credential.

If this scheme is configured for either cookieorbasic or cookieorforms mode, and both an eSSO cookie and login name and password credentials are passed to it, the eSSO cookie is ignored, and the login name and password are used to authenticate the user to SiteMinder.

When the eSSO cookie is the only credential, the authentication scheme uses the ETWAS API to connect to the configured eSSO Policy Server to validate the cookie and extract the user Distinguished Name (DN) from it.

Use this table when configuring an an smauthetsso authentication scheme, which is based on the scheme type Custom. You create custom schemes using the C-language Authentication API, which is available with the SiteMinder SDK.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type Custom.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 0 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthetsso") CreateAuthScheme() param: *schemeLib* The name of the library for this authentication scheme.

Information Type	**Value Assignment and Meaning**
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* An ordered set of tokens, separated by semi-colons: <*Mode*>[; <*Target*>]; <*Admin*>; <*eTPS_Host*> You can add spaces to make the string easier to read. <*Mode*> specifies the type of credentials that the authenticaion scheme will accept. The following values are possible: ■ cookie -- Only eTrust SSO Cookies are acceptable ■ cookieorbasic -- If an eTrust SSO Cookie is not provided, a login name and password are requested by using Basic Authentication. ■ cookieorforms -- If an eTrust SSO Cookie is not provided, a login name and password are requested by using Forms Authentication. <*Target*> is valid only with cookieorforms mode. This is identical to the Target field for standard HTML Forms Authentication Scheme. <*Admin*> specifies the login ID of an administrator for the Policy Server. The password for this administrator has been specified in the Shared Secret field. <*eTPO_Host*> specifies the name of the amchine on which the Policy Server is installed. SiteMinder will authenticate itself as <*Admin*> to the Policy Server on the <*eTPS_Host*> so that SiteMinder can request validation of eTrust SSO cookies. Examples: "cookie; SMPS_sso; myserver.myco.com" "cookieorforms; /siteminderagent/forms/login.fcc; SMPS_sso; myserver.myco.com"
Shared secret	CustomSecret(*secret*) CreateAuthScheme() param: *secret* The password of the Policy Server administrator named in the Parameter field.

Information Type	Value Assignment and Meaning
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(*flag*) CreateAuthScheme() param: *isUsedByAdmin* Set to true (1) to specify that the scheme can be used to authenticate administrators, or to false (0) to specify that the scheme cannot be used to authenticate administrators. Default is 0.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(*flag*) CreateAuthScheme() param: *ignorePwd* Set to 1 to ignore password checking, or 0 to check passwords. Default is 0.

TeleID Template

Use this table when configuring an authentication scheme based on the scheme type TeleID.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type TeleID.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 15.

Information Type	**Value Assignment and Meaning**
Library	CustomLib("smauthencotone") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam("") CreateAuthScheme() param: *schemeParam* Set to an empty string. Not applicable to this scheme.
Shared secret	CustomSecret(*seed*) CreateAuthScheme() param: *secret* The encryption seed. SiteMinder uses this value as an encryption seed for initializing hardware tokens.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(1) CreateAuthScheme() param: *isUsedByAdmin* Set to 1—scheme can be used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(1) CreateAuthScheme() param: *isRadius* Set to 1—scheme can be used with RADIUS agents.
Ignore password check?	IgnorePwd(1) CreateAuthScheme() param: *ignorePwd* Set to 1—ignore password checking.

Windows Authentication Template

Use this table when configuring an Integrated Windows Authentication scheme based on the scheme type Windows Authentication (previously known as NTLM). This scheme type is used to authenticate against WinNT or Active Directory user stores.

An Active Directory can be configured to run in *mixed mode* or *native mode*. An Active Directory supports WinNT style authentication when running in mixed mode. In native mode, an Active Directory supports only LDAP style lookups.

This authentication scheme supports either mixed mode or native mode.

Information Type	**Value Assignment and Meaning**
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type Windows Authentication (NTLM).
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthntlm") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.

Information Type	**Value Assignment and Meaning**
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* The value of *param* determines the style of authentication to perform for this scheme: **NTLM authentication** (for WinNT or Active Directory running in mixed mode) Format: *iis-web-server-url*/*path-to-ntc-file* In the format, *iis-web-server-url* is the name of the IIS web server that is the target of the redirection, and *path-to-ntc-file* is the location of the .ntc file that collects the WinNT credentials. For example: http://myiiswebserver.mycompany.com/ siteminderagent/ntlm/creds.ntc A SiteMinder Web Agent must be installed on the specified server. By default, the Web Agent installation creates a virtual directory for NTLM credential collection. **Windows Authentication (for Active Directory running in native mode)** With this authentication style, *param* has an LDAP filter added to the beginning of the redirection URL. The filter and URL are separated by a semi-colon (;). For example: cn=%{UID},ou=Users,ou=USA,dc=%{DOMAIN}, dc=mycompany,dc=com;http:// myiiswebserver.mycompany.com/ siteminderagent/ntlm/creds.ntc SiteMinder uses the LDAP filter to map credentials received from the browser/Web Agent to an LDAP DN or search filter.
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme is not used to authenticate administrators.

Information Type	**Value Assignment and Meaning**
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(*flag*) CreateAuthScheme() param: *ignorePwd* For WinNT and for Active Directory running in mixed mode, this property must be true (1)—ignore password checking. For Active Directory running in native mode, set to true (1) to ignore password checking, or false (0) to check passwords. Default is 0.

WS-Federation Template

This authentication scheme is based on the WS-Federation scheme type. It is configured by a WS-Federation Resource Partner.

A Resource Partner uses this authentication scheme to validate a user transparently based on the information in a SAML 1.1 assertion.

An authentication scheme based on the WS-Federation Template differs from other types of authentication schemes in two ways:

- WS-Federation authentication schemes are created with the method CreateWSFEDAuthScheme(). This method creates a PolicyMgtAuthScheme object, just as CreateAuthScheme() does.
- WS-Federation authentication schemes have two sets of properties:
  - The properties listed in the table that follows. These properties are stored in the PolicyMgtAuthScheme object.

    Typically, the only properties in this set that you might choose to modify in an existing WS-Federation authentication scheme are name, description, and protection level. Modify these properties with the appropriate method in the PolicyMgtAuthScheme object.
  - Metadata properties for the associated Account Partner. The associated Account Partner is the one that supplies the assertion to the Resource Partner.

These properties are stored with the PolicyMgtAuthScheme object as a hashtable.

Note: For information about the metadata properties you can assign to a WS-Federation authentication scheme, see the section WS-Federation Property Reference in the online Policy Management API Reference.

This authentication scheme requires CA Federation Manager: Legacy Federation. Legacy Federation is licensed separately.

Information Type	**Value Assignment and Meaning**
Metadata properties	CreateWSFEDAuthScheme() param: propsHash_ref The hashtable of WS-Federation metadata properties associated with the authentication scheme object. Call WSFEDAuthSchemeProperties() to modify metadata properties associated with an existing WS-Federation authentication scheme.
Scheme type	Type(templateObject) The scheme type WSFED.

Information Type	Value Assignment and Meaning
Description	Description(schemeDesc) CreateWSFEDAuthScheme() param: schemeDesc The description of the authentication scheme.
Protection level	ProtectionLevel(nLevel) CreateWSFEDAuthScheme() param: protLevel A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthsaml") The default library for this scheme type.
Parameter	CustomParam(param) Set as an empty string.
Shared secret	CustomSecret("") Set as an empty string. Not applicable to this scheme.
Is template?	IsTemplate(0) Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) Set to 0—scheme cannot be used to authenticate administrators.
Save credentials?	SaveCredentials(0) Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(1) Set to 1—ignore password checking.

X.509 Client Cert and Basic Template

Use this table when configuring an authentication scheme based on the scheme type X.509 Client Certificate *and* Basic.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type X.509 Client Certificate and Basic.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 15.
Library	CustomLib("smauthcert") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the domain or IP address of the SSL server and the name and path of the SSL Credentials Collector (SCC). The server redirects a user’s X.509 certificate over an SSL connection. Format: https://*server:port/SCC*?cert+basic The following example uses the default SCC: https://my.server.com:80/siteminderagent/ cert/smgetcred.scc?cert+basic
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.

Information Type	**Value Assignment and Meaning**
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme is not used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(*flag*) CreateAuthScheme() param: *ignorePwd* Set to 1 to ignore password checking, or 0 to check passwords. Default is 0.

X.509 Client Cert and Form Template

Use this table when configuring an authentication scheme based on the scheme type X.509 Client Certificate *and* Form.

Information Type	**Value Assignment and Meaning**
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type X.509 Client Certificate and HTML Form.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 15.
Library	CustomLib("smauthcert") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.

Information Type	**Value Assignment and Meaning**
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the domain or IP address of the SSL server and the name and path of the forms credentials collector (FCC). The server redirects a user's X.509 certificate over an SSL connection. Format: https://*server:port*/*FCC*?cert+forms The following example uses the default FCC: https://my.server.com:80/siteminderagent/ certoptional/forms/login.fcc?cert+forms
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme is not used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(*flag*) CreateAuthScheme() param: *ignorePwd* Set to 1 to ignore password checking, or 0 to check passwords. Default is 0.

X.509 Client Cert or Basic Template

Use this table when configuring an authentication scheme based on the scheme type X.509 Client Certificate *or* Basic.

Information Type	**Value Assignment and Meaning**
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type X.509 Client Certificate or Basic.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthcert") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the following information: ■ Server for establishing an SSL connection. This server redirects a user's X.509 certificate over an SSL connection. ■ Name and path of the SSL Credentials Collector (SSC). If you are using basic authentication over SSL, also provide the following two pieces of information: ■ The fully qualified name of the SSL server used for establishing an SSL connection for basic authentication. ■ Name and path of the SSL Credentials Collector (SSC). https://*SSLserver:port/SCC*?certorbasic; [https://*BasicServer/SCC*] The following example uses the default SCC values: https://my.SSLserver.com:80/siteminderagent/ certoptional/smgetcred.scc?certorbasic; https://my.BasicServer.com/ siteminderagent/nocert/smgetcred.scc

Information Type	Value Assignment and Meaning
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme is not used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(*flag*) CreateAuthScheme() param: *ignorePwd* Set to 1 to ignore password checking, or 0 to check passwords. Default is 0.

X.509 Client Cert or Form Template

Use this table when configuring an authentication scheme based on the scheme type X.509 Client Certificate *or* Form.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type X.509 Client Certificate or HTML Form.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.

Information Type	**Value Assignment and Meaning**
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthcertorform") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the following information: ■ Server for establishing an SSL connection. This server redirects a user's X.509 certificate over an SSL connection. ■ Name and path of the SSL and forms credentials collector (SFCC). If you are using an alternate forms-based authentication over SSL, also provide the following two pieces of information: ■ The fully qualified name of the SSL server used for establishing an SSL connection for authentication. ■ Name and path of the Forms Credentials Collector (FCC). https://*SSLserver:port*/*SFCC*?certorform; [https://*BasicServer*/*FCC*] The following example uses the default SCC values: https://my.SSLserver.com:80/siteminderagent/certoptional/forms/login.sfcc?certorform; https://my.BasicServer.com/siteminderagent/forms/login.fcc
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.

Information Type	Value Assignment and Meaning
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme is not used to authenticate administrators.
Save credentials?	SaveCredentials(0) CreateAuthScheme() param: *saveCreds* Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(*flag*) CreateAuthScheme() param: *ignorePwd* Set to 1 to ignore password checking, or 0 to check passwords. Default is 0.

X.509 Client Cert Template

Use this table when configuring an authentication scheme based on the scheme type X.509 Client Certificate.

Information Type	Value Assignment and Meaning
Scheme type	Type(*templateObject*) CreateAuthScheme() param: *schemeTemplate* The scheme type X.509 Client Certificate.
Description	Description(*schemeDesc*) CreateAuthScheme() param: *schemeDesc* The description of the authentication scheme.
Protection level	ProtectionLevel(*nLevel*) CreateAuthScheme() param: *protLevel* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	CustomLib("smauthcert") CreateAuthScheme() param: *schemeLib* The default library for this scheme type.

Information Type	**Value Assignment and Meaning**
Parameter	CustomParam(*param*) CreateAuthScheme() param: *schemeParam* A string containing the domain or IP address of the server responsible for establishing the SSL connection and the name and path of the SSL Credentials Collector (SCC). The server redirects a user's X.509 certificate over an SSL connection. Format: https://*server*/*SCC*?cert The following example uses the default SCC value: https://my.server.com/siteminderagent/ cert/smgetcred.scc?cert
Shared secret	CustomSecret("") CreateAuthScheme() param: *secret* Set to an empty string. Not applicable to this scheme.
Is template?	IsTemplate(*templateFlag*) CreateAuthScheme() param: *isTemplate* Set to 0 to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	IsUsedByAdmin(0) CreateAuthScheme() param: *isUsedByAdmin* Set to 0—scheme is not used to authenticate administrators.
Save credentials?	setAllowSaveCreds(0) Set to 0 to indicate that user credentials will not be saved.
Is RADIUS?	IsRadius(0) CreateAuthScheme() param: *isRadius* Set to 0—scheme is not used with RADIUS agents.
Ignore password check?	IgnorePwd(1) CreateAuthScheme() param: *ignorePwd* Set to 1—ignore password checking.

Chapter 6: CLI Policy Management Methods

This section contains the following topics:

Administrator Methods

The following methods act on PolicyMgtAdmin objects:

- AuthScheme Method—Sets or retrieves the authentication scheme for an administrator stored in an external directory
- Description Method—Sets or retrieves the description of the administrator
- Name Method—Sets or retrieves the name of the administrator
- ManageAllDomains Method—Adds or revokes the administrator's authority to manage domains
- ManageDomainObjects Method—Adds or revokes the administrator's authority to manage domain objects
- ManageKeysAndPwdPolicy Method—Adds or revokes the administrator's authority to manage keys and password policies
- ManageUsers Method—Adds or revokes the administrator's authority to manage users
- Password Method—Sets or retrieves the administrator password
- UserDirectory Method—Sets or retrieves an external user directory for the administrator

AuthScheme Method—Sets or Retrieves an Authentication Scheme

The AuthScheme method sets or retrieves the authentication scheme for an administrator stored in an external directory.

Syntax

The AuthScheme method has the following format:

```
Netegrity::PolicyMgtAdmin->AuthScheme( [authScheme] )
```

Parameters

The AuthScheme method accepts the following parameter:

authScheme (PolicyMgtAuthScheme)

(Optional) Specifies the authentication scheme to set.

Return Value

The AuthScheme method returns one of the following values:

- A PolicyMgtAuthScheme object
- **undef** if no authentication scheme exists, or if the call was unsuccessful

Description Method—Sets or Retrieves the Description of an Administrator

The Description method sets or retrieves the description of the administrator.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtAdmin->Description( [adminDesc] )
```

Parameters

The Description method accepts the following parameter:

adminDesc (string)

(Optional) Specifies the description of the administrator.

Return Value

The Description method returns one of the following values:

- The new or existing administrator directory description
- An empty string if unsuccessful

ManageAllDomains Method—Grants or Revokes Privileges to Manage Policy Server Objects

The ManageAllDomains method grants or revokes privileges to manage all system-level and domain-level Policy Server objects.

Syntax

The ManageAllDomains method has the following format:

```
Netegrity::PolicyMgtAdmin->ManageAllDomains( [allDomFlag] )
```

Parameters

The ManageAllDomains method accepts the following parameter:

allDomFlag (int)

(Optional) Specifies whether system-level and domain-level privileges are enable (set to a value of 1), or disabled (set to a value of 0).

Return Value

The ManageAllDomains method returns one of the following values:

- 1 if the administrator can manage all domains
- 0 if the administrator cannot manage all domains

Remarks

Privileges include:

- Management of system-level Policy Store objects such as administrators, agents, directories, policy domains, authentication schemes, registration schemes, ODBC query schemes, and password policies
- Management of agent groups, directory mappings, and certificate mappings

Note: These objects cannot be managed through the Scripting Interface.

- All of the domain-level privileges granted through the ManageDomainObjects method

ManageDomainObjects Method—Grants or Revokes Privileges to Manage Domain Objects

The ManageDomainObjects method grants or revokes privileges to manage domain-level Policy Server objects.

Syntax

The ManageDomainObjects method has the following format:

```
Netegrity::PolicyMgtAdmin->ManageDomainObjects( [domFlag] )
```

Parameters

The ManageDomainObjects method accepts the following parameter:

domFlag (int)

(Optional) Specifies whether domain object management privileges are granted (set to a value of 1), or revoked (set to a value of 0).

Return Value

The ManageDomainObjects method returns one of the following values:

- 1 if the administrator can manage domain objects
- 0 if the administrator cannot manage domain objects

Remarks

Privileges include:

- Management of rules, responses, policies, and realms
- Management of rule and response groups

Note: These objects cannot be managed through the Scripting Interface.

- Flushing of realms from the resource cache

ManageKeysAndPwdPolicy Method—Grants or Revokes Privileges to Manage Keys and Password Policies

The ManageKeysAndPwdPolicy method grants or revokes administrator privileges to manage keys and password policies.

Syntax

The ManageKeysAndPwdPolicy method has the following format:

```
Netegrity::PolicyMgtAdmin->ManageKeysAndPwdPolicy( [pwdPolFlag] )
```

Parameters

The ManageKeysAndPwdPolicy method accepts the following parameters:

pwdPolFlag (int)

(Optional) Specifies granting or revoking privileges. Setting this flag to 1 has different meanings for different types of administrators:

- System-level administrators will be able to manage both keys and password policies.
- Domain-level administrators will be able to manage password policies only.

Note: You can only create system-level administrators with the Command Line Interface. To create a domain-level administrator, use the Administrative UI.

Setting this flag to 0 revokes these privileges.

Return Value

The ManageKeysAndPwdPolicy method returns one of the following values:

- 1 privileges enabled
- 0 privileges disabled

ManageUsers Method—Grants or Revokes Privileges to Manage Users

The ManageUsers method grants or revokes privileges to manage users.

Syntax

The ManageUsers method has the following format:

```
Netegrity::PolicyMgtAdmin->ManageUsers( [userFlag] )
```

Parameters

The ManageUsers method accepts the following parameter:

userFlag (int)

(Optional) Specifies whether to grant (set value to 1) or revoke (set value to 0) user management privileges.

Return Value

The ManageUsers method returns one of the following values:

- 1 if the administrator can manage users
- 0 if the administrator cannot manage users

Name Method—Sets or Retrieves the Name of an Administrator

The Name method sets or retrieves the name of the administrator.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtAdmin->Name( [adminName] )
```

Parameters

The Name method accepts the following parameter:

adminName (string)

(Optional) Specifies the name of the administrator.

Return Value

The Name method returns one of the following values:

- The new or existing administrator name
- **undef** if the call was unsuccessful

Password Method—Sets or Retrieves the Administrator Password

The Password method sets or retrieves the administrator password.

Syntax

The Password method has the following format:

```
Netegrity::PolicyMgtAdmin->Password([adminPwd])
```

Parameters

The Password method accepts the following parameter:

adminPwd (string)

(Optional) Specifies the administrator password.

Return Value

The Password method returns one of the following values:

- The new or existing administrator password
- **undef** if the call was unsuccessful

UserDirectory Method—Sets or Retrieves an External User Directory

The UserDirectory method sets or retrieves an external user directory for the administrator.

Syntax

The UserDirectory method has the following format:

```
Netegrity::PolicyMgtAdmin->UserDirectory([userDir])
```

Parameters

The UserDirectory method accepts the following parameter:

userDir (PolicyMgtUserDir)

(Optional) Specifies the external user directory.

Return Value

The UserDirectory method returns one of the following values:

- A PolicyMgtUserDir object
- **undef** if no directory exists, or if the call was unsuccessful

Affiliate Attribute Methods

The following methods act on PolicyMgtAffiliateAttr objects:

- GetAttrType Method—Retrieves the type of the affiliate attribute
- GetValue Method—Retrieves the value of the affiliate attribute

GetAttrType Method—Retrieves the Affiliate Attribute Type

The GetAttrType method retrieves the type of the affiliate attribute.

Syntax

The GetAttrType method has the following format:

```
Netegrity::PolicyMgtAffiliateAttr->GetAttrType( )
```

Parameters

The GetAttrType method accepts no parameters.

Return Value

The GetAttrType method returns one of the following values:

- AFFILIATE_HTTP_HEADER_VARIABLE
- AFFILIATE_HTTP_COOKIE_VARIABLE

GetValue Method—Retrieves the Value of the Affiliate Attribute

The GetValue method retrieves the value of the affiliate attribute.

Syntax

The GetValue method has the following format:

```
Netegrity::PolicyMgtAffiliateAttr->GetValue( )
```

Parameters

The GetValue method accepts no parameters.

Return Value

The GetValue method returns one of the following values:

- The value of the affiliate attribute
- **undef** if the call was unsuccessful

Affiliate Domain Methods

The following methods act on PolicyMgtAffDomain objects:

- AddAdmin Method—Associates an administrator with the affiliate domain
- AddUserDir Method—Associates a user directory with the affiliate domain
- CreateAffiliate Method—Creates an affiliate object within the affiliate domain
- CreateSAMLServiceProvider Method—Creates a SAML 2.0 Service Provider within the affiliate domain
- CreateWSFEDResourcePartner Method—Creates a WS-Federation Resource Partner within the affiliate domain
- DeleteAffiliate Method—Deletes an affiliate object from the affiliate domain
- DeleteSAMLServiceProvider Method—Deletes a Service Provider from the affiliate domain

- DeleteWSFEDResourcePartner Method—Deletes a Resource Partner
- Description Method—Sets or retrieves the description of the affiliate domain object
- GetAffiliate Method—Retrieves the specified affiliate object
- GetAllAdmins Method—Retrieves all administrators associated with the affiliate domain
- GetAllAffiliates Method—Retrieves all affiliate objects associated with the affiliate domain
- GetAllSAMLServiceProviders Method—Retrieves all Service Providers in the affiliate domain
- GetAllWSFEDResourcePartners Method—Retrieves all Resource Partners in the affiliate domain
- GetSAMLServiceProvider Method—Retrieves the specified Service Provider
- GetSAMLServiceProviderById Method—Retrieves the Service Provider specified by its provider ID
- GetUserDirSearchOrder Method—Retrieves user directory objects associated with the affiliate domain
- GetWSFEDResourcePartner Method—Retrieves the specified Resource Partner
- GetWSFEDResourcePartnerById Method—Retrieves the Resource Partner specified by its ID
- Name Method—Sets or retrieves the name of the affiliate domain
- RemoveAdmin Method—Disassociates the user directory from the affiliate domain
- RemoveUserDir Method—Disassociates the administrator from the affiliate domain
- SetUserDirSearchOrder—Rearranges the search order of the user directory objects associated with the affiliate domain

AddAdmin Method—Associates an Administrator with an Affiliate Domain

The AddAdmin method associates an administrator with an affiliate domain.

Syntax

The AddAdmin method has the following format:

```
Netegrity::PolicyMgtAffDomain->AddAdmin(admin)
```

Parameter

The AddAdmin method accepts the following parameter:

admin (PolicyMgtAdmin)

Specifies the administrator to associate with the affiliate domain.

Return Values

The AddAdmin method returns one of the following values:

0 on success

-1 on failure

AddUserDir Method—Associates a User Directory with an Affiliate Domain

The AddUserDir method associates a user directory with an affiliate domain.

Syntax

The method has the following format:

```
Netegrity::PolicyMgtAffDomain->AddUserDir(userDir)
```

Parameter

The AddUserDir method accepts the following parameter:

userDir (PolicyMgtUserDir)

Specifies the user directory to associate with the affiliate domain.

Return Values

The AddUserDir method returns one of the following values:

0 on success

-1 on failure

CreateAffiliate Method—Creates an Affiliate Object

The CreateAffiliate method creates and configures an affiliate object within an affiliate domain.

Syntax

The CreateAffiliate method has the following format:

```
Netegrity::PolicyMgtAffDomain->CreateAffiliate( affName, password, authURL,
validityDuration, skewTime [, affDesc] [, allowNotification] [, audience] [,
enableFlag] [, shareSessioning] [, sessionSyncInterval] [, SAMLVersion] [,
SAMLProfile] [,ConsumerURL] )
```

Parameters

The CreateAffiliate method accepts the following parameters:

affName (string)

Specifies the name of the affiliate object. The name should be unique across all affiliate domains.

password (string)

Specifies the password that affiliates use to access SiteMinder Federation Web Services.

authURL (string)

Specifies the URL used to authenticate affiliate users.

validityDuration (long)

Specifies the number of seconds that a SiteMinder-generated SAML assertion is valid. If an affiliate receives the assertion after the specified time, the assertion is considered invalid.

skewTime (long)

Specifies the difference, in seconds, between the system clock time of the assertion producer site and the system clock time of the affiliate site. The skew time is added to validityDuration. Times are relative to GMT.

affDesc (string)

(Optional) Specifies the description of the affiliate.

allowNotification (int)

(Optional) Specifies whether to allow event notifications. Set to 1 to enable event notifications to be sent from the affiliate to SiteMinder on the assertion producer site. Set to 0 to disable the event notification service. Default is 0 (notifications disabled).

audience (string)

(Optional) Specifies the URI of the document that describes the agreement between the assertion producer and the affiliate. This value is included in the SAML assertion passed to the affiliate and can be used for validation purposes. Also, the affiliate can parse the audience document to obtain relevant information. The audience value must match the Assertion Audience setting in the AffiliateConfig.xml configuration file for the SAML Affiliate Agent.

enableFlag (int)

(Optional) Specifies whether to enable the affiliate object. Set to 1 to enable the affiliate object, or 0 to disable it. Default is 1 (object is enabled).

shareSessioning (int)

(Optional) Specifies whether to share session information. Set to 1 to allow the assertion producer and the affiliate to share session information, or set to 0 to have the producer and affiliate maintain separate sessions. Default is 0 (separate sessions). With shared sessions, the sessions on both sites are terminated when the session on either site ends.

sessionSyncInterval (long)

(Optional) Specifies the frequency, in seconds, at which the affiliate contacts the producer site to validate the status of a shared session.

SAMLVersion (long)

(Optional) Specifies the SAML version. One of the following values:

- AFFILIATE_SAML_VER_1_0
- AFFILIATE_SAML_VER_1_1

Specifying a SAML version has effect only if the Policy Manager API's session version is at least v6.0 SP 1.

SAMLProfile (long)

(Optional) Specifies the type of profile used to send and receive SAML assertions. Valid profiles:

AFFILIATE_SAML_PROFILE_ARTIFACT. The SAML assertion is retrieved from a URL associated with the assertion producer. The URL is specified during configuration of the SAML Artifact authentication scheme.

AFFILIATE_SAML_PROFILE_POST. The generated SAML assertion is POSTed to the URL specified in ConsumerURL.

This profile is supported only if the Policy Management API's session version is at least v6.0 SP 2. If an earlier version is involved, the POST profile request is ignored, and an attempt is made to create an affiliate object based on the artifact profile.

ConsumerURL (string)

(Optional) Specifies the URL where the requesting user's browser must POST a generated assertion. The site associated with the URL validates the assertion and uses its contents to make access decisions.

Return Value

The CreateAffiliate method returns one of the following values:

PolicyMgtAffiliate object if successful

undef if unsuccessful

Remarks

An affiliate object represents an affiliate site in a federated business network. Affiliate objects and affiliate domains are available through SiteMinder legacy federation.

CreateSAMLServiceProvider Method—Creates a SAML Service Provider

The CreateSAMLServiceProvider method creates a SAML 2.0 Service Provider object. A Service Provider object contains information that an Identity Provider needs to produce assertions for the Service Provider. The properties you can set for a SAML 2.0 Service Provider object are listed following.

To modify the properties of an existing Service Provider, call the PolicyMgtSAMLServiceProvider->Property method.

Syntax

The CreateSAMLServiceProvider method has the following format:

```
Netegrity::PolicyMgtAffDomain->CreateSAMLServiceProvider(propsHash_ref)
```

Parameters

The CreateSAMLServiceProvider method accepts the following parameter:

propsHash_ref (hash)

Specifies a reference to a hashtable of metadata properties to define for the SAML 2.0 Service Provider (for example: \%myhash).

Return Values

The CreateSAMLServiceProvider method returns one of the following values:

- A PolicyMgtSAMLServiceProvider object on success
- **undef** on failure

Remarks

You can define the following properties for a SAML 2.0 Service Provider:

General Properties

- SAML_NAME
- SAML_DESCRIPTION
- SAML_SP_AUTHENTICATION_URL
- SAML_ENABLED
- SAML_SP_DOMAIN
- SAML_KEY_SPID
- SAML_SP_IDPID
- SAML_MAJOR_VERSION
- SAML_MINOR_VERSION
- SAML_SKEWTIME
- SAML_DISABLE_SIGNATURE_PROCESSING
- SAML_DSIG_VERINFO_ISSUER_DN
- SAML_DSIG_VERINFO_SERIAL_NUMBERSAML_KEY_SPID

Name ID Properties

- SAML_SP_NAMEID_FORMAT
- SAML_SP_NAMEID_TYPE
- SAML_SP_NAMEID_STATIC
- SAML_SP_NAMEID_ATTRNAME
- SAML_SP_NAMEID_DNSPEC
- SAML_AFFILIATION

SSO Properties

- SAML_AUDIENCE
- SAML_SP_ASSERTION_CONSUMER_DEFAULT_URL
- SAML_ENABLE_SSO_ARTIFACT_BINDING
- SAML_SP_ARTIFACT_ENCODING
- SAML_SP_IDP_SOURCEID
- SAML_SP_PASSWORD
- SAML_ENABLE_SSO_POST_BINDING
- SAML_SSOECPPROFILE
- SAML_SP_REQUIRE_SIGNED_AUTHNREQUESTS
- SAML_SP_AUTHENTICATION_LEVEL
- SAML_SP_AUTHN_CONTEXT_CLASS_REF
- SAML_SP_VALIDITY_DURATION
- SAML_SP_STARTTIME
- SAML_SP_ENDTIME

SLO Properties

- SAML_SLO_REDIRECT_BINDING
- SAML_SLO_SERVICE_VALIDITY_DURATION
- SAML_SLO_SERVICE_URL
- SAML_SLO_SERVICE_RESPONSE_URL
- SAML_SLO_SERVICE_CONFIRM_URL

IPD Properties

- SAML_SP_ENABLE_IPD
- SAML_SP_IPD_SERVICE_URL
- SAML_SP_COMMON_DOMAIN
- SAML_SP_PERSISTENT_COOKIE

Attribute Service Properties

- SAML_SP_ATTRSVC_ENABLE
- SAML_SP_ATTRSVC_VALIDITY_DURATION
- SAML_SP_ATTRSVC_SIGN_ASSERTION
- SAML_SP_ATTRSVC_LDAP_SEARCH_SPEC
- SAML_SP_ATTRSVC_ODBC_SEARCH_SPEC
- SAML_SP_ATTRSVC_WINNT_SEARCH_SPEC
- SAML_SP_ATTRSVC_CUSTOM_SEARCH_SPEC
- SAML_SP_ATTRSVC_AD_SEARCH_SPEC

Encryption Properties

- SAML_SP_ENCRYPT_ID
- SAML_SP_ENCRYPT_ASSERTION
- SAML_SP_ENCRYPT_BLOCK_ALGO
- SAML_SP_ENCRYPT_KEY_ALGO
- SAML_SP_ENCRYPT_CERT_ISSUER_DN
- SAML_SP_ENCRYPT_CERT_SERIAL_NUMBER

Advanced Properties

- SAML_SP_PLUGIN_CLASS
- SAML_SP_PLUGIN_PARAMS

CreateWSFEDResourcePartner Method—Creates a WS-Federation Resource Partner

The CreateWSFEDResourcePartner method creates a WS-Federation Resource Partner for the affiliate domain.

Syntax

The CreateWSFEDResourcePartner method has the following format:

```
Netegrity::PolicyMgtAffDomain->CreateWSFEDResourcePartner(propsHash_ref)
```

Parameters

The CreateWSFEDResourcePartner method accepts the following parameters:

propsHash_ref (hash)

Specifies a reference to a hashtable of metadata properties to define for the

WS-Federation Resource Partner, (for example, C<\%myhash>|"hashref".

Return Value

The CreateWSFEDResourcePartner method returns one of the following values:

- A PolicyMgtWSFEDResourcePartner object on success
- **undef** on failure

Remarks

You can define the following properties for a Resource Partner:

General Properties

- WSFED_NAME
- WSFED_DESCRIPTION
- WSFED_MAJOR_VERSION
- WSFED_MINOR_VERSION

- WSFED_SAML_MAJOR_VERSION
- WSFED_KEY_RPID
- WSFED_APID
- WSFED_SAML_MINOR_VERSION
- WSFED_RP_DOMAIN
- WSFED_ENABLED
- WSFED_RP_AUTHENTICATION_URL
- WSFED_KEY_RPID
- WSFED_APID

NameID Properties

- WSFED_RP_NAMEID_FORMAT
- WSFED_RP_NAMEID_TYPE
- WSFED_RP_NAMEID_STATIC
- WSFED_RP_NAMEID_ATTR_NAME
- WSFED_RP_NAMEID_DN_SPEC
- WSFED_RP_NAMEID_ALLOWED_NESTED

SSO Properties

- WSFED_RP_AUTHENTICATION_METHOD
- WSFED_RP_VALIDITY_DURATION
- WSFED_RP_ASSERTION_CONSUMER_DEFAULT_URL
- WSFED_RP_AUTHENTICATION_LEVEL
- WSFED_RP_STARTTIME
- WSFED_RP_ENDTIME

Signout Properties

- WSFED_RP_SLO_ENABLED
- WSFED_RP_SIGNOUT_CLEANUP_URL
- WSFED_RP_SIGNOUT_CONFIRM_URL

Advanced Properties

- WSFED_RP_PLUGIN_CLASS
- WSFED_RP_PLUGIN_PARAMS

DeleteAffiliate Method—Deletes an Affiliate from a Domain

The DeleteAffiliate method deletes the specified affiliate object from the affiliate domain.

Syntax

The DeleteAffiliate method has the following format:

```
Netegrity::PolicyMgtAffDomain->DeleteAffiliate(aff)
```

Parameters

The DeleteAffiliate method accepts the following parameters:

aff (PolicyMgtAffiliate)

Specifies the affiliate object to delete.

Return Value

The DeleteAffiliate method returns one of the following values:

- 0 on success, or if the affiliate domain was not found
- -1 on failure

DeleteSAMLServiceProvider Method—Deletes a SAML Service Provider

The DeleteSAMLServiceProvider method deletes the specified SAML 2.0 Service Provider.

Syntax

The method has the following format:

```
Netegrity::PolicyMgtAffDomain->DeleteSAMLServiceProvider(sp)
```

Parameters

The DeleteSAMLServiceProvider method accepts the following parameters:

sp (PolicyMgtSAMLServiceProvider)

Specifies the Service Provider to delete.

Return Value

The DeleteSAMLServiceProvider method returns the one of the following values:

- 0 on success, or if the Service Provider was not found
- -1 on failure

DeleteWSFEDResourcePartner Method—Deletes a Resource Partner

The DeleteWSFEDResourcePartner method deletes a resource partner.

Syntax

The DeleteWSFEDResourcePartner method has the following format:

```
Netegrity::PolicyMgtAffDomain->DeleteWSFEDResourcePartner(rp)
```

Parameters

The DeleteWSFEDResourcePartner method accepts the following parameter:

rp (PolicyMgtWSFEDResourcePartner object)

Specifies the resource partner to delete.

Return Value

The DeleteWSFEDResourcePartner method returns one of the following values:

- value = 0

  Specifies that the method is successful.
- value = -1

  Specifies that the method is unsuccessful.

Description Method—Retrieves or Sets a Description

The Description method sets or retrieves the description of the affiliate domain.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtAffDomain->Description([domainDesc])
```

Parameters

The Description method accepts the following parameters:

domainDesc (string)

(Optional) Specifies the description to set.

Return Value

The Description method returns one of the following values:

- A new or existing description of the affiliate domain on success
- **undef** on failure

GetAffiliate Method—Retrieves an Affiliate Object

The GetAffiliate method retrieves the specified affiliate object.

Syntax

The GetAffiliate method has the following format:

```
Netegrity::PolicyMgtAffDomain->GetAffiliate(affName)
```

Parameters

The GetAffiliate method accepts the following parameters:

affName (string)

Specifies the name of the affiliate object to retrieve.

Return Value

The GetAffiliate method returns one of the following objects:

- A PolicyMgtAffiliate object on success
- **undef** if the specified affiliate object does not exist, or if the call fails

GetAllAdmins Method—Retrieves all Administrators

The GetAllAdmins method retrieves all administrators associated with the affiliate domain.

Syntax

The GetAllAdmins method has the following format:

```
Netegrity::PolicyMgtAffDomain->GetAllAdmins( )
```

Parameters

The GetAllAdmins method accepts no parameters.

Return Value

The GetAllAdmins method returns one or more of the following values:

- An array of PolicyMgtAdmin objects
- **undef** if no administrator objects are associated with the affiliate domain, or if the call fails

GetAllAffiliates Method—Retrieves All Affiliates in a Domain

The GetAllAffiliates method retrieves all affiliate objects associated with the affiliate domain.

Syntax

The GetAllAffiliates method has the following format:

```
Netegrity::PolicyMgtAffDomain->GetAllAffiliates( )
```

Parameters

The GetAllAffiliates method accepts no parameters.

Return Value

The GetAllAffiliates method returns one of the following values:

- An array of PolicyMgtAffiliate objects on success
- **undef** if unsuccessful

GetAllSAMLServiceProviders Method—Retrieves all Service Providers associated with the Affiliate Domaine

The GetAllSAMLServiceProviders method retrieves all the SAML 2.0 Service Providers associated with the affiliate domain.

Syntax

The GetAllSAMLServiceProviders method has the following format:

```
Netegrity::PolicyMgtAffDomain->GetAllSAMLServiceProviders( )
```

Parameters

The GetAllSAMLServiceProviders method accepts no parameters.

Return Value

The GetAllSAMLServiceProviders method returns one of the following values:

- An array of PolicyMgtSAMLServiceProvider objects
- undef if unsuccessful

GetAllWSFEDResourcePartners Method—Retrieves all WSFED Resource Partners

The GetAllWSFEDResourcePartners method retrieves all Resource Partners associated with the domain.

Syntax

The GetAllWSFEDResourcePartners method has the following format:

```
Netegrity::PolicyMgtAffDomain->GetAllWSFEDResourcePartners( )
```

Parameters

The GetAllWSFEDResourcePartners method accepts no parameters.

Return Value

The GetAllWSFEDResourcePartners method returns one of the following values:

- An array of PolicyMgtWSFEDResourcePartner objects on success
- **undef** on failure

GetSAMLServiceProvider Method—Retrieves a Specified Service Provider

The GetSAMLServiceProvider method retrieves the specified SAML 2.0 Service Provider.

Syntax

The GetSAMLServiceProvider method has the following format:

```
Netegrity::PolicyMgtAffDomain->GetSAMLServiceProvider(spName)
```

Parameters

The GetSAMLServiceProvider method accepts the following parameter:

spName (string)

Specifies the name of the Service Provider to retrieve.

Return Value

The GetSAMLServiceProvider method returns one of the following values:

- A PolicyMgtSAMLServiceProvider object on success
- undef if the specified Service Provider does not exist, or if the call is unsuccessful

GetSAMLServiceProviderByID Method—Retrieves a Specified Service Provider

The GetSAMLServiceProviderById method retrieves the specified SAML 2.0 Service Provider by its provider ID.

Syntax

The GetSAMLServiceProviderById method has the following format:

```
Netegrity::PolicyMgtAffDomain->GetSAMLServiceProviderById(spID)
```

Parameters

The GetSAMLServiceProviderById method accepts the following parameter:

spID (string)

Specifies the provider ID of the Service Provider to retrieve.

Return Value

The GetSAMLServiceProviderById method returns one of the following values:

- A PolicyMgtSAMLServiceProvider object on success
- **undef** if the specified Service Provider does not exist, or if the call is unsuccessful

GetUserDirSearchOrder Method—Retrieves Search Order of a User Directory

The GetUserDirSearchOrder method retrieves user directory objects associated with the affiliate domain. The order of the returned objects is the same order SiteMinder uses when querying the directories.

Syntax

The GetUserDirSearchOrder method has the following format:

```
Netegrity::PolicyMgtAffDomain->GetUserDirSearchOrder( )
```

Parameters

The GetUserDirSearchOrder method accepts no parameters.

Return Value

The GetUserDirSearchOrder method returns one of the following values:

- An array of PolicyMgtUserDir objects on success
- **undef** if unsuccessful

GetWSFEDResourcePartner Method—Retrieves Resource Partner

The GetWSFEDResourcePartner method retrieves the specified WS-Federation Resource Partner associated with the domain.

Syntax

The GetWSFEDResourcePartner method has the following format:

```
Netegrity::PolicyMgtAffDomain->GetWSFEDResourcePartner(rpName)
```

Parameters

The GetWSFEDResourcePartner method accepts the following parameters:

rpName (string)

Specifies the name of the Resource Partner to retrieve.

Return Value

The GetWSFEDResourcePartner method returns the following value:

- A PolicyMgtWSFEDResourcePartner object on success
- **undef** if the specified Resource Partner does not exist, or if the call is unsuccessful

GetWSFEDResourcePartnerById Method—Retrieves Resource Partner by ID

The GetWSFEDResourcePartnerById method retrieves the specified Resource Partner by its Provider ID.

Syntax

The GetWSFEDResourcePartnerById method has the following format:

```
Netegrity::PolicyMgtAffDomain->GetWSFEDResourcePartnerById( rpID )
```

Parameters

The GetWSFEDResourcePartnerById method accepts the following parameters:

rpID (string)

Specifies the ID of the Resource Partner to retrieve.

Return Value

The GetWSFEDResourcePartnerById method returns the following value:

- A PolicyMgtWSFEDResourcePartner object
- **undef** if the specified Resource Partner does not exist, or if the call is unsuccessful

Name Method—Sets or Retrieves Affiliate Domain Name

The Name method sets or retrieves the name of the affiliate domain.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtAffDomain->Name( [domainName] )
```

Parameters

The Name method accepts the following parameter:

domainName (string)

(Optional) Specifies the name to set.

Return Value

The Name method returns one of the following values:

- New or existing affiliate domain name
- **undef** if the call was unsuccessful

RemoveAdmin Method—Dissasocciates an Administrator from an Affiliate Domain

The RemoveAdmin method disassociates the specified administrator from the affiliate domain.

Syntax

The RemoveAdmin method has the following format:

```
Netegrity::PolicyMgtAffDomain->RemoveAdmin( admin )
```

Parameters

The RemoveAdmin method accepts the following parameter:

admin (PolicyMgtAdmin)

Specifies the administrator to remove from the affiliate domain.

Return Value

The RemoveAdmin method returns one of the following values:

- 0 on success
- -1 on failure

RemoveUserDir Method—Disassociates a User Directory from an Affiliate Domain

The RemoveUserDir method disassociates the user directory from the affiliate domain.

Syntax

The RemoveUserDir method has the following format:

```
Netegrity::PolicyMgtAffDomain->RemoveUserDir( userDir )
```

Parameters

The RemoveUserDir method accepts the following parameter:

userDir (PolicyMgtUserDir)

Specifies the user directory to disassociate from the affiliate domain.

Return Value

The RemoveUserDir method returns one of the following values:

- 0 on success
- -1 on failure

SetUserDirSearchOrder Method—Sets the Order for Searching Directory Objects

The SetUserDirSearchOrder method rearranges the search order of the user directory objects associated with the affiliate domain.

Syntax

The SetUserDirSearchOrder method has the following format:

```
Netegrity::PolicyMgtAffDomain->SetUserDirSearchOrder( dirArray )
```

Parameters

The SetUserDirSearchOrder method accepts the following parameter:

dirArray (PolicyMgtUserDir)

Specifies a reference to an array of user directory objects (for example: \@myarray).

Return Value

The SetUserDirSearchOrder method returns the following value:

- Array of (PolicyMgtUserDir) objects on success
- **undef** if unsuccessful

Affiliate Object Methods

The following methods act on PolicyMgtAffiliate objects:

- AddAttribute Method—Adds a new affiliate attribute to the affiliate object
- AddUser Method—Adds a new user to the affiliate object

- AllowNotification Method—Sets or retrieves the event notification property
- AssertionPluginClass Method—Sets or retrieves the fully qualified class name of a custom assertion generator plug-in
- AssertionPluginParameters Method—Sets or retrieves the parameter string to pass to a custom assertion generator plug-in
- Audience Method—Sets or retrieves the audience property
- AuthURL Method—Sets or retrieves the URL used to authenticate affiliate users
- ConsumerURL Method—Sets or retrieves the URL where the requesting user's browser must POST a generated assertion
- CreateIPConfigHostName Method—Creates an IP configuration object from the specified host name
- CreateIPConfigRange Method—Creates an IP configuration object from the specified range of IP addresses
- CreateIPConfigSingleHost Method—Creates an IP configuration object from the specified IP address
- CreateIPConfigSubnetMask Method—Creates an IP configuration object from the specified IP address and subnet mask
- DeleteIPConfig Method—Deletes an IP configuration object
- Description Method—Sets or retrieves the description of the affiliate object
- GetAllAttributes Method—Retrieves all existing affiliate attributes for the affiliate object
- GetAllIPConfigs Method—Retrieves all IP configuration objects for the affiliate object
- GetAllUsers Method—Retrieves all users associated with affiliate object
- IsEnabled Method—Enables or disables the affiliate object
- Name Method—Sets or retrieves the name of the affiliate object
- Password Method—Sets or retrieves the password property
- RemoveAttribute—Removes the specified affiliate attribute from the affiliate object
- RemoveUser—Removes the specified user from the affiliate object
- SAMLProfile—Sets or retrieves the type of profile used for sending and receiving SAML assertions
- SAMLVersion—Sets or retrieves the affiliate’s SAML version
- Save—Saves modifications to an affiliate object
- SessionSyncInterval—Sets or retrieves the session synchronization interval property

- ShareSessioning—Sets or retrieves the shared session property
- SkewTime—Sets or retrieves the skew time property
- ValidityDuration—Sets or retrieves the validity duration property

AddAttribute Method—Adds a New Affiliate Attribute

The AddAttribute method Adds a new affiliate attribute to the affiliate object.

Syntax

The AddAttribute method has the following format:

```
Netegrity::PolicyMgtAffiliate->AddAttribute(attrType, value)
```

Parameters

The AddAttribute method accepts the following parameters:

attrType (int)

Specifies one of the following affiliate attribute types:

- AFFILIATE_HTTP_HEADER_VARIABLE (Value=1). The affiliate attribute is made available as an HTTP header variable.
- AFFILIATE_HTTP_COOKIE_VARIABLE (Value=2). The affiliate attribute is made available as an HTTP cookie variable.

value (int)

Specifies the value for the affiliate attribute. This value specification appears in the Name Value Pair column of the SiteMinder Affiliate Dialog. The format of the value specification depends upon the kind of affiliate attribute you are adding -- Static, User Attribute, or DN Attribute:

Static. A literal attribute value. A static affiliate attribute is useful for passing specific information about the user to an application at the affiliate site -- for example, the user's credit limit at the affiliate site.

- Format: VariableName=value

  VariableName is the name that identifies the attribute in the SAML assertion, and value is the attribute value.

- Example: climit=2000

User Attribute. A user profile attribute name from a user's entry in an LDAP, WinNT, or ODBC user directory -- for example, the attribute name for a user's job title or email address.

- Format: UserAttrVariableName=<%userattr="UserAttrName"%>

  UserAttrVariableName is the name that identifies the attribute in the SAML assertion, and UserAttrName (enclosed in quotes) is the name of the attribute in the user directory.

  userattr= is static text that must be included in the format. The userattr= portion of the name/value pair must be enclosed by percent signs and angle brackets: <% . . . %>.

  Example: email_address=<%userattr="email"%>

DN Attribute. The name of an attribute within an LDAP or ODBC directory object that is associated with the user. Groups to which a user belongs and Organizational Units (ou) that are part of a user DN are examples of directory objects whose attributes can be referenced as DN attributes. For example, a DN attribute can reference a company division for a user, based on the user's membership in a division.

- Format: DNVariableName=<#dn="DNSpec" attr="DNAttrName"#>

  DNVariableName is the name that identifies the attribute in the SAML assertion. DNSpec (enclosed in quotes) is the DN of the directory object, and DNAttrName (enclosed in quotes) is the name of the directory object attribute.

  dn= and attr= are static text strings that must be included in the format. The dn= and attr= portion of the name/value pair must be enclosed by pound signs and angle brackets: <# . . . #>.

  Example: GroupName=<#dn="ou=home,o=security.com" attr="cn"#>

  To allow SiteMinder to retrieve DN attributes from a nested group, begin DNSpec with an exclamation mark (!) -- for example:

  dn="!ou=home,o=security.com"

Return Value

The AddAttribute method returns the following value:

- A PolicyMgtAffiliateAttr object
- **undef** if unsuccessful

Remarks

Affiliate attributes are name/value pairs that SiteMinder provides to an affiliate in a SAML assertion. Attributes include user entitlements (such as the user's credit limit at the affiliate site) and information from a user's profile (such as job title or email address).

When an application at the affiliate site extracts affiliate attributes from the assertion, it can make the attributes available to other applications at the site as HTTP header variables or HTTP cookie variables.

Note: The total size of an assertion passed to an affiliate cannot exceed 4K. If you include a large number of attributes in an affiliate object, you may violate this limit. A maximum assertion size of 3K is recommended.

AddUser Method—Adds a New User to the Affiliate Object

The AddUser method adds a new user to the affiliate object.

Syntax

The AddUser method has the following format:

```
Netegrity::PolicyMgtAffiliate->AddUser( user )
```

Parameters

The AddUser method accepts the following parameter:

user (PolicyMgtUser)

Specifies the user to add.

Return Value

The AddUser method returns one of the following values:

- 0 on success
- -1 on failure

AllowNotification Method—Sets or Retrieves the Event Notification Property

The AllowNotification method sets or retrieves the event notification property. If event notifications are enabled, the affiliate sends notifications about user activities to SiteMinder on the assertion producer site.

Syntax

The AllowNotification method has the following format:

```
Netegrity::PolicyMgtAffiliate->AllowNotification( [notificationFlag] )
```

Parameters

The AllowNotification method accepts the following parameter:

notificationFlag (int)

(Optional) Specifies whether to enable event notification: 1 means to enable event notification; 0 means to disable event notifications.

Return Value

The AllowNotification method returns one of the following values:

- The new or existing notification flag setting
- **undef** if unsuccessful

AssertionPluginClass Method—Sets or Retrieves the Name of an Assertion Generator Plug-in

The AssertionPluginClass method sets or retrieves the fully qualified class name of an assertion generator plug-in.

Syntax

The AssertionPluginClass method has the following format:

```
Netegrity::PolicyMgtAffiliate->AssertionPluginClass( [className] )
```

Parameters

The AssertionPluginClass method accepts the following parameter:

className (string)

(Optional) Specifies the fully qualified class name of the custom assertion generator plug-in, for example, com.samlproducer.assertionplugin.partner1.

Return Value

The AssertionPluginClass method returns one of the following values:

- The new or existing class name
- **undef** if the plug-in is not defined, or the call was unsuccessful

Remarks

The plug-in is a custom Java class that lets you modify the contents of a default SAML assertion generated by SiteMinder. SAML assertions are available in legacy federation, which is licensed separately.

The assertion generator plug-in functionality requires a Policy Management API session version of at least v6.0 SP 2. You can pass a parameter string into the assertion generator plug-in through the method PolicyMgtAffiliate->AssertionPluginParameters.

To create an assertion generator plug-in, implement the AssertionGeneratorPlugin interface in the Java SDK. For information, see the *Programming Guide for Java*.

AssertionPluginParameters Method—Sets or Retrieves a Parameter String

The AssertionPluginParameters method sets or retrieves the parameter string to pass to a custom assertion generator plug-in. The syntax of the parameter string is user-defined--that is, the parameter string must conform to whatever conventions that the custom assertion generator requires.

Syntax

The AssertionPluginParameters method has the following format:

```
Netegrity::PolicyMgtAffiliate->AssertionPluginParameters( [parameter] )
```

Parameters

The AssertionPluginParameters method accepts the following parameter:

parameters (string)

(Optional) Specifies the parameter string to pass to the plug-in.

Return Value

The AssertionPluginParameters method returns one of the following values:

- A new or existing parameter string
- **undef** if the call was unsuccessful

Audience Method—Sets or Retrieves a URI

The Audience method sets or retrieves the URI of the document that describes the agreement between the assertion producer site and the affiliate.

This value is included in the SAML assertion passed to the affiliate and can be used for validation purposes. Also, the affiliate can parse the audience document to obtain relevant information. The audience value must match the Assertion Audience setting in the AffiliateConfig.xml configuration file for the SAML Affiliate Agent.

Syntax

The Audience method has the following format:

```
Netegrity::PolicyMgtAffiliate->Audience( [audience] )
```

Parameters

The Audience method accepts the following parameter:

audience (string)

(Optional) Specifies the audience URI to set.

Return Value

The Audience method returns one of the following values:

- A new or existing audience URI
- **undef** it the call was unsuccessful

AuthURL Method—Sets or Retrieves a URL

The AuthURL method sets or retrieves the URL used to authenticate affiliate users.

Syntax

The AuthURL method has the following format:

```
Netegrity::PolicyMgtAffiliate->AuthURL( [AuthURL] )
```

Parameters

The AuthURL method accepts the following parameter:

AuthURL (string)

(Optional) Specifies the authentication URL to set.

Return Value

The AuthURL method returns one of the following values:

- A new or existing URL
- **undef** if the call was unsuccessful

ConsumerURL Method—Sets or Retrieves a URL

The ConsumerURL method sets or retrieves the URL where the requesting user's browser must POST a generated assertion.

Syntax

The ConsumerURL method has the following format:

```
Netegrity::PolicyMgtAffiliate->ConsumerURL( [ConsumerURL] )
```

Parameters

The ConsumerURL method accepts the following parameter:

ConsumerURL (string)

(Optional) Specifies the URL where the generated assertion is to be sent.

Return Value

The ConsumerURL method returns one of the following values:

- A new or existing URL where the generated assertion is to be sent
- **undef** if the call was unsuccessful

CreateIPHostConfigName Method—Creates an IP Configuration Object from the Specified Host Name

The CreateIPHostConfigName method Creates an IP configuration object from the specified host name.

Syntax

The CreateIPHostConfigName method has the following format:

```
Netegrity::PolicyMgtAffiliate->CreateIPConfigHostName( hostName )
```

Parameters

The CreateIPHostConfigName method accepts the following parameter:

hostName (string)

Specifies the host name upon which to base the IP configuration object.

Return Value

The CreateIPHostConfigName method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** if the call was unsuccessful

Remarks

Only those users who access the affiliate site from the specified host will be accepted at the affiliate site.

CreateIPConfigRange Method—Creates an IP Configuration Object

The CreateIPConfigRange method creates an IP configuration object from the specified range of IP addresses.

Syntax

The CreateIPConfigRange method has the following format:

```
Netegrity::PolicyMgtAffiliate->CreateIPConfigRange( ipAddr1, ipAddr2 )
```

Parameters

The CreateIPConfigRange method accepts the following parameters:

ipAddr1 (string)

Specifies the first IP address in the range of valid IP addresses from which to access the affiliate site.

ipAddr2 (int)

Specifies the last IP address in the range of valid IP addresses from which to access the affiliate site.

Return Value

The CreateIPConfigRange method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** it the call is unsuccessful

Remarks

Only those users who access the affiliate site from an IP address within the specified range are accepted at the affiliate site.

CreateIPConfigSingleHost Method—Creates an IP Configuration Object from the Specified IP Address

The CreateIPConfigSingleHost method creates an IP configuration object from the specified IP address.

Syntax

The CreateIPConfigSingleHost method has the following format:

```
Netegrity::PolicyMgtAffiliate->CreateIPConfigSingleHost( ipAddr )
```

Parameters

The CreateIPConfigSingleHost method accepts the following parameter:

ipAddr (string)

Specifies the IP address from which to access the affiliate site.

Return Value

The CreateIPConfigSingleHost method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** it the call was unsuccessful

Remarks

Only those users who access the affiliate site from the specified IP address are accepted at the affiliate site.

CreateIPConfigSubnetMask Method—Creates an IP Configuration Object

The CreateIPConfigSubnetMask method creates an IP configuration object from the specified IP address and subnet mask.

Syntax

The CreateIPConfigSubnetMask method has the following format:

```
Netegrity::PolicyMgtAffiliate->CreateIPConfigSubnetMask( ipAddr, subnetMask )
```

Parameters

The CreateIPConfigSubnetMask method accepts the following parameters:

ipAddr (string)

Specifies the IP address used to derive the subnet address.

subnetMask (unsigned long)

Specifies the subnet mask used to derive the subnet address.

Return Value

The CreateIPConfigSubnetMask method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** if the call was unsuccessful

Remarks

Only those users who access the affiliate site from the subnet address will be accepted at the affiliate site. The subnet address is derived from the passed IP address and subnet mask.

DeleteIPConfig Method—Deletes an IP Configuration Object

The DeleteIPConfig method deletes the specified IP configuration object.

Syntax

The DeleteIPConfig method has the following format:

```
Netegrity::PolicyMgtAffiliate->DeleteIPConfig( IPConfig )
```

Parameters

The DeleteIPConfig method accepts the following parameter:

IPConfig (PolicyMgtIPConfig)

Specifies the IP configuration object to delete.

Return Value

The DeleteIPConfig method returns one of the following values:

- 0 on success
- -1 if the call fails

Description Method—Sets or Retrieves the Description of an Affiliate Object

The Description method sets or retrieves the description of the affiliate object.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtAffiliate->Description( [affDesc] )
```

Parameters

The Description method accepts the following parameter:

affDesc (string)

(Optional) Specifies the description to set.

Return Value

The Description method returns one of the following values:

- The new or existing description of the affiliate object.
- **undef** if the call was unsuccessful

GetAllAttributes Method—Retrieves Attributes for an Affiliate Object

The GetAllAttributes method retrieves all existing affiliate attributes for the affiliate object.

Syntax

The GetAllAttributes method has the following format:

```
Netegrity::PolicyMgtAffiliate->GetAllAttributes( )
```

Parameters

The GetAllAttributes method accepts no parameters.

Return Value

The GetAllAttributes method returns one of the following values:

- An array of PolicyMgtAffiliateAttr objects
- **undef** if the call was unsuccessful

GetAllIPConfigs Method—Retrieves All IP Configuration Objects for an Affiliate

The GetAllIPConfigs method retrieves all IP configuration objects for the affiliate object.

Syntax

The GetAllIPConfigs method has the following format:

```
Netegrity::PolicyMgtAffiliate->GetAllIPConfigs( )
```

Parameters

The GetAllIPConfigs method accepts no parameters.

Return Value

The GetAllIPConfigs method returns one of the following values:

- An array of PolicyMgtIPConfg objects
- **undef** if no IP Configuration objects were found

GetAllUsers Method—Retrieves All Users Associated with an Affiliate

The GetAllUsers method retrieves all users associated with the affiliate object. If a user directory is specified, the method returns only those users associated with the affiliate and the particular directory.

Syntax

The GetAllUsers method has the following format:

```
Netegrity::PolicyMgtAffiliate->GetAllUsers( [userDir] )
```

Parameters

The GetAllUsers method accepts the following parameter:

userDir (PolicyMgtUserDir)

(Optional) Specifies a user directory that the affiliate users must be members of.

Return Value

The GetAllUsers method returns one of the following values:

- An array of PolicyMgtUser objects
- **undef** if no users were found, or if the call was unsuccessful

IsEnabled Method—Sets or Retrieves the Enabled Flag for the Affiliate

The IsEnabled method sets or retrieves the enabled flag for the affiliate object.

Syntax

The IsEnabled method has the following format:

```
Netegrity::PolicyMgtAffiliate->IsEnabled( [enableFlag] )
```

Parameters

The IsEnabled method accepts the following parameter:

enableFlag (int)

(Optional) Specifies whether to enable the affiliate object:

- A value of 1 enables the affiliate.
- A value of 0 disables the affiliate.

Return Value

The IsEnabled method returns one of the following values:

- 1 if the affiliate object is enabled
- 0 if the affiliate object is disabled
- -1 if the call was unsuccessful

Name Method—Sets or Retrieves the Affiliate Name

The Name method sets or retrieves the name of the affiliate object.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtAffiliate->Name( [affName] )
```

Parameters

The Name method accepts the following parameter:

affName (string)

(Optional) Specifies the name to set.

Return Value

The Name method returns one of the following values:

- The new or existing affiliate object name
- **undef** if the call was unsuccessful

Password Method—Sets or Retrieves a Password for an Affiliate

The Password method sets or retrieves the password that affiliates use to access SiteMinder Federation Web Services.

Syntax

The Password method has the following format:

```
Netegrity::PolicyMgtAffiliate->Password( [affPassword] )
```

Parameters

The Password method accepts the following parameter:

affPassword (string)

(Optional) Specifies the password to set.

Return Value

The Password method returns one of the following values:

- The new or existing password
- **undef** if the call was unsuccessful

RemoveAttribute Method—Removes an Attribute from an Affiliate

The RemoveAttribute method removes the specified affiliate attribute from the affiliate object.

Syntax

The RemoveAttribute method has the following format:

```
Netegrity::PolicyMgtAffiliate->RemoveAttribute( affiliateAttr )
```

Parameters

The RemoveAttribute method accepts the following parameter:

affiliateAttr (PolicyMgtAffiliateAttr)

Specifies the affiliate attribute to remove.

Return Value

The RemoveAttribute method returns one of the following values:

- 0 on success
- -1 on failure

RemoveUser Method—Removes a User from an Affiliate

The RemoveUser method removes the specified user from the affiliate object.

Syntax

The RemoveUser method has the following format:

```
Netegrity::PolicyMgtAffiliate->RemoveUser( user )
```

Parameters

The RemoveUser method accepts the following parameter:

user (type)

Specifies he user to remove.

Return Value

The RemoveUser method returns one of the following values:

- 0 on success
- -1 on failure

SAMLProfile Method—Sets or Retrieves the Type of SAML Profile

The SAMLProfile method sets or retrieves the type of profile used for sending and receiving SAML assertions.

Syntax

The SAMLProfile method has the following format:

```
Netegrity::PolicyMgtAffiliate->SAMLProfile([SAMLProfile])
```

Parameters

The SAMLProfile method accepts the following parameters:

SAMLProfile (long)

(Optional) Specifies one of the following valid SAML profile:

- AFFILIATE_SAML_PROFILE_ARTIFACT. The SAML assertion is retrieved from a URL associated with the assertion producer. The URL is specified during configuration of the SAML Artifact authentication scheme.
- AFFILIATE_SAML_PROFILE_POST. The generated SAML assertion is POSTed to the URL specified in the PolicyMgtAffiliate->ConsumerURL method. This profile is supported only if the Policy Management API's session version is at least v6.0 SP 2.

Return Value

The SAMLProfile method returns one of the following values:

- A new or existing SAML profile type
- **undef** if the call was unsuccessful

SAMLVersion Method—Sets or Retrieves the SAML Version for the Affiliate

The SAMLVersion method sets or retrieves the SAML version for the affiliate.

Syntax

The SAMLVersion method has the following format:

```
Netegrity::PolicyMgtAffiliate->SAMLVersion( [SAMLVer] )
```

Parameters

The SAMLVersion method accepts the following parameter:

SAMLVer (long)

(Optional) Specifies one of the following SAML versions to set:

- AFFILIATE_SAML_VER_1_0
- AFFILIATE_SAML_VER_1_1

Return Value

The SAMLVersion method returns one of the following values:

- A new or existing SAML version
- **undef** if the call was unsuccessful

Remarks

Specifying a SAML version has effect only if the Policy Manager API's session version is at least v6.0 SP 1.

Save Method—Saves the Affiliate to the Policy Store

The Save method saves the affiliate object to the policy store.

Syntax

The Save method has the following format:

```
Netegrity::PolicyMgtAffiliate->Save( )
```

Parameters

The Save method accepts no parameters.

Return Value

The Save method returns one of the following values:

- 0 on success.
- -1 on failure.
- -4 if the user has insufficient privileges to save the changes.
- -10 if the path and class are empty.

Remarks

Call this method once after making all the modifications to the affiliate object that you intend to make. This method must be called for any changes to take effect.

SessionSyncInterval Method—Sets or Retrieves the Session Synchronization Property

The SessionSyncInterval method sets or retrieves the session synchronization interval property. This property specifies the frequency, in seconds, at which the affiliate contacts the assertion producer site to validate the status of a shared session.

Syntax

The SessionSyncInterval method has the following format:

```
Netegrity::PolicyMgtAffiliate->SessionSyncInterval( [SessionSyncInterval] )
```

Parameters

The SessionSyncInterval method accepts the following parameter:

SessionSyncInterval (long)

(Optional) Specifies the session synchronization interval to set.

Return Value

The SessionSyncInterval method returns one of the following values:

- New or existing session synchronization interval
- **undef** if the call was unsuccessful

SharedSessioning Method—Sets or Retrieves the Shared Session Property

The SharedSessioning method sets or retrieves the shared session property. With shared sessions, the sessions on both the assertion producer site and the affiliate are terminated when the session on either site ends.

Syntax

The SharedSessioning method has the following format:

```
Netegrity::PolicyMgtAffiliate->ShareSessioning([shareFlag])
```

Parameters

The SharedSessioning method accepts the following parameter:

shareFlag (int)

(Optional) Specifies the shared session property to set:

- 1 to allow the assertion producer and the affiliate to share session information
- 0 to have the producer and affiliate maintain separate sessions

Return Value

The SharedSessioning method returns one of the following values:

- A new or existing shared session property value
- **undef** if the call was unsuccessful

SkewTime Method—Sets or Retrieves the Skew Time Property

The SkewTime method sets or retrieves the skew time property. The skew time is the difference, in seconds, between the system clock time of the assertion producer site and the system clock time of the affiliate site. Times are relative to GMT.

Syntax

The SkewTime method has the following format:

```
Netegrity::PolicyMgtAffiliate->SkewTime( [SkewTime] )
```

Parameters

The SkewTime method accepts the following parameter:

skewTime (long)

(Optional) Specifies the skew time to set.

Return Value

The SkewTime method returns one of the following values:

- A new or existing skew time
- **undef** if the call was unsuccessful

ValidityDuration Method—Sets or Retrieves the Duration a SAML Assertion Is Valid

The ValidityDuration method sets or retrieves the number of seconds that a SiteMinder-generated SAML assertion is valid. If an affiliate receives the assertion after the specified time, the assertion is considered invalid.

Syntax

The ValidityDuration method has the following format:

```
Netegrity::PolicyMgtAffiliate->ValidityDuration( [ValidityDuration] )
```

Parameters

The ValidityDuration method accepts the following parameter:

validityDuration (long)

(Optional) Specifies the validity duration time to set.

Return Value

The ValidityDuration method returns one of the following values:

- A new or existing validity duration time
- **undef** if the call was unsuccessful

Agent Methods

The following methods act on PolicyMgtAgent objects:

- ConvertFromLegacy Method—Converts a v4.x agent to a v5.x agent
- ConvertToLegacy Method—Converts a v5.x agent to a v4.x agent
- Description Method—Sets or retrieves the agent description
- IPAddress Method—Sets or retrieves the name of the agent
- Name Method—Sets or retrieves the name of the agent
- RealmHintAttrID—Sets or retrieves the hint attribute for a RADIUS agent
- SharedSecret Method—Sets or retrieves the shared secret for a 4.x agent

ConvertFromLegacy Method—Converts a v4.x Agent to a v5.x Agent

The ConvertFromLegacy method converts a v4.x agent to a v5.x agent.

Syntax

The ConvertFromLegacy method has the following format:

```
Netegrity::PolicyMgtAgent->ConvertFromLegacy( )
```

Parameters

The ConvertFromLegacy method accepts no parameters.

Return Value

The ConvertFromLegacy method returns one of the following values:

- 0 on success
- -1 on failure

ConvertToLegacy Method—Converts a v5.x Agent to a v4.x Agent

The ConvertToLegacy method converts a v5.x agent to a v4.x agent.

Syntax

The ConvertToLegacy method has the following format:

```
Netegrity::PolicyMgtAgent->ConvertToLegacy( )
```

Parameters

The ConvertToLegacy method accepts no parameters.

Return Value

The ConvertToLegacy method returns one of the following values:

- 0 on success
- -1 on failure

Description Method—Sets or Retrieves the Agent Description

The Description method sets or retrieves the agent description.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtAgent->Description([agentDesc])
```

Parameters

The Description method accepts the following parameter:

agentDesc (string)

(Optional) Specifies the description to set.

Return Value

The Description method returns one of the following values:

- New or existing description of the agent
- An empty string if unsuccessful

IPAddress Method—Sets or Retrieves the Agent's IP Address

The IPAddress method sets or retrieves the agent's IP address.

Syntax

The IPAddress method has the following format:

```
Netegrity::PolicyMgtAgent->IPAddress([ipAddress])
```

Parameters

The IPAddress method accepts the following parameter:

ipAddress (string)

(Optional) Specifies the IP address to set.

Return Value

The IPAddress method returns one of the following values:

- A new or existing agent IP address
- **undef** if the call was unsuccessful

Name Method—Sets or Retrieves the Name of the Agent

The Name method sets or retrieves the name of the agent.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtAgent->Name([agentName])
```

Parameters

The Name method accepts the following parameter:

agentName (string)

(Optional) Specifies the name to assign to the agent.

Return Value

The Name method returns one of the following values:

- The new or existing name of the agent
- undef if the call was unsuccessful

RealmHintAttrID Method—Sets or Retrieves the Hint Attribute

The RealmHintAttrID method sets or retrieves the hint attribute for a RADIUS agent.

Syntax

The RealmHintAttrID method has the following format:

```
Netegrity::PolicyMgtAgent->RealmHintAttrID([hintID])
```

Parameters

The RealmHintAttrID method accepts the following parameter:

hintID (int)

(Optional) Specifies the hint attribute ID to set.

Return Value

The RealmHintAttrID method returns one of the following values:

- New or existing realm hint attribute to set for the RADIUS agent
- -1 if unsuccessful

SharedSecret Method—Sets or Retrieves the Shared Secret for a v4.x Agent

The SharedSecret method sets or retrieves the shared secret for a v4.x agent. This is the same shared secret used in the Web agent configuration.

Syntax

The SharedSecret method has the following format:

```
Netegrity::PolicyMgtAgent->SharedSecret([sharedSecret])
```

Parameters

The SharedSecret method accepts the following parameter:

sharedSecret (string)

(Optional) Specifies the shared secret to set.

Return Value

The SharedSecret method returns one of the following values:

- The new or existing shared secret
- undef if the call was unsuccessful

Agent Configuration Methods

The following methods act on PolicyMgtAgentConfig objects:

- AddAssociation Method—Adds a parameter name-value pair object for the agent configuration
- AddAssociationMultiValue Method—Adds a multi-valued agent configuration parameter
- Description Method—Sets or retrieves the description of the agent configuration object
- GetAssociations Method—Retrieves an array of all of the parameter name-value pair objects for this agent configuration
- Name Method—Sets or retrieves the name of the agent configuration object
- RemoveAssociation Method—Removes the specified name-value pair object

AddAssociation Method—Adds a Name and Value for this Configuration

The AddAssociation method adds a configuration parameter name and value for this agent configuration.

Syntax

The AddAssociation method has the following format:

```
Netegrity::PolicyMgtAgentConfig->AddAssociation(Name, Value, Flags)
```

Parameters

The AddAssociation method accepts the following parameters:

Name (string)

Specifies the configuration parameter name.

Value (string)

Specifies the configuration parameter value.

Flag (int)

Specifies the encryption flag value:

- 1 if the name/value pair is stored in encrypted format
- 0 if the name/value pair is stored as plain text

Return Value

The AddAssociation method returns one of the following values:

- A PolicyMgtAssociation Object
- **undef** if the call was unsuccessful

AddAssociationMultiValue Method—Adds a Multi-valued Configuration Parameter

The AddAssociationMultiValue method adds a multi-valued configuration parameter for this agent configuration. If the specified configuration parameter exists, the value is updated.

Syntax

The AddAssociationMultiValue method has the following format:

```
Netegrity::PolicyMgtAgentConfig->AddAssociationMultiValue(Name, valueArray)
```

Parameters

The AddAssociationMultiValue method accepts the following parameters:

Name (string)

Specifies the configuration parameter name.

valueArray (string array)

Specifies a reference to an array of values associated with this parameter name (for example: \@myarray).

Return Value

The AddAssociationMultiValue method returns one of the following values:

- A PolicyMgtAssociation object
- **undef** if the call was unsuccessful

Remarks

Multi-valued parameters can be stored only as plain text.

Description Method—Sets or Retrieves the Description of the Agent Configuration Object

The Description method sets or retrieves the description of the agent configuration object.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtAgentConfig->Description([Description])
```

Parameters

The Description method accepts the following parameters:

Description (string)

(Optional) Specifies the description to set.

Return Value

The Description method returns one of the following values:

- The new or existing description of the agent configuration object
- **undef** if the call was unsuccessful

GetAssociations Method—Retrieves a List of All the Configuration Parameters

The GetAssociations method retrieves a list of all the configuration parameters for this agent configuration.

Syntax

The GetAssociations method has the following format:

```
Netegrity::PolicyMgtAgentConfig->GetAssociations( )
```

Parameters

The GetAssociations method accepts no parameters.

Return Value

The GetAssociations method returns one of the following values:

- An array of PolicyMgtAssociation objects. Each object includes a configuration parameter name and its associated value.
- **undef** if no configuration parameter objects exist, or if the call is unsuccessful

Name Method—Sets or Retrieves the Agent Configuration Object Name

The Name method sets or retrieves the agent configuration object name.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtAgentConfig->Name([Name])
```

Parameters

The Name method accepts the following parameters:

Name (string)

(Optional) Specifies the name to set.

Return Value

The Name method returns one of the following values:

- The new or existing agent configuration object name
- **undef** if the call was unsuccessful

RemoveAssociation Method—Removes a Configuration Parameter

The RemoveAssociation method removes a configuration parameter name/value pair from the agent configuration object.

Syntax

The RemoveAssociation method has the following format:

```
Netegrity::PolicyMgtAgentConfig->RemoveAssociation(assoc)
```

Parameters

The RemoveAssociation method accepts the following parameter:

assoc (PolicyMgtAssociation)

Specifies the configuration parameter name/value pair to remove.

Return Value

The RemoveAssociation method returns one of the following values:

- 0 on success
- -1 if the call was unsuccessful

Agent Configuration Parameters Methods

An object of this type represents a configuration parameter name-value pair for an agent configuration. The following methods act on PolicyMgtAssociation objects:

- Name Method—Sets or retrieves the name portion of the name-value pair
- Flags Method—Sets or retrieves the flags attribute for the name-value pair
- Value Method—Sets or retrieves the value portion of the name-value pair

Name Method—Sets or Retrieves the Name Portion of the Agent Configuration Parameter

The Name method sets or retrieves the name portion of the agent configuration parameter name/value pair.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtAssociation->Name([Name])
```

Parameters

The Name method accepts the following parameters:

Name (string)

(Optional) Specifies the name to set.

Return Value

The Name method returns one of the following values:

- The name of the agent configuration parameter.
- **undef** if unsuccessful

Flags Method—Sets or Retrieves the Encryption Flag Attribute

The Flags method sets or retrieves the encryption flag attribute for the agent configuration name/value pair.

Syntax

The Flags method has the following format:

```
Netegrity::PolicyMgtAssociation->Flags([Flags])
```

Parameters

The Flags method accepts the following parameter:

Flags (int)

(Optional) Specifies the flag value to set.

Return Value

The Flags method returns one of the following values:

- 1 if the name/value pair is in encrypted format
- 0 if the name/value pair is plain text
- **undef** if the call was unsuccessful

Value Method—Sets or Retrieves the Value of the Agent Configuration Parameter

The Value method sets or retrieves the value portion of the agent configuration parameter name/value pair.

Syntax

The Value method has the following format:

```
Netegrity::PolicyMgtAssociation->Value([Value])
```

Parameters

The Value method accepts the following parameter:

Value (int)

(Optional) Specifies the value to set.

Return Value

The Value method returns one of the following values:

- The value of the agent configuration parameter
- **undef** if unsuccessful

Agent Type Methods

The following methods act on PolicyMgtAgentType objects:

- GetDescription Method—Retrieves the description of the agent type
- GetName Method—Retrieves the name of the agent type

GetDescription Method—Retrieves the Description of the Agent Type

The GetDescription method retrieves the description of the agent type.

Syntax

The GetDescription method has the following format:

```
Netegrity::PolicyMgtAgentType->GetDescription( )
```

Parameters

The GetDescription method accepts no parameters.

Return Value

The GetDescription method returns one of the following values:

- The new or existing description of the agent type
- A null string if unsuccessful

GetName Method—Retrieves the Name of the Agent Type

The GetName method retrieves the name of the agent type, for example, Web Agent.

Syntax

The GetName method has the following format:

```
Netegrity::PolicyMgtAgentType->GetName( )
```

Parameters

The GetName method accepts no parameters.

Return Value

The GetName method returns one of the following values:

- The name of the agent type
- A null string if unsuccessful

Authentication and Authorization Map Methods

The following methods act on PolicyMgtAuthAzMap objects:

- AuthDir Method—Sets or retrieves the authentication directory of the authentication and authorization map
- AzDir Method—Sets or retrieves the authorization directory of the authentication and authorization map
- MapType Method—Sets or retrieves the type of authentication and authorization map

AuthDir Method—Sets or Retrieves the Authentication Directory

The AuthDir method sets or retrieves the authentication directory of the authentication and authorization map.

Syntax

The AuthDir method has the following format:

```
Netegrity::PolicyMgtAuthAzMap->AuthDir([userDir])
```

Parameters

The AuthDir method accepts the following parameter:

userDir (PolicyMgtUserDir)

(Optional) Specifies the authentication directory to set.

Return Value

The AuthDir method returns one of the following values:

- A new or existing PolicyMgtUserDir object.
- **undef** if the call was unsuccessful

AzDir Method—Sets or Retrieves the Authorization Directory

The AzDir method sets or retrieves the authorization directory of the authentication and authorization map.

Syntax

The AzDir method has the following format:

```
Netegrity::PolicyMgtAuthAzMap->AzDir([userDir])
```

Parameters

The AzDir method accepts the following parameter:

userDir (PolicyMgtUserDir)

(Optional) Specifies the authorization directory to set.

Return Value

The AzDir method returns one of the following values:

- A new or existing PolicyMgtUserDir object
- **undef** if the call was unsuccessful

MapType Method—Sets or Retrieves the Type of Authentication and Authorization Map

The MapType method sets or retrieves the type of authentication and authorization map.

Syntax

The MapType method has the following format:

```
Netegrity::PolicyMgtAuthAzMap->MapType([mapType])
```

Parameters

The MapType method accepts the following parameter:

mapType (int)

(Optional) Specifies the map type. The following values are valid:

AUTHAZMAPTYPE_DN (Value=1). Mapping is based on a DN.

AUTHAZMAPTYPE_UNIVERSALID (Value=2). Mapping is based on a universal identifier.

AUTHAZMAPTYPE_ATTR (Value=3). Mapping is based on an attribute in the directory.

Return Value

The MapType method returns one of the following values:

- A new or existing map type
- -1 if the call was unsuccessful

Authentication Scheme Methods

The following methods act on PolicyMgtAuthScheme objects.:

- AddMessageConsumerPluginToSAML1xScheme Method—Adds MessageConsumerPlugin class name and parameter to SAML1x authentication scheme.
- AddRedirectURLToSAML1xScheme Method—Adds the redirection location to a SAML1x authentication scheme.
- AddTargetConfigToSAML1xScheme Method—Adds default Target and QueryParameterOverridesTarget configuration to SAML1x authentication scheme.
- CustomLib Method—Sets or retrieves the name of the shared library that implements the custom authentication scheme.
- CustomParam Method—Sets or retrieves information that is passed to the custom authentication scheme.
- CustomSecret Method—Sets or retrieves the shared secret for the custom authentication scheme.
- Description Method—Sets or retrieves the description of the authentication scheme.
- GetMessageConsumerPluginFromSAML1xScheme Method—Retrieves the MessageConsumerPlugin class name and parameter from a SAML1x authentication scheme.
- GetTargetConfigFromSAML1xScheme Method—Retrieves the default Target and QueryParameterOverridesTarget configuration from a SAML1x authentication scheme.
- GetRedirectURLFromSAML1xScheme Method—Retrieves the redirect URL from a SAML1x authentication scheme.
- IgnorePwd Method—Sets or retrieves the flag that specifies whether a password should be checked.
- IsRadius Method—Sets or retrieves the flag that specifies whether the authentication scheme supports RADIUS agents.
- IsTemplate Method—Retrieves the flag value that indicates whether the authentication scheme can be used as a template.
- IsUsedByAdmin Method—Determines whether the scheme should be used for SiteMinder administrators.
- Name Method—Sets or retrieves the name of the authentication scheme.
- ProtectionLevel Method—Sets or retrieves the protection level of the authentication scheme.
- Save Method—Saves modifications to an authentication scheme.

- SaveCredentials Method—Sets or retrieves the flag that allows user credentials to be saved.
- Type Method—Sets or retrieves the authentication scheme type.

AddMessageConsumerPluginToSAML1xScheme Method--Adds Message Consumer Plug-in Class Name

The AddMessageConsumerPluginToSAML1xScheme method adds a message consumer plug-in class name and parameter to a SAML1x authentication scheme.

Syntax

The AddMessageConsumerPluginToSAML1xScheme method has the following format:

```
Netegrity::PolicyMgtAuthScheme->AddMessageConsumerPluginToSAML1xScheme(pluginClas
s, pluginParam)
```

Parameters

The AddMessageConsumerPluginToSAML1xScheme method accepts the following parameters:

pluginClass (string)

Specifies the message consumer plug-in class name.

pluginParam (string)

Specifies the message consumer plug-in parameter name.

Return Value

The AddMessageConsumerPluginToSAML1xScheme method returns one of the following values:

- Sm_PolicyApi_Success
- Sm_PolicyApi_Failure

AddRedirectURLToSAML1xScheme Method--Adds Redirect Value to an Authentication Scheme

The AddRedirectURLToSAML1xScheme method adds a redirect URL, type, and mode to a SAML1x authentication scheme.

Syntax

The AddRedirectURLToSAML1xScheme method has the following format:

```
Netegrity::PolicyMgtAuthScheme->AddRedirectURLToSAML1xScheme(iTypeURL, URL,
redirectMode)
```

Parameters

The AddRedirectURLToSAML1xScheme method accepts the following parameters:

iTypeURL (int)

Specifies the redirect URL type, which is one of the following values:

0—User Note Found

1— Invalid Message

2—Unaccepted credential

URL (string)

Specifies the redirect URL site.

redirectMode (int)

Specifies the redirect mode, which can be either of the following values:

- 0—302 no Data
- 1—POST

Return Value

The AddRedirectURLToSAML1xScheme method returns one of the following values:

- Sm_PolicyApi_Success
- Sm_PolicyApi_Failure

AddTargetConfigToSAML1xScheme Method--Sets the Default Target Configuration

The AddTargetConfigToSAML1xScheme method sets the default Target and QueryParameterOverridesTarget configuration to a SAML1x authentication scheme.

Syntax

The AddTargetConfigToSAML1xScheme method has the following format:

```
Netegrity::PolicyMgtAuthScheme->AddTargetConfigToSAML1xScheme(pszTargetURL,
iQPOverrideTarget)
```

Parameters

The AddTargetConfigToSAML1xScheme method accepts the following parameters:

pszTargetURL (string)

Specifies the default Target URL.

iQPOverrideTarget (int)

Specifies whether the query parameter overrides the default Target configuration.

Return Value

The AddTargetConfigToSAML1xScheme method returns one of the following values:

- Sm_PolicyApi_Success
- Sm_PolicyApi_Failure

CustomLib Method—Sets or Retrieves the Name of the Shared Library

The CustomLib method sets or retrieves the name of the shared library that implements the authentication scheme.

Syntax

The CustomLib method has the following format:

```
Netegrity::PolicyMgtAuthScheme->CustomLib([libName])
```

Parameters

The CustomLib method accepts the following parameter:

libName (string)

(Optional) Specifies the shared library name.

Return Value

The CustomLib method returns one of the following values:

- The new or existing library name
- **undef** if the call was unsuccessful

Remarks

Each pre-defined authentication scheme type is shipped with a default library, but you can use a custom library. If you use a custom authentication scheme, you must specify a custom library.

CustomParam Method—Sets or Retrieves Information that Is Passed to the Authentication Scheme

The CustomParam method sets or retrieves information that is passed to the authentication scheme. You can accept the default parameter for the authentication scheme, or you can specify a new one.

Syntax

The CustomParam method has the following format:

```
Netegrity::PolicyMgtAuthScheme->CustomParam([param])
```

Parameters

The CustomParam method accepts the following parameter:

param (string)

(Optional) Specifies the parameter information to pass.

Return Value

The CustomParam method returns one of the following values:

- The new or existing parameter information
- A null string if the call was unsuccessful

CustomSecret Method—Sets or Retrieves the Shared Secret for the Custom Authentication Scheme

The CustomSecret method sets or retrieves the shared secret for the custom authentication scheme.

Syntax

The CustomSecret method has the following format:

```
Netegrity::PolicyMgtAuthScheme->CustomSecret([param])
```

Parameters

The CustomSecret method accepts the following parameter:

param (string)

(Optional) Specifies the shared secret.

Return Value

The CustomSecret method returns one of the following values:

- The new or existing shared secret
- A null string if the call was unsuccessful

Description Method—Sets or Retrieves the Description of the Authentication Scheme

The Description method sets or retrieves the description of the authentication scheme.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtAuthScheme->Description([schemeDesc])
```

Parameters

The Description method accepts the following parameter:

schemeDesc (string)

(Optional) Specifies the description.

Return Value

The Description method returns one of the following values:

- The new or existing authentication scheme description
- An empty string if the call was unsuccessful

GetMessageConsumerPluginFromSAML1xScheme Method--Retrieves Message Consumer Plug-in Class Name

The GetMessageConsumerPluginFromSAML1xScheme method retrieves the message consumer plug-in class name and parameter from a SAML1x authentication scheme.

Syntax

The GetMessageConsumerPluginFromSAML1xScheme method has the following format:

```
Netegrity::PolicyMgtAuthScheme->GetMessageConsumerPluginFromSAML1xScheme(pluginCl
ass, pluginParam)
```

Parameters

The GetMessageConsumerPluginFromSAML1xScheme method accepts the following parameters:

pluginClass (string)

Specifies the message consumer plug-in class name.

pluginParam (string)

Specifies the message consumer plug-in parameter name.

Return Value

The GetMessageConsumerPluginFromSAML1xScheme method returns one of the following values:

- Sm_PolicyApi_Success
- Sm_PolicyApi_Failure

GetRedirectURLFromSAML1xScheme Method--Retrieves a Redirect URL

The GetRedirectURLFromSAML1xScheme method retrieves a redirect URL, type, and mode from a SAML1x authentication scheme.

Syntax

The GetRedirectURLFromSAML1xScheme method has the following format:

```
Netegrity::PolicyMgtAuthScheme->GetRedirectURLFromSAML1xScheme(iTypeURL, URL,
redirectMode)
```

Parameters

The GetRedirectURLFromSAML1xScheme method accepts the following parameters:

iTypeURL (int)

Specifies the redirect URL type, which is one of the following values:

0—User Note Found

1— Invalid Message

2—Unaccepted credential

URL (string)

Specifies the redirect URL site.

redirectMode (int)

Specifies the redirect mode, which can be either of the following values:

- 0—302 No Data
- 1—POST

Return Value

The GetRedirectURLFromSAML1xScheme method returns one of the following values:

- Sm_PolicyApi_Success
- Sm_PolicyApi_Failure

GetTargetConfigFromSAML1xScheme Method--Retrieves the Target Configuration

The GetTargetConfigFromSAML1xScheme method retrieves the default Target and QueryParameterOverridesTarget configuration from a SAML1x authentication scheme.

Syntax

The GetTargetConfigFromSAML1xScheme method has the following format:

```
Netegrity::PolicyMgtAuthScheme->GetTargetConfigFromSAML1xScheme(pszTargetURL,
iQPOverrideTarget)
```

Parameters

The GetTargetConfigFromSAML1xScheme method accepts the following parameters:

pszTargetURL (string)

Specifies the default Target URL.

iQPOverrideTarget (int)

Specifies whether the query parameter overrides the default Target configuration.

Return Value

The GetTargetConfigFromSAML1xScheme method returns one of the following values:

- Sm_PolicyApi_Success
- Sm_PolicyApi_Failure

IgnorePwd Method—Specifies whether Password Policies Should Be Checked

The IgnorePwd method sets or retrieves the flag that specifies whether password policies should be checked for the authentication scheme.

Syntax

The IgnorePwd method has the following format:

```
Netegrity::PolicyMgtAuthScheme->IgnorePwd([pwdFlag])
```

Parameters

The IgnorePwd method accepts the following parameter:

pwdFlag (int)

> (Optional) Specifies whether to ignore password policies (set to 1), or enforce them (set to 0).

Return Value

The IgnorePwd method returns one of the following values:

- 1 if password policies should be ignored
- 0 if password policies
- -1 if the call was unsuccessful

IsRadius Method—Determines whether the Authentication Scheme Supports RADIUS Agents

The IsRadius method sets or retrieves the flag that specifies whether the authentication scheme supports RADIUS agents.

Syntax

The IsRadius method has the following format:

```
Netegrity::PolicyMgtAuthScheme->IsRadius([radFlag])
```

Parameters

The IsRadius method accepts the following parameter:

radFlag (int)

> (Optional) Specifies whether the authentication scheme supports RADIUS agents (1=yes; 0=no).

Return Value

The IsRadius method returns one of the following values:

- 1 if the authentication scheme supports RADIUS agents
- 0 if the authentication scheme does not support RADIUS agents
- -1 if the call was unsuccessful

IsTemplate Method—Determines whether the Authentication Scheme Is a Template

The IsTemplate method retrieves the flag value that indicates whether the authentication scheme is a template.

Syntax

The IsTemplate method has the following format:

```
Netegrity::PolicyMgtAuthScheme->IsTemplate( )
```

Parameters

The IsTemplate method accepts no parameters.

Return Value

The IsTemplate method returns one of the following values:

- 1 if the authentication scheme is a template
- 0 if the authentication scheme is not a template
- -1 the call was unsuccessful

Remarks

Setting an authentication scheme as a template with the Perl Policy Management API is deprecated in SiteMinder v6.0 SP3.

IsUsedByAdmin Method—Determines whether the Scheme Authenticates Administrators

The IsUsedByAdmin method determines whether the scheme should be used to authenticate administrators.

Syntax

The IsUsedByAdmin method has the following format:

```
Netegrity::PolicyMgtAuthScheme->IsUsedByAdmin([useAdminFlag])
```

Parameters

The IsUsedByAdmin method accepts the following parameter:

useAdminFlag (int)

(Optional) Specifies whether the scheme should be used to authenticate administrators:

- 1 to allow the scheme to be used for administrator authentication
- 0 to disallow the scheme to be used for administrator authentication

Return Value

The IsUsedByAdmin method returns one of the following values:

- 1 if the scheme can be used to authenticate administrators
- 0 if the scheme cannot be used to authenticate administrators
- -1 if the call was unsuccessful

Name Method—Sets or Retrieves the Name of the Authentication Scheme

The Name method sets or retrieves the name of the authentication scheme.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtAuthScheme->Name([authSchemeName])
```

Parameters

The Name method accepts the following parameter:

authSchemeName (string)

(Optional) Specifies the name to assign to the authentication scheme.

Return Value

The Name method returns one of the following values:

- The new or existing authentication scheme name
- undef if the call was unsuccessful

ProtectionLevel Method—Sets or Retrieves the Protection Level of the Authentication Scheme

The ProtectionLevel method sets or retrieves the protection level of the authentication scheme.

Syntax

The ProtectionLevel method has the following format:

```
codefirstNetegrity::PolicyMgtAuthScheme->ProtectionLevel([nlevel])
```

Parameters

The ProtectionLevel method accepts the following parameter:

nlevel (int)

(Optional) Specifies the protection level to set.

Return Value

The ProtectionLevel method returns one of the following values:

- The new or existing authorization scheme protection level
- -1 if unsuccessful

Remarks

The level can vary from 1 to 1000. The higher the number, the more secure is the scheme. With Anonymous authentication schemes, set this value to 0.

Save Method—Saves the Authentication Scheme to the Policy Store

The Save method saves the authentication scheme to the policy store.

Syntax

The Save method has the following format:

```
Netegrity::PolicyMgtAuthScheme->Save( )
```

Parameters

The Save method accepts no parameters.

Return Value

The Save method returns one of the following values:

- 0 on success
- -1 on failure
- -4 if the user has insufficient privileges to save the changes
- -100 if the scheme object identifier is not found

Remarks

Call this method once after making all the modifications to the authentication scheme that you intend to make. This method must be called for any changes to take effect.

SaveCredentials Method—Determines whether User Credentials Can Be Saved

The SaveCredentials method sets or retrieves the flag that allows user credentials to be saved.

Syntax

The SaveCredentials method has the following format:

```
Netegrity::PolicyMgtAuthScheme->SaveCredentials([credFlag])
```

Parameters

The SaveCredentials method accepts the following parameter:

credFlag (int)

(Optional) Specifies the flag value:

- 1 if credentials can be saved
- 0 if credentials cannot be saved

Return Value

The SaveCredentials method returns one of the following values:

- 1 if user credentials can be saved
- 0 if user credentials cannot be saved

Type Method—Sets or Retrieves the Authentication Scheme Type

The Type method sets or retrieves the authentication scheme type.

Syntax

The Type method has the following format:

```
Netegrity::PolicyMgtAuthScheme->Type([template])
```

Parameters

The Type method accepts the following parameter:

template (PolicyMgtAuthScheme)

(Optional) Specifies the authentication scheme type.

Return Value

The Type method returns one of the following values:

- The new or existing authentication scheme type
- **undef** if the call was unsuccessful

Certificate Mapping Methods

The following methods act on PolicyMgtCertMap objects:

- AttrMap Method—Sets or retrieves the AttributeMap
- CacheCRL Method—Sets or retrieves the flag that determines whether to cache Certificate Revocation List (CRL) entries
- CertRequired Method—Sets or retrieves the flag that requires SiteMinder to verify that the certificate presented by the user matches the certificate stored in the user's entry in the user directory
- CRLUserDirectory Method—Specifies or retrieves the LDAP directory where the CRL is located
- Description Method—Sets or retrieves the description of the certificate map
- DirectoryType Method—Sets or retrieves the type of user directory (LDAP, ODBC database, or WinNT) involved in the user authentication
- EnableCRL Method—Sets or retrieves the flag that determines whether to check the CRL for revoked certificates
- IssuerDN Method—Sets or retrieves the DN of the certificate issuer

- UseDistributionPoints Method—Sets or retrieves the flag indicating whether CRL searches should use distribution points
- VerifySignature Method—Sets or retrieves the flag indicating whether SiteMinder should verify the Certificate Authority's signature in the CRL

AttrMap Method—Sets or Retrieves the Attribute Map for Certificate Mapping

The AttrMap method sets or retrieves the attribute map for Certificate mapping.

Syntax

The AttrMap method has the following format:

```
Netegrity::PolicyMgtCertMap->AttrMap ([attribute_map])
```

Parameters

The AttrMap method accepts the following parameter:

attribute_map (string)

(Optional) Specifies the attribute map to be set.

Return Value

The AttrMap method returns one of the following values:

- A new or existing attribute of the Certificate map
- An empty string if the call was unsuccessful

CacheCRL Method—Determines whether To Cache Certificate Revocation List (CRL) entries

The CacheCRL method sets or retrieves the flag that determines whether to cache Certificate Revocation List (CRL) entries. Setting this flag causes SiteMinder to use cached CRL information until the date specified in the NextUpdate field in the CRL.

Syntax

The CacheCRL method has the following format:

```
Netegrity::PolicyMgtCertMap->CacheCRL([cacheFlag])
```

Parameters

The CacheCRL method accepts the following parameter:

cacheFlag (int)

(Optional) Specifies whether to cache CRL entries:

- 1 specifies that cache entries are used
- 0 specifies that cache entries are not used

Return Value

The CacheCRL method returns one of the following values:

- The new or existing cache flag setting
- -1 if the call was unsuccessful

CertRequired Method—Determines whether Certificate Validation is Required

The CertRequired method sets or retrieves the flag that requires SiteMinder to verify that the certificate presented by the user matches the certificate stored in the user's entry in the user directory. The user directory must be an LDAP user directory.

Syntax

The CertRequired method has the following format:

```
Netegrity::PolicyMgtCertMap->CertRequired([certFlag])
```

Parameters

The CertRequired method accepts the following parameter:

certFlag (int)

(Optional) Specifies whether certificate verification is required:

- 1 certificate verification is required
- 0 certificate verification is not required

Return Value

The CertRequired method returns one of the following values:

- The new or existing flag setting
- -1 if the call was unsuccessful

CRLUserDirectory Method—Sets or Retrieves the LDAP Directory where the Certificate Revocation List (CRL) Is Located

The CRLUserDirectory method specifies or retrieves the LDAP user directory where the Certificate Revocation List (CRL) is located.

Syntax

The CRLUserDirectory method has the following format:

```
Netegrity::PolicyMgtCertMap->CRLUserDirectory([crlDir])
```

Parameters

The CRLUserDirectory method accepts the following parameter:

crlDir (PolicyMgtUserDir)

(Optional) Specifies the user directory where the CRL is located.

Return Value

The CRLUserDirectory method returns one of the following values:

- A PolicyMgtUserDir object
- **undef** if the call was unsuccessful

Description Method—Sets or Retrieves the Description of the Certificate Map

The Description method sets or retrieves the description of the certificate map.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtCertMap->Description([certMapDesc])
```

Parameters

The Description method accepts the following parameter:

certMapDesc (string)

(Optional) Specifies the description to set.

Return Value

The Description method returns one of the following values:

- A new or existing certificate map description
- An empty string if the call was unsuccessful

DirectoryType Method—Sets or Retrieves the Type of User Directory

The DirectoryType method sets or retrieves the type of user directory involved in the user authentication.

Syntax

The DirectoryType method has the following format:

```
Netegrity::PolicyMgtCertMap->DirectoryType([dirType])
```

Parameters

The DirectoryType method accepts the following parameter:

dirType (int)

(Optional) Specifies one of the following types of user directory:

- Sm_PolicyApi_DirType_LDAP
- Sm_PolicyApi_DirType_WinNT
- Sm_PolicyApi_DirType_ODBC

Return Value

The DirectoryType method returns one of the following values:

- The new or existing directory type
- **undef** if the call was unsuccessful

EnableCRL Method—Determines whether To Check the Certificate Revocation List (CRL) for Revoked Certificates

The EnableCRL method sets or retrieves the flag that determines whether to check the Certificate Revocation List (CRL) for revoked certificates.

Syntax

The EnableCRL method has the following format:

```
Netegrity::PolicyMgtCertMap->EnableCRL([ckCRLFlag])
```

Parameters

The EnableCRL method accepts the following parameter:

ckCRLFlag (int)

(Optional) Specifies whether to check certificates against the CRL:

- 1 specifies that certificates should be checked
- 0 specifies that certificates should not be checked

Return Value

The EnableCRL method returns one of the following values:

- The new or existing flag setting
- -1 if the call was unsuccessful

Remarks

A CRL is a list of revoked X.509 client certificates published by the Certificate Authority. Comparing a certificate against a CRL is one way to ensure that certificates are valid. When a user with such a certificate tries to access a protected resource, SiteMinder finds the user's certificate in the CRL and rejects the authentication.

Before you enable CRL checking, call the method PolicyMgtCertMap->CRLUserDirectory to specify the user directory where the CRL is located.

IssuerDN Method—Sets or Retrieves the DN of the Certificate Issuer

The IssuerDN method sets or retrieves the DN of the certificate issuer.

Syntax

The IssuerDN method has the following format:

```
Netegrity::PolicyMgtCertMap->IssuerDN([issuerDN])
```

Parameters

The IssuerDN method accepts the following parameter:

issuerDN (string)

(Optional) Specifies the issuer DN to set.

Return Value

The IssuerDN method returns one of the following values:

- The new or existing issuer DN
- An empty string if the call is unsuccessful

UseDistributionPoints Method—Determines whether Certificate Revocation List (CRL) Searches Use a Distribution Point

The UseDistributionPoints method sets or retrieves the flag indicating whether Certificate Revocation List (CRL) searches should use a distribution point as a starting point for a search.

Syntax

The UseDistributionPoints method has the following format:

```
Netegrity::PolicyMgtCertMap->UseDistributionPoints([distPointsFlag])
```

Parameters

The UseDistributionPoints method accepts the following parameters:

distPointsFlag (int)

(Optional) Specifies whether to use distribution points for CRL searches:

- 1 specifies that distribution points should be used
- 0 specifies that the whole CRL should be searched

Return Value

The UseDistributionPoints method returns one of the following values:

- The new or existing flag setting
- -1 if the call was unsuccessful

Remarks

Large CRLs may contain multiple distribution points that can be used to locate a revoked user. Distribution points indicate a starting point in the CRL LDAP directory. By providing a starting point for a CRL check, distribution points save the processing time that it would take to search the entire CRL.

VerifySignature Method—Determines whether SiteMinder Verifies the Certificate Authority's Signature

The VerifySignature method sets or retrieves the flag indicating whether SiteMinder should verify the Certificate Authority's signature in the Certificate Revocation List (CRL).

Syntax

The VerifySignature method has the following format:

```
Netegrity::PolicyMgtCertMap->VerifySignature([verifyFlag])
```

Parameters

The VerifySignature method accepts the following parameter:

verifyFlag (int)

(Optional) Specifies whether to verify the CA's signature in the CRL:

- 1 specifies that the signature should be verified
- 0 specifies that the signature should not be verified

Return Value

The VerifySignature method returns one of the following values:

- The new or existing flag setting
- -1 if the call was unsuccessful

Cluster Methods

The following methods act on PolicyMgtCluster objects:

- AddServer Method—Adds a server to the cluster
- GetAllServers Method—Retrieves an array of all servers in the cluster

AddServer Method—Adds a Server to the Cluster

The AddServer method adds a server to the cluster.

Syntax

The AddServer method has the following format:

```
Netegrity::PolicyMgtCluster->AddServer(Host, Port)
```

Parameters

The AddServer method accepts the following parameters:

Host (string)

Specifies the host IP address.

Port (int)

Specifies the server port.

Return Value

The AddServer method returns one of the following values:

- A PolicyMgtServer object
- **undef** if the call was unsuccessful

Remarks

The servers in a cluster are referenced in an array. When you add a server to a cluster, it is added to the end of the server array.

Due to dynamic load balancing, in which requests are sent to the highest-capacity available server in the cluster, the order in which servers are added to the cluster does not matter.

To add a non-clustered server to a host configuration, call the PolicyMgtHostConfig->AddServer method.

GetAllServers Method—Retrieves an Array of All the Servers in a Cluster

The GetAllServers method retrieves an array of all the servers in the cluster.

Syntax

The GetAllServers method has the following format:

```
Netegrity::PolicyMgtCluster->GetAllServers( )
```

Parameters

The GetAllServers method accepts no parameters.

Return Value

The GetAllServers method returns one of the following values:

- An array of PolicyMgtServer objects
- **undef** if the call was unsuccessful

Remarks

To retrieve the servers that are not members of clusters, call the PolicyMgtHostConfig->GetAllServers method.

Domain Methods

The following methods act on PolicyMgtDomain objects:

- AddAdmin Method—Adds an administrator to the domain
- AddUserDir Method—Associates a user directory with the domain
- CreatePolicy Method—Creates a policy in the domain
- CreateRealm Method—Creates a realm in the domain
- CreateResponse Method—Creates a response in the domain
- CreateResponseGroup Method—Creates a response group for the domain
- CreateRuleGroup Method—Creates a rule group for the domain
- DeleteGroup Method—Deletes a group
- DeletePolicy Method—Deletes a policy
- DeleteVariable Method—Deletes a variable
- Description Method—Sets or retrieves the description of the domain

- GetAllPolicies Method—Retrieves an array of policy objects in the domain
- GetAllRealms Method—Retrieves an array of all top-level realms in the domain
- GetAllResponseGroups Method—Retrieves an array of all the response groups for the domain
- GetAllResponses Method—Retrieves an array of all responses associated with the domain
- GetAllRuleGroups Method—Retrieves an array of all the rule groups for the domain
- GetAllVariables Method—Retrieves all variable objects in the domain
- GetPolicy Method—Retrieves a policy in the domain
- GetRealm Method—Retrieves a top-level realm in the domain
- GetResponse Method—Retrieves a response associated with the domain
- GetResponseGroup Method—Retrieves the specified response group
- GetRuleGroup Method—Retrieves the specified rule group
- GetUserDirSearchOrder Method—Retrieves user directory objects associated with the domain
- GetVariable Method—Retrieves the specified variable object
- GlobalPoliciesApply Method—Sets or retrieves the flag that specifies whether global policies are enabled for the domain
- Name Method—Sets or retrieves the domain name
- RemoveAdmin Method—Disassociates the administrator from the domain
- RemoveUserDir Method—Disassociates the user directory from the domain
- SetUserDirSearchOrder Method—Rearranges the search order of the user directory objects associated with the domain

AddAdmin Method—Adds an Administrator to the Domain

The AddAdmin method adds an administrator to the domain.

Syntax

The AddAdmin method has the following format:

```
Netegrity::PolicyMgtDomain->AddAdmin(admin)
```

Parameters

The AddAdmin method accepts the following parameter:

admin (type)

Specifies the administrator to add to the domain.

Return Value

The AddAdmin method returns one of the following values:

- 0 on success
- -1 if the call was unsuccessful

Remarks

Administrators can create, edit, and delete SiteMinder objects within the domain.

You cannot use the Policy Management API to create an administrator for a particular domain. However, if you use the Administrative UI to create an administrator for a domain, you can add that administrator to another domain by calling the PolicyMgtAffDomain->AddAdmin method.

AddUserDir Method—Associates a User Directory with the Domain

The AddUserDir method associates a user directory with the domain.

Syntax

The AddUserDir method has the following format:

```
Netegrity::PolicyMgtDomain->AddUserDir(userDir)
```

Parameters

The AddUserDir method accepts the following parameter:

userDir (PolicyMgtUserDir)

Specifies the user directory to associate with the domain.

Return Value

The AddUserDir method returns one of the following values:

- 0 on success
- -1 if the call was unsuccessful

Remarks

During user authentication, the user's supplied credentials are checked against the credentials stored in this user directory.

The directory object is appended to the end of the search order. To change the search order, call the PolicyMgtAffDomain->SetUserDirSearchOrder method.

CreatePolicy Method—Creates and Configures a Policy in the Domain

The CreatePolicy method creates and configures a policy in the domain.

Syntax

The CreatePolicy method has the following format:

```
Netegrity::PolicyMgtDomain->CreatePolicy(policyName [, policyDesc] [, enableFlag]
[, activeExpr])
```

Parameters

The CreatePolicy method accepts the following parameters:

policyName (string)

Specifies the name of the policy.

policyDesc (string)

(Optional) Specifies the description of the policy.

enableFlag (int)

(Optional) Specifies whether to enable (1) or disable (0) the policy. Default is enabled.

activeExpr (string)

(Optional) Specifies the active expression of the policy.

Return Value

The CreatePolicy method returns one of the following values:

- A PolicyMgtPolicy object
- **undef** if the call was unsuccessful

CreateRealm Method—Creates and Configures a Top-level Realm in the Domain

The CreateRealm method creates and configures a top-level realm in the domain.

Syntax

The CreateRealm method has the following format:

```
Netegrity::PolicyMgtDomain->CreateRealm(realmName, agent, authScheme [, realmDesc]
[, resFilter] [, procAuthEvents] [, procAzEvents] [, protectAll] [, maxTimeout] [,
idleTimeout] [, syncAudit] [, azUserDir] [, regScheme])
```

Parameters

The CreateRealm method accepts the following parameters:

realmName (string)

Specifies the name of the realm.

agent (PolicyMgtAgent)

Specifies the agent or agent group that protects the realm.

authScheme (PolicyMgtAuthScheme)

Specifies the authentication scheme to associate with the realm.

realmDesc (string)

(Optional) Specifies the realm description.

resFilter (string)

(Optional) Specifies the resource filter for the realm.

procAuthEvents (int)

(Optional) Specifies whether to process authentication events -- 1 to enable or 0 to disable. Default is enabled. Authentication event processing affects performance. If no rules in the realm are to be triggered by authentication events, set this flag to 0.

procAzEvents (int)

(Optional) Specifies whether to process authorization events -- 1 to enable or 0 to disable. Default is enabled. Authorization event processing affects performance. If no rules in the realm are to be triggered by authorization events, set this flag to 0.

protectAll (int)

> (Optional) Specifies whether to activate default resource protection -- 1 to enable or 0 to disable. Default is enabled.

maxTimeout (int)

> (Optional) Specifies the maximum time, in seconds, a user can access the realm before re-authentication is required. Default is 7200 (2 hours).

idleTimeout (int)

> (Optional) Specifies the maximum time, in seconds, a user can remain inactive in the realm before re-authentication is required. Default is 3600 (1 hour).

syncAudit (int)

> (Optional) Specifies lag for enabling synchronous auditing -- 1 to enable or 0 to disable. When this flag is enabled, SiteMinder logs Policy Server and agent actions before it allows access to resources. Default is disabled.

azUserDir (PolicyMgtUserDir)

> (Optional) Specifies The directory where users in the realm will be authorized. Default is the default directory.

regScheme (type)

> (Optional) Specifies the registration scheme used to register new users accessing resources in the realm.

Return Value

The CreateRealm method returns one of the following values:

- A PolicyMgtRealm object
- **undef** if the call was unsuccessful

Remarks

This method creates a realm that is configured for non-persistent sessions. To configure the realm for SiteMinder 5.0 persistent sessions, edit the realm in the Administrative UI.

Note: The Policy Management API only manipulates realms that are direct descendants of the object whose method has been called, as follows:

- For a realm under a domain, you can only manipulate the top-level realms in a domain object.
- For a realm under a realm, you can only manipulate realms that are directly under the parent realm.

CreateResponse Method—Creates a Response

The CreateResponse method creates a response.

Syntax

The CreateResponse method has the following format:

```
Netegrity::PolicyMgtDomain->CreateResponse(resName, agentType [, resDesc])
```

Parameters

The CreateResponse method accepts the following parameters:

resName (string)

Specifies the name of the response.

agentType (PolicyMgtAgentType)

Specifies the agent type associated with the response. Call the PolicyMgtSession->GetAgentType method to get the agent type object.

resDesc (string)

(Optional) Specifies the description of the response.

Return Value

The CreateResponse method returns one of the following values:

- A PolicyMgtResponse object
- **undef** if the call was unsuccessful

Remarks

The agent returns responses based on certain events. For example, if an unauthorized user attempts to access a protected Web page, a response can redirect the user to an HTML page that displays an appropriate message.

CreateResponseGroup Method—Creates a Response Group for the Domain

The CreateResponseGroup method creates a response group for the domain.

Syntax

The CreateResponseGroup method has the following format:

```
Netegrity::PolicyMgtDomain->CreateResponseGroup(groupName, agentType, [,
groupDesc])
```

Parameters

The CreateResponseGroup method accepts the following parameters:

groupName (string)

Specifies the name of the group.

agentType (PolicyMgtAgentType)

Specifies the agent type associated with this response group. Call the PolicyMgtSession->GetAgentType method to get the agent type object.

groupDesc (string)

(Optional) Specifies the description of the group.

Return Value

The CreateResponseGroup method returns one of the following values:

- A PolicyMgtGroup object
- **undef** if the call was unsuccessful

CreateRuleGroup Method—Creates a Rule Group for the Domain

The CreateRuleGroup method creates a rule group for the domain.

Syntax

The CreateRuleGroup method has the following format:

```
Netegrity::PolicyMgtDomain->CreateRuleGroup(groupName, agentType [, groupDesc])
```

Parameters

The CreateRuleGroup method accepts the following parameters:

groupName (string)

Specifies the name of the group.

agentType (PolicyMgtAgentType)

Specifies the agent type associated with this rule group. Call the PolicyMgtSession->GetAgentType method to get the agent type object.

groupDesc (string)

(Optional) Specifies the description of the group.

Return Value

The CreateRuleGroup method returns one of the following values:

- A PolicyMgtGroup object
- **undef** if the call was unsuccessful

DeleteGroup Method—Deletes a Group from the Domain

The DeleteGroup method deletes the specified group in the domain.

Syntax

The DeleteGroup method has the following format:

```
Netegrity::PolicyManagementDomain->DeleteGroup(group)
```

Parameters

The DeleteGroup method accepts the following parameter:

group (PolicyMgrGroup)

Specifies the group to delete.

Return Value

The DeleteGroup method returns one of the following values:

- 0 on success, or the group was not found
- -1 if the call failed

DeletePolicy Method—Deletes a Policy

The DeletePolicy method deletes a policy.

Syntax

The DeletePolicy method has the following format:

```
Netegrity::PolicyMgtDomain->DeletePolicy(policy)
```

Parameters

The DeletePolicy method accepts the following parameter:

policy (PolicyMgtPolicy)

Specifies the policy to delete.

Return Value

The DeletePolicy method returns one of the following values:

- 0 on success
- -1 if the call failed

DeleteRealm Method—Deletes a Realm in the Domain

The DeleteRealm method deletes a top-level realm in the domain.

Syntax

The DeleteRealm method has the following format:

```
Netegrity::PolicyMgtDomain->DeleteRealm(realm)
```

Parameters

The DeleteRealm method accepts the following parameter:

realm (PolicyMgtRealm)

Specifies the realm to delete.

Return Value

The DeleteRealm method returns one of the following values:

- 0 on success, or if the real was not found
- -1 if the call failed

DeleteResponse Method—Deletes a Response

The DeleteResponse method deletes a response.

Syntax

The DeleteResponse method has the following format:

```
Netegrity::PolicyMgtDomain->DeleteResponse(response)
```

Parameters

The DeleteResponse method accepts the following parameter:

response (PolicyMgtResponse)

Specifies the response to delete.

Return Value

The DeleteResponse method returns one of the following values:

- 0 on success
- -1 if the call failed

Description Method—Sets or Retrieves the Description of the Domain

The Description method sets or retrieves the description of the domain.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtDomain->Description([domainDesc])
```

Parameters

The Description method accepts the following parameter:

domainDesc (string)

(Optional) Specifies the description to set.

Return Value

The Description method returns one of the following values:

- A new or existing domain description
- An empty string if unsuccessful

GetAllPolicies Method—Retrieves All Policies Associated with the Domain

The GetAllPolicies method retrieves all policies associated the domain.

Syntax

The GetAllPolicies method has the following format:

```
Netegrity::PolicyMgtDomain->GetAllPolicies( )
```

Parameters

The GetAllPolicies method accepts no parameters.

Return Value

The GetAllPolicies method returns one of the following values

- An array of PolicyMgtPolicy objects
- **undef** if unsuccessful

GetAllRealms Method—Retrieves All Top-level Realms in the Domain

The GetAllRealms method Retrieves all top-level realms in the domain.

Syntax

The GetAllRealms method has the following format:

```
Netegrity::PolicyMgtDomain->GetAllRealms( )
```

Parameters

The GetAllRealms method accepts no parameters.

Return Value

The GetAllRealms method returns one of the following values

- An array of PolicyMgtRealm objects
- **undef** if unsuccessful

Remarks

To retrieve all top-level realms under a realm, call the PolicyMgtRealm->GetAllChildRealms method.

GetAllResponseGroups Method—Retrieves All the Response Groups Associated with the Domain

The GetAllResponseGroups method retrieves all of the response groups associated with the domain.

Syntax

The GetAllResponseGroups method has the following format:

```
Netegrity::PolicyMgtDomain->GetAllResponseGroups( )
```

Parameters

The GetAllResponseGroups method accepts no parameters.

Return Value

The GetAllResponseGroups method returns one of the following values

- An array of PolicyMgtGroup objects
- **undef** if unsuccessful

GetAllResponses Method—Retrieves All Responses Associated with the Domain

The GetAllResponses method retrieves all responses associated with the domain.

Syntax

The GetAllResponses method has the following format:

```
Netegrity::PolicyMgtDomain->GetAllResponses()
```

Parameters

The GetAllResponses method accepts no parameters.

Return Value

The GetAllResponses method returns one of the following values

- An array of PolicyMgtResponse objects
- **undef** if the call was unsuccessful

GetAllRuleGroups Method—Retrieves All Rule Groups Associated with the Domain

The GetAllRuleGroups method retrieves all rule groups associated with the domain.

Syntax

The GetAllRuleGroups method has the following format:

```
Netegrity::PolicyMgtDomain->GetAllRuleGroups( )
```

Parameters

The GetAllRuleGroups method accepts no parameters.

Return Value

The GetAllRuleGroups method returns one of the following values:

- An array of PolicyMgtGroup objects
- **undef** if the call was unsuccessful

GetPolicy Method—Retrieves a Policy in the Domain

The GetPolicy method retrieves a policy in the domain.

Syntax

The GetPolicy method has the following format:

```
Netegrity::PolicyMgtDomain->GetPolicy(policyName)
```

Parameters

The GetPolicy method accepts the following parameter:

policyName (string)

Specifies the policy to retrieve.

Return Value

The GetPolicy method returns one of the following values

- A PolicyMgtPolicy object
- **undef** if the call fails, or if the specified policy does not exist

GetRealm Method—Retrieves a Top-level Realm in the Domain

The GetRealm method retrieves a top-level realm in the domain.

Syntax

The GetRealm method has the following format:

```
Netegrity::PolicyMgtDomain->GetRealm(realmName)
```

Parameters

The GetRealm method accepts the following parameter:

realmName (string)

Specifies the realm to retrieve.

Return Value

The GetRealm method returns one of the following values:

- A PolicyMgtRealm object
- **undef** if the call failed, or if the specified realm does not exist

GetResponse Method—Retrieves a Response Associated with the Domain

The GetResponse method retrieves a response associated with the domain.

Syntax

The GetResponse method has the following format:

```
Netegrity::PolicyMgtDomain->GetResponse(resName)
```

Parameters

The GetResponse method accepts the following parameter:

resName (string)

Specifies the response to retrieve.

Return Value

The GetResponse method returns one of the following values:

- A PolicyMgtResponse object
- **undef** if the call was unsuccessful, or if the specified response does not exist

GetResponseGroup Method—Retrieves the Specified Response Group

The GetResponseGroup method retrieves the specified response group.

Syntax

The GetResponseGroup method has the following format:

```
Netegrity::PolicyMgtDomain->GetResponseGroup(groupName)
```

Parameters

The GetResponseGroup method accepts the following parameter:

groupName (string)

Specifies the name of the response group to retrieve.

Return Value

The GetResponseGroup method returns one of the following values:

- A PolicyMgtGroup object
- **undef** if the call was unsuccessful

GetRuleGroup Method—Retrieves the Specified Rule Group

The GetRuleGroup method retrieves the specified rule group.

Syntax

The GetRuleGroup method has the following format:

```
Netegrity::PolicyMgtDomain->GetRuleGroup(groupName)
```

Parameters

The GetRuleGroup method accepts the following parameter:

groupName (string)

Specifies the name of the group to retrieve.

Return Value

The GetRuleGroup method returns one of the following values:

- A PolicyMgtGroup object
- **undef** if the call was unsuccessful

GetUserDirSearchOrder Method—Retrieves User Directory Objects Associated with the Domain

The GetUserDirSearchOrder method retrieves user directory objects associated with the domain.

Syntax

The GetUserDirSearchOrder method has the following format:

```
Netegrity::PolicyMgtDomain->GetUserDirSearchOrder( )
```

Parameters

The GetUserDirSearchOrder method accepts no parameters:

Return Value

The GetUserDirSearchOrder method returns one of the following values:

- An PolicyMgtUserDir objects
- **undef** if the call was unsuccessful

Remarks

The order of the returned objects is the same order that SiteMinder uses when querying the directories. To change the search order, call the PolicyMgtAffDomain->SetUserDirSearchOrder method.

GlobalPoliciesApply Method—Determines whether the Domain Is Enabled for Global Policies

The GlobalPoliciesApply method sets or retrieves the flag indicating whether the domain is enabled for global policies. If the domain is enabled for global policies, both global and domain-specific policies can apply to the domain.

Syntax

The GlobalPoliciesApply method has the following format:

```
Netegrity::PolicyMgtDomain->GlobalPoliciesApply([globalFlag])
```

Parameters

The GlobalPoliciesApply method accepts the following parameter:

globalFlag (int)

(Optional) Specifies whether to enable the domain for global polices:

- 1 specifies that global policies should be enable
- 0 specifies that global policies should not be enabled

Return Value

The GlobalPoliciesApply method returns one of the following values:

- A new or the existing flag setting

Name Method—Sets or Retrieves the Domain Name

The Name method sets or retrieves the domain name.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtDomain->Name([domainName])
```

Parameters

The Name method accepts the following parameter:

domainName (string)

(Optional) Specifies the name to assign to the domain.

Return Value

The Name method returns one of the following values:

- A new or the existing domain name
- **undef** if the call was unsuccessful

RemoveAdmin Method—Disassociates an Administrator from the Domain

The RemoveAdmin method disassociates an administrator from the domain.

Syntax

The RemoveAdmin method has the following format:

```
Netegrity::PolicyMgtDomain->RemoveAdmin(admin)
```

Parameters

The RemoveAdmin method accepts the following parameter:

admin (PolicyMgtAdmin)

Specifies the administrator to remove from the domain.

Return Value

The RemoveAdmin method returns one of the following values:

- 0 on success
- -1 if the call was unsuccessful

Remarks

See also the PolicyMgtSession->DeleteAdmin method to delete an administrator from the policy store.

You cannot use the Policy Management API to create an administrator for a particular domain. However, if an administrator is associated with a domain either through the Administrative UI or the PolicyMgtAffDomain->AddAdmin method, you can remove that administrator from the domain by calling the RemoveAdmin method.

RemoveUserDir Method—Disassociates the User Directory from the Domain

The RemoveUserDir method disassociates the user directory from the domain.

Syntax

The RemoveUserDir method has the following format:

```
Netegrity::PolicyMgtDomain->RemoveUserDir(userDir)
```

Parameters

The RemoveUserDir method accepts the following parameter:

userDir (PolicyMgtUserDir)

Specifies the user directory to disassociate from the domain.

Return Value

The RemoveUserDir method returns one of the following values:

- 0 on success
- -1 if the call was unsuccessful

SetUserDirSearchOrder Method—Rearranges the Search Order of the User Directory Objects

The SetUserDirSearchOrder method rearranges the search order of the user directory objects associated with the domain.

Syntax

The SetUserDirSearchOrder method has the following format:

```
Netegrity::PolicyMgtDomain->SetUserDirSearchOrder(dirArray)
```

Parameters

The SetUserDirSearchOrder method accepts the following parameter:

dirArray ()

Specifies a reference to an array of user directory objects (for example: \@myarray).

Return Value

The SetUserDirSearchOrder method returns one of the following values:

- An array of PolicyMgtUserDir objects
- **undef** if the call was unsuccessful

Group Methods

The following methods act on PolicyMgtGroup objects. This object can contain either PolicyMgtAgent objects, PolicyMgtResponse objects, PolicyMgtRule objects, or nested PolicyMgtGroup objects.

- Add Method—Adds an agent, response, rule, or nested group object to the group
- Contains Method—Checks whether the group contains the specified agent, response, rule, or nested group object
- Description Method—Sets or retrieves the description of the group
- GetAgent Method—Retrieves the specified agent object from the group
- GetAgentGroup Method—Retrieves an agent group object nested within the group
- GetAgentType Method—Retrieves the type of the agent objects contained in the group
- GetAllAgentGroups Method—Retrieves an array of all the agent group objects nested in the group
- GetAllAgents Method— Retrieves an array of all the agent objects in the group
- GetAllResponseGroups Method—Retrieves an array of all the response group objects nested in the group
- GetAllResponses Method—Retrieves an array of all the response objects in the group
- GetAllRuleGroups Method—Retrieves an array of all the rule group objects nested in the group
- GetAllRules Method—Retrieves an array of all the rule objects in the group
- GetResponse Method—Retrieves the specified response object from the group
- GetResponseGroup Method—Retrieves a response group object nested within the group
- GetRule Method—Retrieves the specified rule object from the group
- GetRuleGroup Method—Retrieves a rule group object nested within the group
- Name Method—Sets or retrieves the group name
- Remove Method—Removes the specified group member from the group

Add Method—Adds an Agent, Response, Rule, or Nested Group Object to the Group

The Add method adds an agent, response, rule, or nested group object to the group.

Syntax

The Add method has the following format:

```
Netegrity::PolicyMgtGroup->Add(newMember)
```

Parameters

The Add method accepts the following parameter:

newMember (*objectType*)

Specifies the member to add to the group. *objectType* can be any one of the following:

- PolicyMgtAgent
- PolicyMgtResponse
- PolicyMgtRule
- PolicyMgtGroup

Return Value

The Add method returns one of the following values:

- 0 on success
- -1 if the call was unsuccessful

Contains Method—Determines whether the Group Contains the Specified Agent, Response, Rule, or Nested Group Object

The Contains method determines whether the group contains the specified agent, response, rule, or nested group object.

Syntax

The Contains method has the following format:

```
Netegrity::PolicyMgtGroup->Contains(object)
```

Parameters

The Contains method accepts the following parameter:

object (objectType)

Specifies the object to check. *objectType* can be any one of the following:

- PolicyMgtAgent
- PolicyMgtResponse
- PolicyMgtRule
- PolicyMgtGroup

Return Value

The Contains method returns one of the following values:

- 1 if the group contains the specified object
- 0 if the group does not contain the specified object
- **undef** if the call was unsuccessful

Description Method—Sets or Retrieves the Description of the Group Object

The Description method sets or retrieves the description of the group object.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtGroup->Description([Description])
```

Parameters

The Description method accepts the following parameter:

Description (string)

(Optional) Specifies the description to set.

Return Value

The Description method returns one of the following values:

- A new or existing description
- An empty string if the call was unsuccessful

GetAgent Method—Retrieves the Specified Agent Object from the Group

The GetAgent method retrieves the specified agent object from the group.

Syntax

The GetAgent method has the following format:

```
Netegrity::PolicyMgtGroup->GetAgent(agentName)
```

Parameters

The GetAgent method accepts the following parameter:

agentName (string)

Specifies the name of the agent to retrieve.

Return Value

The GetAgent method returns one of the following values:

- A PolicyMgtAgent object
- **undef** if no such agent is found, if the group contains objects of another type, or if the call was unsuccessful

GetAgentGroup Method—Retrieves an Agent Group Object Nested within the Group

The GetAgentGroup method retrieves an agent group object nested within the group.

Syntax

The GetAgentGroup method has the following format:

```
Netegrity::PolicyMgtGroup->GetAgentGroup(groupName)
```

Parameters

The GetAgentGroup method accepts the following parameter:

groupName (string)

Specifies the name of the agent group to retrieve.

Return Value

The GetAgentGroup method returns one of the following values:

- A PolicyMgtGroup object
- **undef** if the call was unsuccessful, or if the group does not exist

GetAgentType Method—Retrieves the Type of the Agent Objects Contained in the Group

The GetAgentType method retrieves the type of the agent objects contained in the group (for example, Web Agent).

Syntax

The GetAgentType method has the following format:

```
Netegrity::PolicyMgtGroup->GetAgentType( )
```

Parameters

The GetAgentType method accepts no parameters:

Return Value

The GetAgentType method returns one of the following values:

- A PolicyMgtAgentType object
- **undef** if the call was unsuccessful

GetAllAgentGroups Method—Retrieves All the Agent Group Objects Nested within the Group

The GetAllAgentGroups method retrieves all the agent group objects nested within the group.

Syntax

The GetAllAgentGroups method has the following format:

```
Netegrity::PolicyMgtGroup->GetAllAgentGroups( )
```

Parameters

The GetAllAgentGroups method accepts no parameters.

Return Value

The GetAllAgentGroups method returns one of the following values:

- An array of PolicyMgtGroup objects
- **undef** if the call is unsuccessful

GetAllAgents Method—Retrieves All the Agent Objects in the Group

The GetAllAgents method retrieves all the agent objects in the group.

Syntax

The GetAllAgents method has the following format:

```
Netegrity::PolicyMgtGroup->GetAllAgents( )
```

Parameters

The GetAllAgents method accepts no parameters.

Return Value

The GetAllAgents method returns one of the following values:

- An array of PolicyMgtAgent objects
- **undef** if no agents are found, if the group contains objects of another type, or if the call is unsuccessful

GetAllResponseGroups Method—Retrieves All the Response Group Objects Nested within the Group

The GetAllResponseGroups method retrieves all the response group objects nested within the group.

Syntax

The GetAllResponseGroups method has the following format:

```
Netegrity::PolicyMgtGroup->GetAllResponseGroups( )
```

Parameters

The GetAllResponseGroups method accepts no parameters.

Return Value

The GetAllResponseGroups method returns one of the following values:

- An array of PolicyMgtGroup objects
- **undef** if no response groups are found, if the group contains objects of another type, or if the call is unsuccessful.

GetAllResponses Method—Retrieves All the Response Objects in the Group

The GetAllResponses method retrieves all the response objects in the group.

Syntax

The GetAllResponses method has the following format:

```
Netegrity::PolicyMgtGroup->GetAllResponses( )
```

Parameters

The GetAllResponses method accepts no parameters.

Return Value

The GetAllResponses method returns one of the following values:

- An array of PolicyMgtResponse objects
- **undef** if no response objects are found, if the group contains objects of another type, or if the call is unsuccessful.

GetAllRuleGroups Method—Retrieves All the Rule Group Objects Nested within the Group

The GetAllRuleGroups method retrieves all the rule group objects nested within the group.

Syntax

The GetAllRuleGroups method has the following format:

```
Netegrity::PolicyMgtGroup->GetAllRuleGroups( )
```

Parameters

The GetAllRuleGroups method accepts no parameters.

Return Value

The GetAllRuleGroups method returns one of the following values:

- An array of PolicyMgtGroup objects
- **undef** if no rule groups are found, if the groups contains objects of another type, or if the call is unsuccessful.

GetAllRules Method—Retrieves All the Rule Objects in the Group

The GetAllRules method retrieves all the rule objects in the group.

Syntax

The GetAllRules method has the following format:

```
Netegrity::PolicyMgtGroup->GetAllRules( )
```

Parameters

The GetAllRules method accepts no parameters.

Return Value

The GetAllRules method returns one of the following values:

- An array of PolicyMgtRule objects
- **undef** if no rule objects are found, if the group contains objects of another type, or if the call is unsuccessful

GetResponse Method—Retrieves the Specified Response Object from the Group

The GetResponse method retrieves the specified response object from the group.

Syntax

The GetResponse method has the following format:

```
Netegrity::PolicyMgtGroup->GetResponse(responseName)
```

Parameters

The GetResponse method accepts the following parameter:

responseName (type)

Specifies the name of the response to retrieve.

Return Value

The GetResponse method returns one of the following values:

- A PolicyMgtResponse object
- **undef** if no such response is found, if the group contains objects of another type, or if the call is unsuccessful

GetResponseGroup Method—Retrieves a Response Group Object Nested within the Group

The GetResponseGroup method retrieves a response group object nested within the group.

Syntax

The GetResponseGroup method has the following format:

```
Netegrity::PolicyMgtGroup->GetResponseGroup(groupName)
```

Parameters

The GetResponseGroup method accepts the following parameter:

groupName (string)

Specifies the name of the response group to retrieve.

Return Value

The GetResponseGroup method returns one of the following values:

- A PolicyMgtGroup object
- **undef** if the group does not exist, or if the call is unsuccessful

GetRule Method—Retrieves the Specified Rule Object from the Group

The GetRule method retrieves the specified rule object from the group.

Syntax

The GetRule method has the following format:

```
Netegrity::PolicyMgtGroup->GetRule(ruleName)
```

Parameters

The GetRule method accepts the following parameter:

ruleName (string)

Specifies the name of the rule to retrieve.

Return Value

The GetRule method returns one of the following values:

- A PolicyMgtRule object
- **undef** if no such rule is found, if the group contains objects of another type, or if the call is unsuccessful

GetRuleGroup Method—Retrieves a Rule Group Object Nested within the Group

The GetRuleGroup method retrieves a rule group object nested within the group.

Syntax

The GetRuleGroup method has the following format:

```
Netegrity::PolicyMgtGroup->GetRuleGroup(groupName)
```

Parameters

The GetRuleGroup method accepts the following parameter:

groupName (string)

Specifies the name of the rule group to retrieve.

Return Value

The GetRuleGroup method returns one of the following values:

- A PolicyMgtGroup object
- **undef** if the group does not exist, or if the call is unsuccessful

Name Method—Sets or Retrieves the Name of the Group Object

The Name method sets or retrieves the name of the group object.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtGroup->Name([Name])
```

Parameters

The Name method accepts the following parameter:

Name (string)

(Optional) Specifies the name to set.

Return Value

The Name method returns one of the following values:

- The new or existing name
- **undef** if the call is unsuccessful

Remove Method—Removes the Specified Group Member from the Group

The Remove method removes the specified group member from the group.

Syntax

The Remove method has the following format:

```
Netegrity::PolicyMgtGroup->Remove(member)
```

Parameters

The Remove method accepts the following parameters:

member (*objectType*)

Specifies the group member to remove, which can be any of the following object types:

- PolicyMgtAgent
- PolicyMgtResponse
- PolicyMgtRule
- PolicyMgtGroup

Return Value

The Remove method returns one of the following values:

- 0 on success
- **undef** if the call is unsuccessful

Host Configuration Methods

The following methods act on PolicyMgtHostConfig objects:

- AddCluster Method—Adds a cluster object to the host configuration
- AddServer Method—Adds a Policy Server object to the host configuration
- Description Method—Sets or retrieves the description of the host configuration
- EnableFailover Method—Sets or retrieves the EnableFailover value for a host configuration
- FailoverThreshold Method—Sets or retrieves the failover threshold percentage for the clusters in the host configuration
- GetAllClusters Method—Retrieves an array of cluster objects
- GetAllServers Method—Retrieves an array of Policy Server connectivity objects
- MaxSocketsPerPort Method—Sets or retrieves the MaxSocketsPerPort value for a host configuration
- MinSocketsPerPort Method—Sets or retrieves the MinSocketsPerPort value for a host configuration
- Name Method—Sets or retrieves the host configuration name
- NewSocketStep Method—Sets or retrieves the NewSocketStep value for a host configuration
- RemoveAllClusters Method—Removes all PolicyMgtCluster objects associated with this host configuration
- RemoveAllServers Method—Removes all PolicyMgtServer objects associated with this host configuration
- RequestTimeout Method—Sets or retrieves the RequestTimeout value for a host configuration

AddCluster Method—Adds an Empty Cluster to the Host Configuration

The AddCluster method adds an empty cluster to the host configuration. Call the AddServer method to populate the cluster with servers.

Syntax

The AddCluster method has the following format:

```
Netegrity::PolicyMgtHostConfig->AddCluster( )
```

Parameters

The AddCluster method accepts no parameters.

Return Value

The AddCluster method returns one of the following values:

- An empty PolicyMgtCluster object
- **undef** if the call is unsuccessful

Remarks

The clusters in a host configuration are referenced in a cluster array. When you add a cluster, the cluster is added to the end of the cluster array. The order in which you add clusters to a host configuration object determines the failover sequence. The first cluster you add (that is, the first cluster in the cluster array) is the primary cluster. This is the first cluster in the failover sequence that SiteMinder sends requests to. If there are not enough available servers in the primary cluster (that is, if the number of available servers in the cluster falls below the failover threshold), failover to the next cluster occurs (the second cluster that was added to the host configuration object). If that cluster also fails, failover to the third cluster added to the host configuration object occurs, and so on.

AddServer Method—Adds a Non-clustered Server to the Host Configuration

The AddServer method adds a non-clustered server to the host configuration.

Syntax

The AddServer method has the following format:

```
Netegrity::PolicyMgtHostConfig->AddServer(Host [, AcctPort] [, AuthPort] [, AzPort])
```

Parameters

The AddServer method accepts the following parameters:

Host (string)

Specifies the IP address of the Policy Server.

AcctPort (string)

(Optional) Specifies the IP port for the accounting server.

AuthPort (string)

(Optional) Specifies the IP port for the authentication server.

AzPort (string)

(Optional) Specifies the IP port for the authorization server.

Return Value

The AddServer method returns one of the following values:

- 0 on success
- -1 on failure

Remarks

The single-process Policy Server introduced in SiteMinder v6.0 combines the previously separate Authentication, Authorization, and Accounting processes into one combined process whose requests go through one TCP port. As a result, the arguments AcctPort, AuthPort, and AzPort all reference the same port number. The three arguments are maintained for backward compatibility.

To add a server to a cluster, call the PolicyMgtCluster->AddServer method.

Description Method—Sets or Retrieves the Description of the Host Configuration Object

The Description method sets or retrieves the description of the host configuration object.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtHostConfig->Description([Description])
```

Parameters

The method accepts the following parameter:

Description (string)

(Optional) Specifies the description to set.

Return Value

The Description method returns one of the following values:

- The new or existing description
- **undef** if the call is unsuccessful

EnableFailover Method—Sets or Retrieves the Enable Failover Flag

The EnableFailover method Sets or retrieves the enable failover flag. This flag determines whether an agent and the Policy Server should communicate through failover or round-robin.

Syntax

The EnableFailover method has the following format:

```
Netegrity::PolicyMgtHostConfig->EnableFailover([EnableFailover])
```

Parameters

The EnableFailover method accepts the following parameter:

EnableFailover (int)

(Optional) Specifies the value of the flag to set.

Return Value

The EnableFailover method returns one of the following values:

- The new or existing flag setting:
  - 1 for failover
  - 0 for round-robin
- -1 if the call is unsuccessful

FailoverThreshold Method—Sets or Retrieves the Failover Threshold Percentage

The FailoverThreshold method sets or retrieves the failover threshold percentage for the clusters in the host configuration.

Syntax

The FailoverThreshold method has the following format:

```
Netegrity::PolicyMgtHostConfig->FailoverThreshold([FailoverThreshold])
```

Parameters

The FailoverThreshold method accepts the following parameter:

FailoverThreshold (int)

(Optional) Specifies the failover threshold percentage to set.

Return Value

The FailoverThreshold method returns one of the following values:

- The new or existing failover threshold percentage
- **undef** if the call is unsuccessful

Remarks

The threshold percentage represents the minimum number of servers in a cluster that must be available for requests. If the number of available servers falls below the threshold, failover to the next cluster occurs. To determine the number of servers represented by the percentage, multiply the threshold percentage by the number of servers in a cluster, rounding up to the next highest integer. For example:

- With a 60-percent failover threshold for a cluster of five servers, failover to the next cluster occurs when the number of available servers in the cluster falls below 3.
- With a 61-percent failover threshold for the same cluster, failover occurs when the number of available servers falls below 4.

GetAllClusters Method—Retrieves an Array of Policy Management Cluster Objects

The GetAllClusters method retrieves an array of Policy Management Cluster objects.

Syntax

The GetAllClusters method has the following format:

```
Netegrity::PolicyMgtHostConfig->GetAllClusters()
```

Parameters

The GetAllClusters method accepts no parameters.

Return Value

The GetAllClusters method returns one of the following values:

- An array of PolicyMgtCluster objects
- **undef** if the call is unsuccessful

GetAllServers Method—Retrieves an Array of Non-clustered Server Objects

The GetAllServers method retrieves an array of non-clustered server objects for the host configuration.

Syntax

The GetAllServers method has the following format:

```
Netegrity::PolicyMgtHostConfig->GetAllServers( )
```

Parameters

The GetAllServers method accepts no parameters.

Return Value

The GetAllServers method returns one of the following values:

- An array of PolicyMgtServer objects
- **undef** if no server objects are found, or if the call is unsuccessful

Remarks

To retrieve the servers that are members of clusters, call the PolicyMgtCluster->GetAllServers method.

MaxSocketsPerPort Method—Sets or Retrieves the Maximum Number of TCP/IP Sockets

The MaxSocketsPerPort method sets or retrieves the maximum number of TCP/IP sockets that can be opened between an agent and the Policy Server.

Syntax

The MaxSocketsPerPort method has the following format:

```
Netegrity::PolicyMgtHostConfig->MaxSocketsPerPort([MaxSocketsPerPort])
```

Parameters

The MaxSocketsPerPort method accepts the following parameter:

MaxSocketsPerPort (int)

(Optional) Specifies the new maximum number of sockets per port.

Return Value

The MaxSocketsPerPort method returns one of the following values:

- The new or existing setting for maximum number of sockets
- -1 if the call is unsuccessful

MinSocketsPerPort Method—Sets or Retrieves the Minimum Number of TCP/IP Sockets

The MinSocketsPerPort method sets or retrieves the minimum number of TCP/IP sockets that should be opened between an agent and the Policy Server.

Syntax

The MinSocketsPerPort method has the following format:

```
Netegrity::PolicyMgtHostConfig->MinSocketsPerPort([MinSocketsPerPort])
```

Parameters

The MinSocketsPerPort method accepts the following parameter:

MinSocketsPerPort (int)

(Optional) Specifies the new minimum socket value.

Return Value

The MinSocketsPerPort method returns one of the following values:

- The new or existing setting for minimum number of sockets
- -1 if the call is unsuccessful

Name Method—Sets or Retrieves the Name of the Host Configuration Object

The Name method sets or retrieves the name of the host configuration object.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtHostConfig->Name([Name])
```

Parameters

The Name method accepts the following parameter:

Name (string)

(Optional) Specifies the name to set.

Return Value

The Name method returns one of the following values:

- The new or existing name
- **undef** if the call is unsuccessful

NewSocketStep Method—Sets or Retrieves the New Socket Step Value for the Host Configuration

The NewSocketStep method sets or retrieves the new socket step value for the host configuration. This value is an incremental number of TCP/IP sockets that should be opened between an agent and the Policy Server when demand increases.

Syntax

The NewSocketStep method has the following format:

```
Netegrity::PolicyMgtHostConfig->NewSocketStep([NewSocketStep])
```

Parameters

The NewSocketStep method accepts the following parameter:

NewSocketStep (int)

(Optional) Specifies the new sockets step value to set.

Return Value

The NewSocketStep method returns one of the following values:

- The new or existing sockets step value
- -1 if the call is unsuccessful

RemoveAllClusters Method—Removes All Cluster Objects Associated with This Host Configuration

The RemoveAllClusters method removes all cluster objects associated with this host configuration.

Syntax

The RemoveAllClusters method has the following format:

```
Netegrity::PolicyMgtHostConfig->RemoveAllClusters()
```

Parameters

The RemoveAllClusters method accepts no parameters.

Return Value

The RemoveAllClusters method returns one of the following values:

- 0 if the call is successful
- -1 if the call is unsuccessful

RemoveAllServers Method—Removes All Non-clustered Policy Server Objects from the Host Configuration

The RemoveAllServers method removes all non-clustered PolicyMgtServer objects from the host configuration.

Syntax

The RemoveAllServers method has the following format:

```
Netegrity::PolicyMgtHostConfig->RemoveAllServers()
```

Parameters

The RemoveAllServers method accepts no parameters.

Return Value

The RemoveAllServers method returns one of the following values:

- 0 if the call is successful
- -1 if the call is unsuccessful

RequestTimeout Method—Sets or Retrieves the Request Timeout Value

The RequestTimeout method sets or retrieves the request timeout value, in seconds. This value represents the length of time that an agent will wait for a response from the Policy Server.

Syntax

The RequestTimeout method has the following format:

```
Netegrity::PolicyMgtHostConfig->RequestTimeout([RequestTimeout])
```

Parameters

The RequestTimeout method accepts the following parameter:

RequestTimeout (int)

(Optional) Specifies the new timeout value to set.

Return Value

The RequestTimeout method returns one of the following values:

- The new or existing timeout value
- -1 if the call is unsuccessful

Initialization Methods

The following methods act on PolicyMgtAPI objects:

- CreateSession Method—Creates a Policy Server session
- DisableAudit Method—Enables or disables user and session auditing
- DisableCacheUpdates Method—Deprecated as of SiteMinder v6.0
- DisableManagementWatchDog Method—Enables or disables the SiteMinder Management Watchdog
- DisableValidation Method—Enables or disables the validation of Policy Server objects
- EnableCache Method—Deprecated as of SiteMinder v6.0
- LoadAgentTypeDictionary Method—Enables or disables the loading of the agent type dictionary by the Policy Server
- New Method—Constructor for the Policy Management API
- PreLoadCache Method—Enables or disables the preloading of caches by the Policy Server
- PrintDebugTrace Method—Enables or disables the printing of debug (trace) information to the console

CreateSession Method—Creates a Policy Server Session

The CreateSession method creates a Policy Server session. A session is required before Policy Server objects can be manipulated. All necessary initializations and logging are performed at this stage.

Syntax

The CreateSession method has the following format:

```
Netegrity::PolicyMgtAPI->CreateSession(username, userpwd[, clientIP])
```

Parameters

The CreateSession method accepts the following parameters:

username (string)

Specifies the administrator's login ID.

userpwd (string)

Specifies the administrator's password.

clientIP (string)

(Optional) Specifies the IP address of the local machine.

The CreateSession method returns one of the following values:

- A PolicyMgtSession object
- **undef** if the call is unsuccessful

DisableAudit Method—Sets the Flag to Enable or Disable Auditing

The DisableAudit method sets a flag to enable or disable auditing.

Syntax

The DisableAudit method has the following format:

```
Netegrity::PolicyMgtAPI->DisableAudit([auditFlag])
```

Parameters

The DisableAudit method accepts the following parameter:

auditFlag (int)

(Optional) Specifies the value to set the flag:

- 0 to enable auditing
- 1 to disable auditing

Return Value

The DisableAudit method returns one of the following values:

- The existing enabled state (0 or 1) if no argument is specified.
- The new enabled state if a flag value is passed to the method.

Remarks

Reads or sets the enabled state for the following operations:

- Auditing of user activities, including authentication, authorization, and administration activities. Administration activities include changes to the policy store.
- Monitoring of user sessions.

The default state is enabled. The enabled state reverts to the default at the start of each new session.

Attempting to set the enabled state has no effect after the PolicyMgtAPI->CreateSession method is called.

DisableCacheUpdates Method—Deprecated

The DisbleCacheUpdates method is deprecated in SiteMinder v6.0. Caches affected by this method are automatically enabled.

DisableManagementWatchDog Method—Reads or sets the Enabled State of the SiteMinder Management Watchdog

The DisableManagementWatchdog method reads or sets the enabled state of the SiteMinder Management Watchdog.

Note: The watchdog is used internally and should not be disabled.

Syntax

The DisableManagementWatchdog method has the following format:

```
Netegrity::PolicyMgtAPI->DisableManagementWatchDog([watchDogFlag])
```

Parameters

The DisableManagementWatchdog method accepts the following parameter:

watchDogFlag (int)

(Optional) Specifies the value of the flag to set:

- 0 to enable the WatchDog
- 1 to disable the WatchDog

Return Value

The DisableManagementWatchdog method returns one of the following values:

- The existing enabled state (0 or 1) if no argument is specified.
- The new enabled state if a flag value is passed to the method.

Remarks

The default state is enabled. The enabled state reverts to the default at the start of each new session.

Attempting to set the enabled state has no effect after PolicyMgtAPI->CreateSession is called.

DisableValidation Method—Reads or Sets the Enabled State for Validation of Policy Server Objects

The DisableValidation method reads or sets the enabled state regarding validation of Policy Server objects.

Syntax

The DisableValidation method has the following format:

```
Netegrity::PolicyMgtAPI->DisableValidation([validationFlag])
```

Parameters

The DisableValidation method accepts the following parameter:

validationFlag (int)

(Optional) Specifies the value to set the flag::

- 0 to enable validation
- 1 to disable validation

Return Value

The DisableValidation method returns one of the following values:

- The existing enabled state (0 or 1) if no argument is specified.
- The new enabled state if a flag value is passed to the method.

Remarks

The default state is enabled. The enabled state reverts to the default at the start of each new session.

Attempting to set the enabled state has no effect after the PolicyMgtAPI->CreateSession method is called.

EnableCache Method—Deprecated

The EnableCache method is deprecated in SiteMinder v6.0. Beginning with this release, caches affected by this method are automatically enabled.

LoadAgentTypeDictionary Method—Reads or Sets the Enabled State for the Agent Type Dictionary

The LoadAgentTypeDirectory method reads or sets the enabled state for the loading of the agent type dictionary by the Policy Server.

Syntax

The LoadAgentTypeDirectory method has the following format:

```
Netegrity::PolicyMgtAPI->LoadAgentTypeDictionary([loadFlag])
```

Parameters

The LoadAgentTypeDirectory method accepts the following parameter:

loadFlag (int)

(Optional) Specifies the value to set the flag:

0 to disable loading the agent type dictionary

1 to enable loading the agent type dictionary

Return Value

The LoadAgentTypeDirectory method returns one of the following values:

- The existing enabled state (0 or 1) if no argument is specified.
- The new enabled state if a flag value is passed to the method.

Remarks

The default state is disabled. The enabled state reverts to the default at the start of each new session.

Attempting to set the enabled state has no effect after the PolicyMgtAPI->CreateSession method is called.

New Method—Constructor for the Policy Management API

The New method is the constructor for the Policy Management API. This method must be called before the Policy Management API can be used.

Syntax

The New method has the following format:

```
Netegrity::PolicyMgtAPI->New( )
```

Parameters

The New method accepts no parameters.

Return Value

The New method returns one of the following values:

- A PolicyMgtAPI object
- **undef** if the call is unsuccessful

PreLoadCache Method—Reads or Sets the Enabled State for Preloading of Caches

The PreLoadCache method reads or sets the enabled state for preloading of caches by the Policy Server.

Syntax

The PreLoadCache method has the following format:

```
Netegrity::PolicyMgtAPI->PreLoadCache([cacheFlag])
```

Parameters

The PreLoadCache method accepts the following parameter:

cacheFlag (int)

(Optional) Specifies the value to set the flag:

- 0 disables cache preloading
- 1 enables cache preloading

Return Value

The PreLoadCache method returns one of the following values:

- The existing enabled state (0 or 1) if no argument is specified.
- The new enabled state if a flag value is passed to the method.

Remarks

The default state is disabled. The enabled state reverts to the default at the start of each new session.

Attempting to set the enabled state has no effect after the PolicyMgtAPI->CreateSession method is called.

Note: By disabling this flag, you can reduce the time it takes for Policy Management scripts to make policy store changes.

PrintDebugTrace Method—Enables or Disables Printing Debug (Trace) Information Example

The PrintDebugTrace method enables or disables the printing of debug (trace) information to the console.

Syntax

The PrintDebugTrace method has the following format:

```
Netegrity::PolicyMgtAPI->PrintDebugTrace([debugFlag])
```

Parameters

The PrintDebugTrace method accepts the following parameter:

debugFlag (int)

(Optional) Specifies the value to set the flag:

- 0 disables trace printing
- 1 enables trace printing

Return Value

The PrintDebugTrace method returns one of the following values:

- 0 if trace printing is disabled
- 1 if trace printing is enabled

IP Configuration Methods

The following methods act on PolicyMgtIPConfig objects. These methods manage IP address restrictions (that is, IP addresses where requests must originate).

- GetEndIPAddress Method—Retrieves the ending IP address in a range of accepted IP addresses
- GetHostName Method—Retrieves the host name associated with the IP address restriction
- GetIPAddress Method—Retrieves the IP address restriction or the first IP address in a range of accepted IP addresses
- GetSubnetMask Method—Retrieves the subnet mask used to derive the IP address restriction
- GetType Method—Retrieves the type of IP address restriction

GetEndIPAddress Method—Retrieves the Ending IP Address

The GetEndIPAddress method retrieves the ending IP address for an IP address range.

Syntax

The GetEndIPAddress method has the following format:

```
Netegrity::PolicyMgtIPConfig->GetEndIPAddress( )
```

Parameters

The GetEndIPAddress method accepts no parameters.

Return Value

The GetEndIPAddress method returns one of the following values:

- The ending IP address in a range of accepted IP addresses.
- **undef** if the call is unsuccessful

Remarks

See the method PolicyMgtAffiliate->CreateIPConfigRange (see page 184) for more information.

GetHostName Method—Retrieves the Host Name Associated with a Host Name IP Address Restriction

The GetHostName method retrieves the host name associated with a host name IP address restriction.

Syntax

The GetHostName method has the following format:

```
Netegrity::PolicyMgtIPConfig->GetHostName()
```

Parameters

The GetHostName method accepts no parameters.

Return Value

The GetHostName method returns one of the following values:

- The host name of the machine where requests originate
- **undef** if the call is unsuccessful

Remarks

See the method PolicyMgtAffiliate->CreateIPConfigHostName (see page 183) for more information.

GetIPAddress Method— Retrieves an IP address for an IP address restriction

The GetIPAddress method retrieves an IP address for an IP address restriction, as follows:

- For IPCFG_TYPE_SINGLEHOST IP addresses, retrieves the IP address of the machine where requests must originate for the policy to fire. See the PolicyMgtAffiliate->CreateIPConfigSingleHost (see page 185) method description for more information.
- For IPCFG_TYPE_RANGE IP address restrictions, retrieves the starting IP address in the range of accepted IP addresses. See the description of the PolicyMgtAffiliate->CreateIPConfigRange (see page 184) method for more information.

To determine the type of IP address restriction, call the GetType method.

Syntax

The GetIPAddress method has the following format:

```
Netegrity::PolicyMgtIPConfig->GetIPAddress()
```

Parameters

The GetIPAddress method accepts no parameters.

Return Value

The GetIPAddress method returns one of the following values:

- The IP address where requests must originate, or the starting address in a range of accepted addresses.
- **undef** if the call is unsuccessful

GetSubnetMask Method—Retrieves the Subnet Mask for a Subnet Address

The GetSubnetMask method retrieves the subnet mask for a subnet address derived from a specified subnet mask and IP address.

Syntax

The GetSubnetMask method has the following format:

```
Netegrity::PolicyMgtIPConfig->GetSubnetMask( )
```

Parameters

The GetSubnetMask method accepts no parameters.

Return Value

The GetSubnetMask method returns one of the following values:

- The subnet mask
- **undef** if the call is unsuccessful

Remarks

See the description of the PolicyMgtPolicy->CreateIPConfigSubnetMask (see page 346) method for more information.

GetType Method—Retrieves the Type of the IP Address Restriction

The GetType method retrieves the type of the IP address restriction. An IP address restriction specifies where a request must originate before it can be honored.

Syntax

The GetType method has the following format:

```
Netegrity::PolicyMgtIPConfig->GetType()
```

Parameters

The GetType method accepts no parameters.

Return Value

The GetType method returns one of the following values:

- IPCFG_TYPE_SINGLEHOST (Value=1). The request must come from the specified IP address. This type of IP address restriction is created with the PolicyMgtAffiliate->CreateIPConfigSingleHost method.
- IPCFG_TYPE_HOSTNAME (Value=2). The request must come from a machine with a specific host name. This type of IP address restriction is created with the PolicyMgtAffiliate->CreateIPConfigHostName method.
- IPCFG_TYPE_SUBNETMASK (Value=3). The request must come from the specified subnet mask. This type of IP address restriction is created with the PolicyMgtPolicy->CreateIPConfigSubnetMask method.
- IPCFG_TYPE_RANGE (Value=4). The request must come from a range of IP addresses. This type of IP address restriction is created with the PolicyMgtAffiliate->CreateIPConfigRange method.
- **undef** if the call is unsuccessful

ODBC Query Scheme Methods

The following methods act on PolicyMgtODBCQueryScheme objects:

- Description Method—Sets or retrieves the description of the ODBC query scheme
- Name Method—Sets or retrieves the ODBC query scheme name
- QueryAuthenticateUser Method—Sets or retrieves the query that retrieves a user's password
- QueryEnumerate Method—Sets or retrieves the query that lists the names of user objects in the directory
- QueryGetGroupProp Method—Retrieves the value of a group property
- QueryGetGroupProps Method—Retrieves a comma-separated list of group properties
- QueryGetGroups Method—Retrieves the names of the groups that the user is a member of
- QueryGetObjInfo Method—Fetches the class of the object
- QueryGetUserProp Method—Retrieves the value of a user property
- QueryGetUserProps Method—Retrieves a comma-separated list of user properties
- QueryGetGroups Method—Retrieves the names of the groups that the user is a member of
- QueryGetObjInfo Method—Fetches the class of the object

- QueryGetUserProp Method—Retrieves the value of a user property
- QueryGetUserProps Method—Retrieves a comma-separated list of user properties
- QueryInitUser Method—Determines whether a particular user exists in the database
- QueryIsGroupMember—Lists the group membership of a particular user
- QueryLookup Method—Returns objects based on a property specified in a group table
- QueryLookupUser Method—Returns a user name based on a property specified in the user table
- QuerySetGroupProp Method—Sets the value of a group property
- QuerySetPassword Method—Changes a user password
- QuerySetUserProp Method—Sets or retrieves a query that sets the value of a user property

Description Method—Sets or Retrieves the Description of the ODBC Query Scheme

The Description method sets or retrieves the description of the ODBC query scheme.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->Description([schemeDesc])
```

Parameters

The Description method accepts the following parameter:

schemeDesc (string)

(Optional) Specifies the description of the ODBC query scheme.

Return Value

The Description method returns one of the following values:

- The new or existing ODBC query scheme description
- An empty if the call is unsuccessful

Name Method—Sets or Retrieves the ODBC Query Scheme Name

The Name method sets or retrieves the ODBC query scheme name.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->Name([schemeName])
```

Parameters

The Name method accepts the following parameter:

schemeName (string)

Specifies the ODBC query scheme name.

Return Value

The Name method returns one of the following values:

- The new or existing ODBC query scheme name
- **undef** if the call is unsuccessful

QueryAuthenticateUser Method—Sets or Retrieves a Query that Fetches a User's Password

The QueryAuthenticateUser method sets or retrieves a query that fetches a user's password.

Syntax

The QueryAuthenticateUser method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryAuthenticateUser([queryAuthUser])
```

Parameters

The QueryAuthenticateUser method accepts the following parameter:

queryAuthUser (string)

(Optional) Specifies the query that fetches a user's password.

Return Value

The QueryAuthenticateUser method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expression is a placeholder for the user's name parameter to be supplied by SiteMinder when the query is executed:

```
select Name from SmUser where Name = '%s' and Password = '%s'
```

If you are configuring a query scheme for an Oracle database and you are using Oracle's encrypted password feature, replace the entire query string with the word connect. Using the word connect for this query indicates to SiteMinder that a user's name and password should be evaluated by the Oracle encrypted password feature.

QueryEnumerate Method—Sets or Retrieves a Query that Lists the Names of User Objects

The QueryEnumerate method sets or retrieves a query that lists the names of user objects in the directory.

Syntax

The QueryEnumerate method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryEnumerate([queryEnumerate])
```

Parameters

The QueryEnumerate method accepts the following parameter:

queryEnumerate (string)

(Optional) Specifies the query that lists the names of user objects in the directory.

Return Value

The QueryEnumerate method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers):

```
select Name, 'Group' as Class from SmGroup order by Class
```

QueryGetGroupProp Method—Sets or Retrieves a Query that Fetches the Value of a Group Property

The QueryGetGroupProp method sets or retrieves a query that fetches the value of a group property. The property must be one of the properties specified through the QueryGetGroupProps method.

Syntax

The QueryGetGroupProp method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryGetGroupProp([queryGetGroupProp])
```

Parameters

The QueryGetGroupProp method accepts the following parameter:

queryGetGroupProp (string)

(Optional) Specifies the query that fetches the group property.

Return Value

The QueryGetGroupProp method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expressions are placeholders for property name and group name parameters to be supplied by SiteMinder when the query is executed:

```
select %s from SmGroup where Name = '%s'
```

QueryGetGroupProps Method—Sets or Retrieves a List of Group Properties

The QueryGetGroupProps method sets or retrieves a comma-separated list of group properties. These attributes are used to search the contents of a group, or to bind policies to group attributes. The attributes are expected to reside in the same table as the group name.

Syntax

The QueryGetGroupProps method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryGetGroupProps([queryGetGroupProps])
```

Parameters

The QueryGetGroupProps method accepts the following parameters:

queryGetGroupProps (string)

(Optional) Specifies the comma-separated list of group properties.

Return Value

The QueryGetGroupProps method returns one of the following values:

- The new or existing group properties list
- **undef** if the call is unsuccessful

Remarks

Sample list:

```
Name, GroupId
```

QueryGetGroups Method—Sets or Retrieves a Query that Fetches the Names of the Groups that the User Is a Member of

The QueryGetGroups method sets or retrieves a query that fetches the names of the groups that the user is a member of.

Syntax

The QueryGetGroups method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryGetGroups([queryGetGroups])
```

Parameters

The QueryGetGroups method accepts the following parameters:

queryGetGroups (string)

(Optional) Specifies the query that fetches the names of the user's groups.

Return Value

The QueryGetGroups method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expression is a placeholder for a user name parameter to be supplied by SiteMinder when the query is executed:

```
select SmGroup.Name from SmGroup, SmUser, SmUserGroup where SmUser.Name = '%s' and
SmUser.UserId = SmUserGroup.UserId and SmGroup.GroupId = SmUserGroup.GroupId
```

QueryGetObjInfo Method—Sets or Retrieves a Query that Fetches the Class of the Object

The QueryGetObjInfo method sets or retrieves a query that fetches the class of the object.

Syntax

The QueryGetObjInfo method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryGetObjInfo([queryGetObjInfo])
```

Parameters

The QueryGetObjInfo method accepts the following parameter:

queryGetObjInfo (string)

(Optional) Specifies the query that fetches the class of the object.

Return Value

The QueryGetObjInfo method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expression is a placeholder for a user or group object name to be supplied by SiteMinder when the query is executed:

```
select Name, 'User' from SmUser where Name = '%s' Union select Name, 'Group' from
SmGroup where Name = '%s'
```

QueryGetUserProp Method—Sets or Retrieves a Query that Fetches the Value of a User Property

The QueryGetUserProp method sets or retrieves a query that fetches the value of a user property. The property must be one of the properties specified through the PolicyMgtODBCQueryScheme->QueryGetUserProps method.

Syntax

The QueryGetUserProp method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryGetUserProp([queryGetUserProp])
```

Parameters

The QueryGetUserProp method accepts the following parameter:

queryGetUserProp (string)

(Optional) Specifies the query that fetches the user property.

Return Value

The QueryGetUserProp method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expressions are placeholders for property name and user name parameters to be supplied by SiteMinder when the query is executed:

```
select %s from SmUser where Name = '%s'
```

QueryGetUserProps Method—Sets or Retrieves a List of User Properties

The QueryGetUserProps method sets or retrieves a comma-separated list of user properties. The properties reside in the same table as the user name.

Syntax

The QueryGetUserProps method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryGetUserProps([queryGetUserProps])
```

Parameters

The QueryGetUserProps method accepts the following parameter:

queryGetUserProps (string)

(Optional) Specifies the comma-separated list of user properties.

Return Value

The QueryGetUserProps method returns one of the following values:

- The new or existing user properties list
- **undef** if the call is unsuccessful

Remarks

Sample list:

```
Name, UserId, FirstName, LastName, TelephoneNumber, EmailAddress, PIN, Mileage,
Disabled
```

QueryInitUser Method—Sets or Retrieves a Query that Determines whether a User Exists in the Database

The QueryInitUser method sets or retrieves a query that determines whether a particular user exists in the database.

Syntax

The QueryInitUser method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryInitUser([queryGetInitUser])
```

Parameters

The QueryInitUser method accepts the following parameter:

queryGetInitUser (string)

(Optional) Specifies the query that determines whether the user exists in the database.

Return Value

The QueryInitUser method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expression is a placeholder for the user name parameter to be supplied by SiteMinder when the query is executed:

```
select Name from SmUser where Name = '%s'
```

QueryIsGroupMember Method—Sets or Retrieves a Query that Lists the Group Membership for a Particular User

The QueryIsGroupMember method sets or retrieves a query that lists the group membership for a particular user.

Syntax

The QueryIsGroupMember method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryIsGroupMember([queryIsGroupMember])
```

Parameters

The QueryIsGroupMember method accepts the following parameters:

queryIsGroupMember (string)

(Optional) Specifies the query that determines a user's group membership.

Return Value

The QueryIsGroupMember method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expressions are placeholders for user name and group name parameters to be supplied by SiteMinder when the query is executed:

```
select Id from SmUserGroup where UserId = (select UserId from SmUser where Name = '%s')
and GroupId = (select GroupId from SmGroup where Name = '%s')
```

QueryLookup Method—Sets or Retrieves a Query that Fetches Objects

The QueryLookup method sets or retrieves a query that fetches objects based on a property specified in a group table.

Syntax

The QueryLookup method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryLookup([queryLookup])
```

Parameters

The QueryLookup method accepts the following parameter:

queryLookup (string)

(Optional) Specifies the query that fetches the objects.

Return Value

The QueryLookup method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expression is a placeholder for a parameter to be supplied by SiteMinder when the query is executed:

```
select Name, 'User' as Class from SmUser where Name %s Union select Name, 'Group' as
Class from SmGroup where Name %s order by Class
```

QueryLookupGroup Method—Sets or Retrieves a Query that Fetches a Group Name

The QueryLookupGroup method sets or retrieves a query that fetches a group name based on a property specified in a group table.

Syntax

The QueryLookupGroup method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryLookupGroup([queryLookupGrp])
```

Parameters

The QueryLookupGroup method accepts the following parameter:

queryLookupGrp (string)

(Optional) Specifies the query that fetches the group name.

Return Value

The QueryLookupGroup method returns one of the following values:

- A new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expression is a placeholder for a parameter to be supplied by SiteMinder when the query is executed:

```
select Name, 'Group' as Class from SmGroup where %s
```

QueryLookupUser Method—Sets or Retrieves a Query that Fetches a User Name

The QueryLookupUser method sets or retrieves a query that fetches a user name based on a property specified in the user table.

Syntax

The QueryLookupUser method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QueryLookupUser([queryLookupUsr])
```

Parameters

The QueryLookupUser method accepts the following parameter:

queryLookupUsr (string)

(Optional) Specifies the query that fetches the user name.

Return Value

The QueryLookupUser method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expression is a placeholder for a parameter to be supplied by SiteMinder when the query is executed:

```
select Name, 'User' as Class from SmUser where %s
```

QuerySetGroupProp Method—Sets or Retrieves a Query that Sets the Value of a Group Property

The QuerySetGroupProp method sets or retrieves a query that sets the value of a group property. The property must be one of the properties specified through the QueryGetGroupProps method.

Syntax

The QuerySetGroupProp method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QuerySetGroupProp([querySetGroupProp])
```

Parameters

The QuerySetGroupProp method accepts the following parameter:

querySetGroupProp (string)

(Optional) Specifies the query that sets the property value for the group.

Return Value

The QuerySetGroupProp method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expressions are placeholders for property name, property value, and group name parameters to be supplied by SiteMinder when the query is executed:

```
update SmGroup set %s = %s where Name = '%s'
```

QuerySetPassword Method—Sets or Retrieves a Query that Changes a User Password

The QuerySetPassword method sets or retrieves a query that changes a user password.

Syntax

The QuerySetPassword method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QuerySetPassword([querySetPassword])
```

Parameters

The QuerySetPassword method accepts the following parameter:

querySetPassword (string)

(Optional) Specifies the query that changes a user password.

Return Value

The QuerySetPassword method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expressions are placeholders for user password and user name parameters to be supplied by SiteMinder when the query is executed:

update SmUser set Password = '%s' where Name = '%s'

QuerySetUserProp Method—Sets or Retrieves a Query that Sets the Value of a User Property

The QuerySetUserProp method sets or retrieves a query that sets the value of a user property. The property must be one of the properties specified through the PolicyMgtODBCQueryScheme->QueryGetUserProps method.

Syntax

The QuerySetUserProp method has the following format:

```
Netegrity::PolicyMgtODBCQueryScheme->QuerySetUserProp([querySetUserProp])
```

Parameters

The QuerySetUserProp method accepts the following parameters:

querySetUserProp (string)

(Optional) Specifies the query that sets the property value for the group.

Return Value

The QuerySetUserProp method returns one of the following values:

- The new or existing query
- **undef** if the call is unsuccessful

Remarks

Sample query (based on the SiteMinder sample database schema SmSampleUsers). The %s expressions are placeholders for property name, property value, and user name parameters to be supplied by SiteMinder when the query is executed:

```
update SmUser set %s = %s where Name = '%s'
```

Password Policy Methods

The following methods act on PolicyMgtPwdPolicy objects:

- AllowNestedGroups Method—Allows the password policy to be configured for nested groups
- ApplyLowerPriorityPolicies Method—Sets or retrieves the flag for evaluating lower-priority password policies after the current policy is evaluated
- AuthLoginTrackFailure Method—Sets or retrieves the flag for allowing a user to log in if login tracking data fails to be written to the user directory
- BadLoginDisablementPeriod Method—Sets or retrieves the number of minutes after which a user is disabled after too many failed login attempts
- Description Method—Sets or retrieves the description of the password policy
- DictionaryMatch Method—Sets the minimum number of letters required to qualify a password for dictionary checking
- DictionaryPath Method—Sets or retrieves the location of a dictionary file that lists words that cannot be used in a password
- DisableAfterInactivityExpiration Method—Sets or retrieves the flag for disabling a user’s account if it has been inactive for a specified period
- DisableAfterPwdExpiration Method—Sets or retrieves the flag for disabling a user’s account after the user’s password expires
- EntireDir Method—Determines whether the password policy applies to the entire directory or just a part of it
- ExpirationDelay Method—Specifies the number of days a password can be used until it must be changed
- IsEnabled Method—Enables or disables a password policy
- MaxLoginFailures Method—Sets the maximum number of failed login attempts a user can make before the user account is disabled
- MaxLoginInactive Method—Sets the number of days of inactivity allowed before a user's password expires
- Name Method—Sets or retrieves the password policy name
- PwdAddRegExpMatch Method—Adds a regular expression that new passwords must match
- PwdAddRegExpNoMatch Method—Adds a regular expression that new passwords must not match
- PwdAllowDigits Method—Sets or retrieves the flag specifying whether passwords are allowed to have numeric characters

- PwdAllowLowercase Method—Sets or retrieves the flag specifying whether passwords are allowed to have lower case letters
- PwdAllowNonAlphaNum Method—Sets or retrieves the flag specifying whether passwords are allowed to have non-alphanumeric characters
- PwdAllowNonPrintable Method—Sets or retrieves the flag specifying whether passwords are allowed to have non-printable characters
- PwdAllowPunctuation Method—Sets or retrieves the flag specifying whether passwords are allowed to have punctuation mark characters
- PwdAllowUppercase Method—Sets or retrieves the flag specifying whether passwords are allowed to have uppercase letters
- PwdExpiryWarning Method—Sets the number of days in advance to notify the user that the password will expire
- PwdForceLowerCase Method—Sets or retrieves the flag for forcing a new password to lower case
- PwdForceUpperCase Method—Sets or retrieves the flag for forcing a new password to uppercase
- PwdGetAllRegExpMatch Method—Retrieves the tags of all the regular expressions that new passwords must match
- PwdGetAllRegExpNoMatch Method—Retrieves the tags of all the regular expressions that new passwords must not match
- PwdGetRegExp Method—Retrieves the regular expression for the specified tag
- PwdIgnoreSequence Method—Indicates whether to ignore sequence (that is, character position) when the different-from-previous-characters percentage is calculated
- PwdMaxLength Method—Sets or retrieves the maximum length for user passwords
- PwdMaxRepeatingChar Method—Sets or retrieves the maximum number of identical characters that can appear consecutively in a password
- PwdMinAlpha Method—Sets or retrieves the minimum number of alphabetic characters (A-Z, a-z) that a password must contain
- PwdMinAlphaNum Method—Sets or retrieves the minimum number of alphanumeric characters (A-Z, a-z, 0-9) that a password must contain
- PwdMinLength Method—Sets or retrieves the minimum number of alphanumeric characters (A-Z, a-z, 0-9) that a password must contain
- PwdMinLowercase Method—Sets or retrieves the minimum number of lower case letters that a password must contain
- PwdMinNonAlpha Method—Sets or retrieves the minimum number of non-alphanumeric characters that a password must contain

- PwdMinNonPrintable Method—Sets or retrieves the minimum number of non-printable characters that a password must contain
- PwdMinNumbers Method—Sets or retrieves the minimum number of numeric characters (0-9) that a password must contain
- PwdMinProfileMatch Method—Specifies the minimum character sequence to check against the user's personal information
- PwdMinPunctuation Method—Sets or retrieves the minimum number of punctuation marks that a password must contain
- PwdMinUppercase Method—Sets or retrieves the minimum number of uppercase letters that a password must contain
- PwdPercentDiff Method—Specifies the percentage of characters that a new password must contain that differ from characters in the previous password
- PwdPolicyPriority Method—Sets or retrieves the password’s priority setting (1-1000)
- PwdRedirectionURL Method—Sets or retrieves the URL where the user is re-directed when an invalid password is provided
- PwdRemoveRegExp Method—Removes the regular expression associated with the specified tag
- PwdReuseCount Method—Specifies the number of new passwords that must be used before an old password can be reused
- PwdReuseDelay Method—Specifies the number of days a user must wait before reusing a password
- ReEnableAfterIncorrectPwd Method—Specifies whether to re-enable a user account after the entry of an incorrect password
- Save Method—Saves modifications to a password policy
- StripEmbeddedWhitespace Method—Sets or retrieves the flag for stripping new passwords of embedded white space
- StripLeadingWhitespace Method—Sets or retrieves the flag for stripping new passwords of leading white space
- StripTrailingWhitespace Method—Sets or retrieves the flag for stripping new passwords of trailing white space
- TrackLoginDetails Method—Sets or retrieves the flag for tracking authentication attempts and successful logins

- UserDirClass Method—Sets or retrieves the directory class if the password policy applies to a part of the directory
- UserDirectory Method—Sets or retrieves the user directory for the password policy
- UserDirPath Method—Sets or retrieves the directory path if the password policy applies to a part of the directory

AllowNestedGroups Method—Allows the Password Policy To Be Configured for Nested Groups

The AllowNestedGroups method allows the password policy to be configured for nested groups. This method applies only to LDAP directories.

Syntax

The AllowNestedGroups method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->AllowNestedGroups([groupFlag])
```

Parameters

The AllowNestedGroups method accepts the following parameter:

groupFlag (int)

(Optional) Specifies whether to allow nested groups:

- 1 to allow nested groups
- 0 to disallow nested groups

Return Value

The AllowNestedGroups method returns one of the following values:

- 0 if nested groups are not allowed.
- 1 if nested groups are allowed.

AllowLowerPriorityPolicies Method—Sets Flag To Determine whether Password Policies with Lower Priority Should Be Evaluated

The ApplyLowerPriorityPolicies method sets or retrieves the flag that determines whether password policies with lower priority should be evaluated after the current password policy is evaluated.

Syntax

The ApplyLowerPriorityPolicies method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->ApplyLowerPriorityPolicies([lowerPriorityFlag])
```

Parameters

The ApplyLowerPriorityPolicies method accepts the following parameters:

lowerPriorityFlag (int)

(Optional) Specifies whether to enable evaluation of lower-priority password policies:

- 1 enables evaluation of lower-priority password policies
- 0 disables evaluation of lower-priority password policies

Return Value

The ApplyLowerPriorityPolicies method returns one of the following values:

- A new or existing flag setting
- **undef** if the call is unsuccessful

AuthLoginTrackFailure Method—Allows a User To Login if Login Tracking Data Fails

The AuthLoginTrackFailure method sets or retrieves the flag for allowing a user to log in if login tracking data fails to be written to the user directory. Login tracking data includes login attempts and successful logins.

Syntax

The AuthLoginTrackFailure method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->AuthLoginTrackFailure([trackingFlag])
```

Parameters

The AuthLoginTrackFailure method accepts the following parameter:

trackingFlag (int)

(Optional) Specifies whether to allow the user to login when login tracking fails:

- 1 allows the user to login
- 0 does not allow the user to login

Return Value

The AuthLoginTrackFailure method returns one of the following values:

- The new or existing flag setting
- **undef** if the call is unsuccessful

Remarks

If you enable this flag, users are allowed to log in even if login tracking data cannot be written to the user directory. If you disable this flag, users are not allowed to log in if login tracking data cannot be written to the user directory.

BadLoginDisablementPeriod Method—Sets or Retrieves the Number of Minutes Before a User Account Is Disabled

The BadLoginDisablementPeriod method sets or retrieves the number of minutes before a user account is disabled after too many failed login attempts.

Syntax

The BadLoginDisablementPeriod method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->BadLoginDisablementPeriod([disablementPeriod])
```

Parameters

The BadLoginDisablementPeriod method accepts the following parameters:

disablementPeriod (int)

(Optional) Specifies the number of minutes to allow before the user account is disabled.

Return Value

The BadLoginDisablementPeriod method returns one of the following values:

- The new or existing disablement period
- **undef** if the call is unsuccessful

Description Method—Sets or Retrieves the Description of the Password Policy

The Description method sets or retrieves the description of the password policy.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->Description([policyDesc])
```

Parameters

The Description method accepts the following parameter:

policyDesc (string)

(Optional) Specifies the description of the password policy.

Return Value

The Description method returns one of the following values:

- The new or existing policy description
- An empty string if the call is unsuccessful.

DictionaryMatch Method—Sets the Minimum Number of Letters Required To Qualify a Password for Dictionary Checking

The DictionaryMatch method sets the minimum number of letters required to qualify a password for dictionary checking.

Syntax

The DictionaryMatch method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->DictionaryMatch([dicMatchLen])
```

Parameters

The DictionaryMatch method accepts the following parameter:

dicMatchLen (int)

(Optional) Specifies the minimum number of letters required.

Return Value

The DictionaryMatch method returns one of the following values:

- A new or existing minimum setting
- **undef** if the call is unsuccessful

DictionaryPath Method—Sets or Retrieves the Location of a Dictionary File

The DictionaryPath method sets or retrieves the location of a dictionary file that lists words that cannot be used in a password.

Syntax

The DictionaryPath method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->DictionaryPath([dicPath])
```

Parameters

The DictionaryPath method accepts the following parameter:

dicPath (string)

(Optional) Specifies the new dictionary path.

Return Value

The DictionaryPath method returns one of the following values:

- The new or existing dictionary path.
- **undef** if the call is unsuccessful

Remarks

The dictionary file must be a text file located in a directory that all Policy Servers can access.

DisableAfterInactivityExpiration Method—Disables an Inactive User's Account

The DisableAfterInactivityExpiration method sets or retrieves the flag for disabling a user's account if it has been inactive for a specified period.

Syntax

The DisableAfterInactivityExpiration method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->DisableAfterInactivityExpiration([inactivityFlag])
```

Parameters

The DisableAfterInactivityExpiration method accepts the following parameters:

inactivityFlag (int)

(Optional) Specifies whether to disable the user's account

1 disables the user's account after a specified period of inactivity

0 keeps the account enabled and forces a password change

Return Value

The DisableAfterInactivityExpiration method returns one of the following values:

- The new or existing flag setting
- **undef** if the call is unsuccessful

Remarks

If the flag is set not to disable the user's account after the inactivity period, the user is required to change the password at the next login.

DisableAfterPwdExpiration Method—Disables a User's Aaccount after the User's Password Expires

The DisableAfterPwdExpiration method sets or retrieves the flag for disabling a user's account after the user's password expires.

Syntax

The DisableAfterPwdExpiration method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->DisableAfterPwdExpiration([expireFlag])
```

Parameters

The DisableAfterPwdExpiration method accepts the following parameter:

expireFlag (type)

(Optional) Specifies whether to disable the user's account:

1 disable the user's account after the user's password expires

0 keeps the account enabled and forces a password change

Return Value

The DisableAfterPwdExpiration method returns one of the following values:

- The new or existing flag setting
- **undef** if the call is unsuccessful

Remarks

If the flag is set not to disable the user's account after the password expires, the user is required to change the password at next login.

EntireDir Method—Determines Whether the Password Policy Applies to the Entire Directory

The EntireDir method determines whether the password policy applies to the entire directory or just a part of it.

Syntax

The EntireDir method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->EntireDir([dirFlag])
```

Parameters

The EntireDir method accepts the following parameters:

dirFlag (int)

(Optional) Specifies whether to apply the password policy to an entire directory:

1 applies the password policy to the entire directory

0 applies the password policy to just a portion of the directory

Return Value

The EntireDir method returns one of the following values:

- 1 if the policy applies to the entire directory.
- 0 if the policy applies to part of the directory.

Remarks

For information about specifying a part of an entire directory, see the descriptions of the PolicyMgtPwdPolicy->UserDirPath (see page 339) method and the PolicyMgtPwdPolicy->UserDirClass (see page 338) method.

ExpirationDelay Method—Specifies the Number of Days a Password Can Be Used

The ExpirationDelay method specifies the number of days a password can be used until it must be changed.

Syntax

The ExpirationDelay method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->ExpirationDelay([expDelay])
```

Parameters

The ExpirationDelay method accepts the following parameter:

expDelay (int)

(Optional) Specifies the number of days that the password can be used.

Return Value

The ExpirationDelay method returns one of the following values:

- The new or existing number of days
- -1 if the call is unsuccessful

IsEnabled Method—Enables or Disables a Password Policy

The IsEnabled method enables or disables a password policy.

Syntax

The IsEnabled method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->IsEnabled([enableFlag])
```

Parameters

The IsEnabled method accepts the following parameter:

enableFlag (int)

(Optional) Specifies whether the password policy is enabled:

- 1 enables the password policy
- 0 disables the password policy

Return Value

The IsEnabled method returns one of the following values:

- 1 if the policy is enabled
- 0 if the policy is disabled

MaxLoginFailures Method—Sets or Retrieves the Maximum Number of Failed Login Attempts

The MaxLoginFailures method sets or retrieves the maximum number of failed login attempts a user can make before the user account is disabled.

Syntax

The MaxLoginFailures method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->MaxLoginFailures([maxLogin])
```

Parameters

The MaxLoginFailures method accepts the following parameter:

maxLogin (int)

(Optional) Specifies the number of failed login attempts.

Return Value

The MaxLoginFailures method returns one of the following values:

- The new or existing failed login attempt setting
- **undef** if the call is unsuccessful

MaxLoginInactive Method—Sets or Retrieves the Number of Days of Inactivity Are Allowed

The MaxLoginInactive method sets or retrieves the number of days of inactivity allowed before a user's password expires.

Syntax

The MaxLoginInactive method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->MaxLoginInactive([maxLoginInactive])
```

Parameters

The MaxLoginInactive method accepts the following parameters:

maxLoginInactive (int)

(Optional) Specifies the number of days of inactivity.

Return Value

The MaxLoginInactive method returns one of the following values:

- The new or existing maximum inactivity period setting
- **undef** if the call is unsuccessful

Name Method—Sets or Retrieves the Password Policy Name

The Name method sets or retrieves the password policy name.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->Name([policyName])
```

Parameters

The Name method accepts the following parameter:

policyName (string)

(Optional) Specifies the password policy name.

Return Value

The Name method returns one of the following values:

- The new or existing policy name
- **undef** if the call is unsuccessful

PwdAddRegExpMatch Method—Adds a Regular Expression to the List of Expressions that New Passwords Must Match

The PwdAddRegExpMatch method adds a regular expression to the list of expressions that new passwords must match.

Syntax

The PwdAddRegExpMatch method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdAddRegExpMatch([tag] [, expression])
```

Parameters

The PwdAddRegExpMatch method accepts the following parameters:

tag (string)

(Optional) Specifies the name of the regular expression.

expression (string)

(Optional) Specifies the regular expression.

Return Value

The PwdAddRegExpMatch method returns one of the following values:

- 0 if the regular expression is successfully added
- -1 if the call is unsuccessful

PwdAddRegExpNoMatch Method—Adds a Regular Expression to the List of Expressions that New Passwords Must NOT Match

The PwdAddRegExpNoMatch method adds a regular expression to the list of expressions that new passwords must *not* match.

Syntax

The PwdAddRegExpNoMatch method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdAddRegExpNoMatch([tag] [, expression])
```

Parameters

The PwdAddRegExpNoMatch method accepts the following parameters:

tag (string)

(Optional) Specifies the name of the regular expression.

expression (string)

(Optional) Specifies the regular expression.

Return Value

The PwdAddRegExpNoMatch method returns one of the following values:

- 0 if the regular expression is successfully added
- -1 if the call is unsuccessful

PwdAllowDigits Method—Specifies whether Passwords Are Allowed To Have Numeric Characters

The PwdAllowDigits method sets or retrieves the flag that specifies whether passwords are allowed to have numeric characters.

Syntax

The PwdAllowDigits method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdAllowDigits([digitFlag])
```

Parameters

The PwdAllowDigits method accepts the following parameter:

digitFlag (int)

(Optional) Specifies whether passwords are allowed to have numeric characters:

1 numeric characters are allowed

0 if numeric characters are not allowed

Return Value

The PwdAllowDigits method returns one of the following values:

- A new or existing flag setting
- **undef** if the call is unsuccessful

PwdAllowLowercase Method—Specifies whether Passwords Are Allowed To Have Lower Case Letters

The PwdAllowLowercase method sets or retrieves the flag that specifies whether passwords are allowed to have lower case letters.

Syntax

The PwdAllowLowercase method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdAllowLowercase([lcFlag])
```

Parameters

The PwdAllowLowercase method accepts the following parameters:

lcFlag (int)

(Optional) Specifies whether lowercase letters are allowed in passwords:

- 1 allows lowercase letters
- 0 disallows lowercase letters

Return Value

The PwdAllowLowercase method returns one of the following values:

- The new or existing flag setting
- **undef** if the call is unsuccessful

PwdAllowNonAlphNum Method—Specifies whether Passwords Are Allowed To Have Non-Alphanumeric Characters

The PwdAllowNonAlphNum method sets or retrieves the flag that specifies whether passwords are allowed to have non-alphanumeric characters.

Syntax

The PwdAllowNonAlphNum method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdAllowNonAlphaNum([nonAlphaNumFlag])
```

Parameters

The PwdAllowNonAlphNum method accepts the following parameters:

nonAlphaNumFlag (int)

(Optional) Specifies whether non-alphanumeric characters are allowed in passwords

- 1 allows non-alphanumeric characters
- 0 disallows non-alphanumeric characters

Return Value

The PwdAllowNonAlphNum method returns one of the following values:

- The new or existing flag setting
- **undef** if the call is unsuccessful

PwdAllowNonPrintable Method—Specifies whether Passwords Are Allowed To Have Non-Printable Characters

The PwdAllowNonPrintable method sets or retrieves the flag that specifies whether passwords are allowed to have non-printable characters. These characters cannot be displayed on a computer screen.

Syntax

The PwdAllowNonPrintable method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdAllowNonPrintable([nonPrintFlag])
```

Parameters

The PwdAllowNonPrintable method accepts the following parameters:

nonPrintFlag (int)

(Optional) Specifies whether non-printable characters are allowed in passwords:

- 1 allows non-printable characters
- 0 disallows non-printable characters

Return Value

The PwdAllowNonPrintable method returns one of the following values:

- The new or existing flag setting
- **undef** if the call is unsuccessful

PwdAllowPunctuation Method—Specifies whether Passwords Are Allowed To Have Punctuation Mark Characters

The PwdAllowPunctuation method sets or retrieves the flag that specifies whether passwords are allowed to have punctuation mark characters.

Syntax

The PwdAllowPunctuation method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdAllowPunctuation([punctuationMarkFlag])
```

Parameters

The PwdAllowPunctuation method accepts the following parameters:

punctuationMarkFlag (int)

(Optional) Specifies whether punctuation mark characters are allowed in passwords:

- 1 allows punctuation mark characters
- 0 disallows punctuation mark characters

Return Value

The PwdAllowPunctuation method returns one of the following values:

- The new or existing flag setting
- **undef** if the call is unsuccessful

PwdAllowUpperCase Method—Specifies whether Passwords Are Allowed To Have Upper Case Letters

The PwdAllowUpperCase method sets or retrieves the flag that specifies whether passwords are allowed to have upper case letters.

Syntax

The PwdAllowUpperCase method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdAllowUppercase([upperCaseFlag])
```

Parameters

The PwdAllowUpperCase method accepts the following parameter:

upperCaseFlag (int)

(Optional) Specifies whether upper case letters are allowed in passwords:

- 1 allows upper case letters
- 0 disallows upper case letters

Return Value

The PwdAllowUpperCase method returns one of the following values:

- The new or existing flag setting
- **undef** if the call is unsuccessful

PwdExpiryWarning Method—Sets or Retrieves the Number of Days in Advance To Notify the User that the Password Will Expire

The PwdExpiryWarning method sets or retrieves the number of days in advance to notify the user that the password will expire.

Syntax

The PwdExpiryWarning method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdExpiryWarning([warningDays])
```

Parameters

The PwdExpiryWarning method accepts the following parameters:

warningDays (int)

(Optional) Specifies the number of days of advance notice.

Return Value

The PwdExpiryWarning method returns one of the following values:

- The new or existing advance notice setting
- **undef** if the call is unsuccessful

PwdForceLowerCase Method—Determines whether To Convert Upper Case Letters in a New Password to Lower Case

The PwdForceLowerCase method sets or retrieves the flag that determines whether to convert any upper case letters in a new password to lower case.

Syntax

The PwdForceLowerCase method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdForceLowerCase([forceLCFlag])
```

Parameters

The PwdForceLowerCase method accepts the following parameters:

forceLCFlag (int)

(Optional) Specifies whether for force new passwords into lower vase:

- 1 converts any upper case letters to lower case
- 0 does not convert upper case letters

Return Value

The PwdForceLowerCase method returns one of the following values:

- The new or existing flag setting
- **undef** if the call is unsuccessful

PwdForceUpperCase Method—Determines whether To Convert Lower Case Letters in a New Password to Upper Case

The PwdForceUpperCase method sets or retrieves the flag that determines whether to convert any lower case letters in a new password to upper case.

Syntax

The PwdForceUpperCase method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdForceUpperCase([forceUCFlag])
```

Parameters

The PwdForceUpperCase method accepts the following parameters:

forceUCFlag (int)

(Optional) Specifies whether to force new passwords to use only upper case:

- 1 forces upper case
- 0 does not force upper case

Return Value

The PwdForceUpperCase method returns one of the following values:

- The new or existing flag setting
- **undef** if the call is unsuccessful

PwdGetAllRegExpMatch Method—Retrieves the Name Tags of the Regular Expressions that New Passwords Must Match

The PwdGetAllRegExpMatch method retrieves the name tags of all the regular expressions that new passwords must match.

Syntax

The PwdGetAllRegExpMatch method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdGetAllRegExpMatch()
```

Parameters

The PwdGetAllRegExpMatch method accepts no parameters.

Return Value

The PwdGetAllRegExpMatch method returns one of the following values:

- An array of name tags for the regular expressions that new passwords must match
- **undef** if the call is unsuccessful

PwdGetAllRegExpNoMatch Method—Retrieves the Name Tags of the Regular Expressions that New Passwords Must NOT Match

The PwdGetAllRegExpNoMatch method retrieves the name tags of all the regular expressions that new passwords must *not* match.

Syntax

The PwdGetAllRegExpNoMatch method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdGetAllRegExpNoMatch()
```

Parameters

The PwdGetAllRegExpNoMatch method accepts no parameters.

Return Value

The PwdGetAllRegExpNoMatch method returns one of the following values:

- An array of name tags for the regular expressions that new passwords must not match.
- **undef** if the call is unsuccessful

PwdGetRegExp Method—Retrieves the Regular Expression for the Specified Name Tag

The PwdGetRegExp method retrieves the regular expression for the specified name tag.

Syntax

The PwdGetRegExp method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdGetRegExp(tag)
```

Parameters

The PwdGetRegExp method accepts the following parameter:

tag (string)

Specifies the name of the regular expression to retrieve.

Return Value

The PwdGetRegExp method returns one of the following values:

- The specified regular expression
- **undef** if the call is unsuccessful

PwdIgnoreSequence Method—Determines whether To Ignore Sequence when Calculating the New Password

The PwdIgnoreSequence method specifies whether to ignore sequence (that is, character position) when the different-from-previous-characters percentage is calculated.

Syntax

The PwdIgnoreSequence method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdIgnoreSequence([pwdPctSeq])
```

Parameters

The PwdIgnoreSequence method accepts the following parameter:

pwdPctSeq (int)

(Optional) Specifies whether to ignore the sequence of characters when creating a new password:

- 1 ignores sequence when calculating the previous password difference percentage
- 0 considers sequence

Return Value

The PwdIgnoreSequence method returns one of the following values:

- 1 to ignore sequence
- 0 to consider sequence

Remarks

For example, suppose a user's previous password is BASEBALL12:

- If you set this method to 1 (ignore sequence), the user can't choose 12BASEBALL as the new password. That's because the characters are the same as in the previous password, regardless of the character sequence.
- If you set this method to 0 (consider sequence), the user can choose 12BASEBALL as the new password because the characters occur in a different sequence.

For greater security, pass 1 into this method.

PwdMaxLength Method—Sets or Retrieves the Maximum Length for User Passwords

The PwdMaxLength method sets or retrieves the maximum length for user passwords.

Syntax

The PwdMaxLength method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMaxLength([maxPwdLength])
```

Parameters

The PwdMaxLength method accepts the following parameter:

maxPwdLength (int)

(Optional) Specifies the maximum password length.

Return Value

The PwdMaxLength method returns the new or existing password length setting.

PwdMaxRepeatingChar Method—Sets or Retrieves the Maximum Number of Identical Characters

The PwdMaxRepeatingChar method sets or retrieves the maximum number of identical characters that can appear consecutively in a password.

Syntax

The PwdMaxRepeatingChar method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMaxRepeatingChar([maxPwdRepeat])
```

Parameters

The PwdMaxRepeatingChar method accepts the following parameter:

maxPwdRepeat (int)

(Optional) Specifies the maximum number of repeating characters.

Return Value

The PwdMaxRepeatingChar method returns the new or existing setting for repeating characters.

PwdMinAlpha Method—Sets or Retrieves the Minimum Number of Alphabetic Characters a Password Must Contain

The PwdMinAlpha method sets or retrieves the minimum number of alphabetic characters (A-Z, a-z) that a password must contain.

Syntax

The PwdMinAlpha method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMinAlpha([pwdMinAlpha])
```

Parameters

The PwdMinAlpha method accepts the following parameter:

pwdMinAlpha (int)

(Optional) Specifies the minimum number of alphabetic characters required.

Return Value

The PwdMinAlpha method returns the new or existing minimum number of alphabetic characters.

PwdMinAlphaNum Method—Sets or Retrieves the Minimum Number of Alphanumeric Characters a Password Must Contain

The PwdMinAlphaNum method sets or retrieves the minimum number of alphanumeric characters (A-Z, a-z, 0-9) that a password must contain.

Syntax

The PwdMinAlphaNum method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMinAlphaNum([pwdMinAlphaNum])
```

Parameters

The PwdMinAlphaNum method accepts the following parameters:

pwdMinAlphaNum (int)

(Optional) Specifies the minimum number of alphanumeric characters required.

Return Value

The PwdMinAlphaNum method returns the new or existing minimum number of alphanumeric characters.

PwdMinLength Method—Sets or Retrieves the Minimum Length for User Passwords

The PwdMinLength method sets or retrieves the minimum length for user passwords.

Syntax

The PwdMinLength method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMinLength([minPwdLength])
```

Parameters

The PwdMinLength method accepts the following parameters:

minPwdLength (int)

(Optional) Specifies the minimum length for user passwords.

Return Value

The PwdMinLength method returns the new or existing minimum password length.

PwdMinLowercase Method—Sets or Retrieves the Minimum Number of Lower Case Letters a Password Must Contain

The PwdMinLowercase method sets or retrieves the minimum number of lower case letters that a password must contain.

Syntax

The PwdMinLowercase method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMinLowercase([pwdMinLC])
```

Parameters

The PwdMinLowercase method accepts the following parameter:

pwdMinLC (int)

(Optional) Specifies the minimum number of lower case letters that a password must contain.

Return Value

The PwdMinLowercase method returns new or existing minimum for lower case letters.

PwdMinNonAlpha Method—Sets or Retrieves the Minimum Number of Non-Alphanumeric Characters A Password Must Contain

The PwdMinNonAlpha method sets or retrieves the minimum number of non-alphanumeric characters that a password must contain. These characters include punctuation marks and other symbols located on the keyboard, such as @, $, and *.

Syntax

The PwdMinNonAlpha method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMinNonAlpha([pwdMinNonAlpha])
```

Parameters

The PwdMinNonAlpha method accepts the following parameters:

pwdMinNonAlpha (int)

(Optional) Specifies the minimum number of non-alphanumeric characters required.

Return Value

The PwdMinNonAlpha method returns the new or existing minimum number of non-alphanumeric characters.

PwdMinNonPrintable Method—Sets or Retrieves the Minimum Number of Non-Printable Characters a Password Must Contain

The PwdMinNonPrintable method sets or retrieves the minimum number of non-printable characters that a password must contain. These characters cannot be displayed on a computer screen.

Syntax

The PwdMinNonPrintable method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMinNonPrintable([pwdMinNonPrint])
```

Parameters

The PwdMinNonPrintable method accepts the following parameter:

pwdMinNonPrint (int)

(Optional) Specifies the minimum number of non-printable characters required.

Return Value

The PwdMinNonPrintable method returns The new or existing minimum number of non-printable characters.

PwdMinNumbers Method—Sets or Retrieves the Minimum Number of Numeric Characters a Password Must Contain

The PwdMinNumbers method sets or retrieves the minimum number of numeric characters (0-9) that a password must contain.

Syntax

The PwdMinNumbers method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMinNumbers([pwdMinNum])
```

Parameters

The PwdMinNumbers method accepts the following parameter:

pwdMinNum (int)

(Optional) Specifies the minimum number of numeric characters required.

Return Value

The PwdMinNumbers method returns the new or existing minimum number of numeric characters.

PwdMinProfileMatch Method—Specifies the Minimum Character Sequence To Check against the User's Personal Information

The PwdMinProfileMatch method specifies the minimum character sequence to check against the user's personal information.

Syntax

The PwdMinProfileMatch method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMinProfileMatch([pwdMatchAttr])
```

Parameters

The PwdMinProfileMatch method accepts the following parameter:

pwdMatchAttr (int)

(Optional) Specifies the minimum number of sequential characters to check.

Return Value

The PwdMinProfileMatch method returns the new or existing minimum setting.

Remarks

For example, if this value is set to 4, SiteMinder prohibits the use of any four consecutive characters found in the user's personal information, such as the four last digits of the user's telephone number.

This field prevents a user from incorporating personal information in a password. SiteMinder checks the password against attributes in the user's directory entry.

PwdMinPunctuation Method—Sets or Retrieves the Minimum Number of Punctuation Marks a Password Must Contain

The PwdMinPunctuation method sets or retrieves the minimum number of punctuation marks that a password must contain. These characters include periods, commas, exclamation marks, slashes, hyphens, dashes, and other punctuation marks.

Syntax

The PwdMinPunctuation method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMinPunctuation([pwdMinPunc])
```

Parameters

The PwdMinPunctuation method accepts the following parameter:

pwdMinPunc (int)

(Optional) Specifies the minimum number of punctuation marks required.

Return Value

The PwdMinPunctuation method returns the new or existing minimum number of punctuation marks.

PwdMinUppercase Method—Sets or Retrieves the Minimum Number of Upper Case Letters a Password Must Contain

The PwdMinUppercase method sets or retrieves the minimum number of upper case letters that a password must contain.

Syntax

The PwdMinUppercase method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdMinUppercase([pwdMinUC])
```

Parameters

The PwdMinUppercase method accepts the following parameter:

pwdMinUC (int)

(Optional) Specifies the minimum number of upper case letters that a password must contain.

Return Value

The PwdMinUppercase method returns the new or existing minimum for upper case letters.

PwdPercentDiff Method—Sets or Retrieves the Percentage of Different Characters a New Password Must Contain

The PwdPercentDiff method sets or retrieves the percentage of characters that a new password must contain that differ from characters in the previous password. If the value is set to 100, the new password cannot contain any characters that were in the previous password (unless the parameter *PwdIgnoreSeq* is set to 0).

Syntax

The PwdPercentDiff method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdPercentDiff([pwdPctDiff])
```

Parameters

The PwdPercentDiff method accepts the following parameter:

pwdPctDiff (int)

(Optional) Specifies the minimum percentage setting.

Return Value

The PwdPercentDiff method returns the new or existing minimum percentage setting.

PwdPolicyPriority Method—Sets or Retrieves the Password's Evaluation Priority Setting

The PwdPolicyPriority method sets or retrieves the password's evaluation priority setting (1-1000). Policies are evaluated in descending order (1000 first, 1 last).

Syntax

The PwdPolicyPriority method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdPolicyPriority([priority])
```

Parameters

The PwdPolicyPriority method accepts the following parameters:

priority (int)

(Optional) Specifies the evaluation priority of this password policy.

Return Value

The PwdPolicyPriority method returns new or existing evaluation priority setting.

PwdRedirectionURL Method—Sets or Retrieves the URL where the User is Redirected Example

The PwdRedirectionURL method sets or retrieves the URL where the user is redirected when an invalid password is provided. This must be the URL of the Password Services CGI.

Syntax

The PwdRedirectionURL method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdRedirectionURL([URL])
```

Parameters

The PwdRedirectionURL method accepts the following parameter:

URL (string)

(Optional) Specifies the redirection URL.

Return Value

The PwdRedirectionURL method returns one of the following values:

- The new or existing URL
- **undef** if the call is unsuccessful

PwdRemoveRegExp Method—Removes the Regular Expression Associated with the Specified Name Tag

The PwdRemoveRegExp method removes the regular expression associated with the specified name tag.

Syntax

The PwdRemoveRegExp method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdRemoveRegExp(tag)
```

Parameters

The method accepts the following parameter:

tag (string)

Specifies the name of the regular expression to move.

Return Value

The PwdRemoveRegExp method returns one of the following values:

- 0 on success
- -1 if the call is unsuccessful

PwdReuseCount Method—Specifies the Number of New Passwords that Must Be Used

The PwdReuseCount method specifies the number of new passwords that must be used before an old password can be reused.

Syntax

The PwdReuseCount method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdReuseCount([pwdReuseCount])
```

Parameters

The PwdReuseCount method accepts the following parameters:

pwdReuseCount (int)

(Optional) Specifies the password reuse setting.

Return Value

The PwdReuseCount method returns the new or existing password reuse setting.

PwdReuseDelay Method—Specifies the Number of Days a User Must Wait Before Reusing a Password

The PwdReuseDelay method specifies the number of days a user must wait before reusing a password.

Syntax

The PwdReuseDelay method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->PwdReuseDelay([pwdReuseDelay])
```

Parameters

The PwdReuseDelay method accepts the following parameter:

pwdReuseDelay (type)

(Optional) Specifies the password reuse delay setting.

Return Value

The PwdReuseDelay method returns the new or existing password reuse delay setting.

ReEnableAfterIncorrectPwd Method—Determines whether To Re-enable a User Account after the Entry of an Incorrect Password

The ReEnableAfterIncorrectPwd method determines whether to re-enable a user account after the entry of an incorrect password or passwords.

Syntax

The ReEnableAfterIncorrectPwd method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->ReEnableAfterIncorrectPwd([groupFlag])
```

Parameters

The ReEnableAfterIncorrectPwd method accepts the following parameter:

groupFlag (int)

(Optional) Specifies whether to re-enable a user account after the entry of an incorrect password:

- 0 disables the account
- 1 enables the account

Return Value

The ReEnableAfterIncorrectPwd method returns one of the following values:

- 1 if a user account should be re-enabled after entry of an incorrect password or passwords.
- 0 if a user should be allowed 1 login attempt after entry of an incorrect password or passwords.

Save Method—Saves the Password Policy to the Policy Store

The Save method saves the password policy to the policy store.

Syntax

The Save method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->Save( )
```

Parameters

The Save method accepts no parameters.

Return Value

The Save method returns one of the following values:

- 0 if the call is successful
- -1 if the call is unsuccessful
- -4 if the user has insufficient privileges to save the changes.
- 10 if the path and class are empty.

Remarks

Call this method once after making all the modifications to the password policy that you intend to make. This method must be called for any changes to take effect.

StripEmbeddedWhitespace Method—Determines whether To Strip New Passwords of Embedded White Space

The StripEmbeddedWhitespace method sets or retrieves the flag that determines whether to strip new passwords of embedded white space.

Syntax

The StripEmbeddedWhitespace method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->StripEmbeddedWhitespace([stripEmbeddedFlag])
```

Parameters

The StripEmbeddedWhitespace method accepts the following parameter:

stripEmbeddedFlag (int)

(Optional) Specifies whether to strip embedded white space from new passwords:

- 1 strips the embedded white space
- 0 includes embedded white space

Return Value

The StripEmbeddedWhitespace method returns the new or existing flag setting.

StripLeadingWhitespace Method—Determines whether To Strip New Passwords of Leading White Space

The StripLeadingWhitespace method sets or retrieves the flag that determines whether to strip new passwords of leading white space.

Syntax

The StripLeadingWhitespace method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->StripLeadingWhitespace([stripLeadingFlag])
```

Parameters

The StripLeadingWhitespace method accepts the following parameter:

stripLeadingFlag (int)

(Optional) Specifies whether to strip leading white space from passwords:

- 1 strips leading white space
- 0 includes leading white space

Return Value

The StripLeadingWhitespace method returns the new or existing flag setting.

StripTrailingWhitespace Method—Determines whether To Strip New Passwords of Trailing White Space

The StripTrailingWhitespace method sets or retrieves the flag that determines whether to strip new passwords of trailing white space.

Syntax

The StripTrailingWhitespace method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->StripTrailingWhitespace([stripTrailingFlag])
```

Parameters

The StripTrailingWhitespace method accepts the following parameter:

stripTrailingFlag (int)

(Optional) Specifies whether to strip trailing white space from passwords:

- 1 strips trailing white space
- 0 includes trailing white space

Return Value

The StripTrailingWhitespace method returns the new or existing flag setting.

TrackLoginDetails Method—Determines whether To Track Authentication Attempts and Successful Logins

The TrackLoginDetails method sets or retrieves the flag that determines whether to track authentication attempts and successful logins.

Syntax

The TrackLoginDetails method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->TrackLoginDetails([trackingFlag])
```

Parameters

The TrackLoginDetails method accepts the following parameter:

trackingFlag (int)

(Optional) Specifies whether to enable login tracking:

- 1 enables login tracking
- 0 disables login tracking

Return Value

The TrackLoginDetails method returns the new or existing flag setting.

UserDirClass Method—Sets or Retrieves the Directory Class if the Password Policy Applies to a Part of the Directory

The UserDirClass method sets or retrieves the directory class if the password policy applies to a part of the directory.

Syntax

The UserDirClass method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->UserDirClass([path])
```

Parameters

The UserDirClass method accepts the following parameter:

path (string)

(Optional) Specifies the directory class.

Return Value

The UserDirClass method returns the new or existing directory class.

UserDirectory Method—Sets or Retrieves the User Directory for the Password Policy

The UserDirectory method sets or retrieves the user directory for the password policy.

Syntax

The UserDirectory method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->UserDirectory([userDir])
```

Parameters

The UserDirectory method accepts the following parameter:

userDir (PolicyMgtUserDir)

(Optional) Specifies the user directory for the password policy.

Return Value

The UserDirectory method returns a PolicyMgtUserDir object.

UserDirPath Method—Sets or Retrieves the Directory Path if the Password Policy Applies to a Part of the Directory

The UserDirPath method sets or retrieves the directory path if the password policy applies to a part of the directory.

Syntax

The UserDirPath method has the following format:

```
Netegrity::PolicyMgtPwdPolicy->UserDirPath([path])
```

Parameters

The UserDirPath method accepts the following parameter:

path (type)

(Optional) Specifies the directory path.

Return Value

The UserDirPath method returns the new or existing directory path.

Policy Methods

The following methods act on PolicyMgtPolicy objects:

- ActiveExpr Method—Sets or retrieves the active expression associated with the policy.
- AddRule Method—Adds a rule to the policy
- AddUser Method—Adds a user to the policy
- AllowNested Method—Sets or removes the AllowNested flag based on the value of flag specifying recursive evaluation
- CreateIPConfigHostName Method—Creates an IP configuration object from the specified host name
- CreateIPConfigRange Method—Creates an IP configuration object from the specified range of IP addresses
- CreateIPConfigSingleHost Method—Creates an IP configuration object from the specified IP address
- CreateIPConfigSubnetMask Method—Creates an IP Address configuration based on the IP address and subnet mask passed to the method
- DeleteIPConfig Method—Deletes an IP configuration object
- Description Method—Sets or retrieves the description of the policy
- EnforceANDEvaluation Method—Sets or removes the ANDUser/Group flag
- ExcludeUser Method—Excludes or includes a user
- GetAllIPConfigs Method—Retrieves all IP configuration objects in the policy
- GetAllRules Method—Retrieves an array of all rules associated with the policy
- GetAllUsers Method—Retrieves an array of all users associated with the policy
- IsEnabled Method—Enables or disables the policy
- Name Method—Sets or retrieves the policy name

- RemoveResponse Method—Removes the response for a configured rule in the policy
- RemoveRule Method—Removes the specified rule from the policy
- RemoveUser Method—Removes a user from the policy
- SetResponse Method—Sets the response for a configured rule in the policy
- VariableExpr Method—Sets, retrieves, or removes the active expression associated with the policy

ActiveExpr Method—Sets or Retrieves the Active Expression Associated with the Policy

The ActiveExpr method sets or retrieves the active expression associated with the policy.

Syntax

The ActiveExpr method has the following format:

```
Netegrity::PolicyMgtPolicy->ActiveExpr([activeExpr])
```

Parameters

The ActiveExpr method accepts the following parameter:

activeExpr (string)

(Optional) Specifies the active expression to set.

Return Value

The ActiveExpr method returns one of the following values:

- The new or existing active expression
- **undef** if the call is unsuccessful

AddRule Method—Adds a Rule to the Policy

The AddRule method adds a rule to the policy.

Syntax

The AddRule method has the following format:

```
Netegrity::PolicyMgtPolicy->AddRule(rule)
```

Parameters

The AddRule method accepts the following parameter:

rule (PolicyMgtRule)

Specifies the rule to add.

Return Value

The AddRule method returns one of the following values:

- 0 if the call is successful
- -1 if the call is unsuccessful

AddUser Method—Adds a User to the Policy

The AddUser method adds a user to the policy.

Syntax

The AddUser method has the following format:

```
Netegrity::PolicyMgtPolicy->AddUser(user [, iExcludeUser] [, iRecursiveFlag] [,
iANDUserFlag])
```

Parameters

The AddUser method accepts the following parameters:

user (PolicyMgtUser)

Specifies the user to add.

iExcludeUser (int)

(Optional) Specifies whether to exclude a user:

- 1 excludes the user
- 0 includes the user

iRecursiveFlag (int)

(Optional) Specifies the setting for the AllowNested flag:

- 1 sets the AllowNested flag
- 0 disables the AllowNested flag

iANDUserFlag (int)

(Optional) Specifies the setting for the AND flag:

1 set the AND flag

0 disables the AND flag

Return Value

The AddUser method returns one of the following values:

- 0 if the call is successful
- -1 if the call is unsuccessful

AllowNested Method—Sets or Retrieves the AllowNested Flag

The AllowNested method sets or retrieves the AllowNested flag.

Syntax

The AllowNested method has the following format:

```
Netegrity::PolicyMgtPolicy->AllowNested(user[, iRecursiveFlag])
```

Parameters

The AllowNested method accepts the following parameters:

user (PolicyMgtUser)

Specifies the user for which to set or retrieve the AllowNested flag.

iRecursiveFlag (int)

(Optional) Specifies the value of the *AllowNested* flag:

- 1 for recursive evaluation
- 0 for non-recursive evaluation

If this is not passed, the function returns the current value of the *AllowNested* flag. The flag applies to all the users added to the policy for a particular user directory.

Return Value

The AllowNested method returns one of the following values:

- 0 if AllowNested flag is removed successfully.
- 1 if AllowNested flag is set successfully.
- -1 if the call is unsuccessful

CreateIPHostConfigName Method—Creates an IP Address Configuration

The CreateIPConfigHostName method creates an IP Address configuration based on the host name passed to the method. For the policy to fire, a request must come from the machine with the passed host name.

Syntax

The CreateIPConfigHostName method has the following format:

```
Netegrity::PolicyMgtPolicy->CreateIPConfigHostName(hostName)
```

Parameters

The CreateIPConfigHostName method accepts the following parameter:

hostName (string)

Specifies the host name required for the policy to fire.

Return Value

The CreateIPConfigHostName method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** if the call is unsuccessful

CreateIPConfigRange Method—Creates an IP Address Configuration

The CreateIPConfigRange method creates an IP Address configuration based on the range of IP addresses passed to the method. For the policy to fire, a request must come from a machine with an IP address that falls within the range.

Syntax

The CreateIPConfigRange method has the following format:

```
Netegrity::PolicyMgtPolicy->CreateIPConfigRange(ipAddr1, ipAddr2)
```

Parameters

The CreateIPConfigRange method accepts the following parameters:

ipAddr1 (string)

Specifies the beginning IP address in the range of accepted addresses.

ipAddr2 (string)

Specifies the ending IP address in the range of accepted addresses.

Return Value

The CreateIPConfigRange method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** if the call is unsuccessful

CreateIPConfigSingleHost Method—Creates an IP Address Configuration

The CreateIPConfigSingleHost method creates an IP Address configuration based on the IP address passed to the method. For the policy to fire, a request must come from the machine with the passed IP address.

Syntax

The CreateIPConfigSingleHost method has the following format:

```
Netegrity::PolicyMgtPolicy->CreateIPConfigSingleHost(ipAddr)
```

Parameters

The CreateIPConfigSingleHost method accepts the following parameter:

ipAddr (string)

Specifies the IP address required for the policy to fire.

Return Value

The CreateIPConfigSingleHost method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** if the call is unsuccessful

CreateIPConfigSubnetMask Method—Creates an IP Address Configuration Based on the IP Address and Subnet Mask

The CreateIPConfigSubnetMask method creates an IP Address configuration based on the IP address and subnet mask passed to the method. For the policy to fire, a request must come from the subnet address derived from the passed IP address and subnet mask.

Syntax

The CreateIPConfigSubnetMask method has the following format:

```
Netegrity::PolicyMgtPolicy->CreateIPConfigSubnetMask(ipAddr, subnetMask)
```

Parameters

The CreateIPConfigSubnetMask method accepts the following parameters:

ipAddr (string)

Specifies the IP address used to derive the subnet address.

subnetMask (unsigned long)

Specifies the subnet mask used to derive the subnet address.

Return Value

The CreateIPConfigSubnetMask method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** if the call is unsuccessful

Remarks

The subnet mask value is a number of bits. To arrive at this value, count the bits in the binary value of the address. For example, suppose the subnet mask is 255.255.255.128. The binary format is:

11111111 11111111 11111111 10000000

Counting from left to right, the number to pass in subnetMask would be 25.

DeleteIPConfig Method—Deletes the Specified IP Configuration Object

The DeleteIPConfig method deletes the specified IP configuration object.

Syntax

The DeleteIPConfig method has the following format:

```
Netegrity::PolicyMgtPolicy->DeleteIPConfig(ipConfig)
```

Parameters

The DeleteIPConfig method accepts the following parameters:

ipConfig (PolicyMgtIPConfig)

Specifies the IP configuration object to delete.

Return Value

The DeleteIPConfig method returns one of the following values:

- 0 if the deletion is successful
- -1 if the call is unsuccessful

Description Method—Sets or Retrieves the Description of the Policy

The Description method sets or retrieves the description of the policy.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtPolicy->Description([policyDesc])
```

Parameters

The Description method accepts the following parameter:

policyDesc (string)

Specifies the description to set.

Return Value

The Description method returns one of the following values:

- The new or existing policy description
- An empty if the call is unsuccessful

EnforceANDEvaluation Method—Sets or Retrieves the ANDUser/Group Flag

The EnforceANDEvaluation method sets or retrieves the ANDUser/Group flag depending on the value of the iANDUserFlag.

Syntax

The EnforceANDEvaluation method has the following format:

```
Netegrity::PolicyMgtPolicy->EnforceANDEvaluation(user[, iANDUserFlag])
```

Parameters

The EnforceANDEvaluation method accepts the following parameters:

user (PolicyMgtUser)

Specifies the user for which to set or retrieve *iANDUserFlag*.

iANDUserFlag (int)

(Optional) Specifies whether to enforce AND evaluation:

1 to enforce AND evaluation

0 to remove AND evaluation

If this argument is not passed, the function returns the current value of *iANDUserFlag*. This flag applies to all the users added to the policy for a particular user directory.

Return Value

The EnforceANDEvaluation method returns one of the following values:

- 0 if ANDUser/Group flag is removed successfully.
- 1 if ANDUser/Group flag is set successfully.
- -1 if the call is unsuccessful

ExcludeUser Method—Excludes or Includes a User from the Policy

The ExcludeUser method excludes or includes a user from the policy depending on the value of *iExcludeFlag*.

Syntax

The ExcludeUser method has the following format:

```
Netegrity::PolicyMgtPolicy->ExcludeUser(user[, iExcludeFlag])
```

Parameters

The ExcludeUser method accepts the following parameters:

user (PolicyMgtUser)

Specifies the user to exclude or include.

iExcludeFlag (int)

(Optional) Specifies whether to exclude the specified user:

- 1 to exclude the user
- 0 to include the user

If this argument is not passed, the function returns the current value of *iExcludeFlag*.

Return Value

The ExcludeUser method returns one of the following values:

- 0 if the user is included successfully.
- 1 if the user is excluded successfully.
- -1 if the call is unsuccessful

GetAllIPConfigs Method—Retrieves All IP Address Restriction Objects in the Policy

The GetAllIPConfigs method retrieves all IP address restriction objects in the policy.

Syntax

The GetAllIPConfigs method has the following format:

```
Netegrity::PolicyMgtPolicy->GetAllIPConfigs( )
```

Parameters

The GetAllIPConfigs method accepts no parameters.

Return Value

The GetAllIPConfigs method returns one of the following values:

- An array of PolicyMgtIPConfig objects
- **undef** if no IP address restriction objects are found.

Remarks

See the PolicyMgtIPConfig->GetType method for information about IP address restrictions and IP address restriction types.

GetAllRules Method—Retrieves All Rules Associated with the Policy

The GetAllRules method retrieves all rules associated with the policy.

Syntax

The GetAllRules method has the following format:

```
Netegrity::PolicyMgtPolicy->GetAllRules()
```

Parameters

The GetAllRules method accepts no parameters.

Return Value

The GetAllRules method returns one of the following values:

- An array of PolicyMgtRule objects
- **undef** if no rules are found, or if the call is unsuccessful

GetAllUsers Method—Retrieves All Users Associated with the Policy

The GetAllUsers method retrieves all users associated with the policy. If a user directory is specified, only those users associated with that directory are retrieved.

Syntax

The GetAllUsers method has the following format:

```
Netegrity::PolicyMgtPolicy->GetAllUsers([userDir])
```

Parameters

The GetAllUsers method accepts the following parameter:

userDir (PolicyMgtUserDir)

(Optional) Specifies that only users associated with this user directory are retrieved.

Return Value

The GetAllUsers method returns one of the following values:

- An array of PolicyMgtUser objects
- **undef** if no users were found, or if the call is unsuccessful

IsEnabled Method—Enables or Disables the Policy

The IsEnabled method enables or disables the policy.

Syntax

The IsEnabled method has the following format:

```
Netegrity::PolicyMgtPolicy->IsEnabled([enableFlag])
```

Parameters

The IsEnabled method accepts the following parameter:

enableFlag (int)

(Optional) Specifies whether to enable or disable the policy:

- 0 disables the policy.
- 1 enables the policy.

Return Value

The IsEnabled method returns one of the following values:

- 1 if the policy is enabled.
- 0 if the policy is disabled.
- -1 if the call is unsuccessful

Name Method—Sets or Retrieves the Policy Name

The Name method sets or retrieves the policy name.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtPolicy->Name([policyName])
```

Parameters

The Name method accepts the following parameter:

policyName (string)

(Optional) Specifies the name to assign to the policy.

Return Value

The Name method returns one of the following values:

- A new or existing policy name
- **undef** if the call is unsuccessful

RemoveResponse Method—Removes the Response for a Configured Rule in the Policy

The RemoveResponse method removes the response for a configured rule in the policy.

Syntax

The RemoveResponse method has the following format:

```
Netegrity::PolicyMgtPolicy->RemoveResponse(rule)
```

Parameters

The RemoveResponse method accepts the following parameter:

rule (PolicyMgtRule)

Specifies the rule whose response should be removed.

Return Value

The RemoveResponse method returns one of the following values:

- 0 if the call is successful
- -1 if the call is unsuccessful

RemoveRule Method—Removes the Specified Rule from the Policy

The RemoveRule method Removes the specified rule from the policy.

Syntax

The RemoveRule method has the following format:

```
Netegrity::PolicyMgtPolicy->RemoveRule(rule)
```

Parameters

The RemoveRule method accepts the following parameter:

rule (PolicyMgtRule)

Specifies the rule to remove.

Return Value

The RemoveRule method returns one of the following values:

- 0 if the call is successful
- -1 if the call is unsuccessful

RemoveUser Method—Removes a User from the Policy

The RemoveUser method removes a user from the policy.

Syntax

The RemoveUser method has the following format:

```
Netegrity::PolicyMgtPolicy->RemoveUser(user)
```

Parameters

The RemoveUser method accepts the following parameters:

user (PolicyMgtUser)

Specifies the user to remove.

Return Value

The RemoveUser method returns one of the following values:

- 0 if the call is successful
- -1 if the call is unsuccessful

SetResponse Method—Sets the Response for a Configured Rule in the Policy

The SetResponse method sets the response for a configured rule in the policy.

Syntax

The SetResponse method has the following format:

```
Netegrity::PolicyMgtPolicy->SetResponse(rule, response)
```

Parameters

The SetResponse method accepts the following parameters:

rule (PolicyMgtRule)

Specifies the rule whose response is being set.

response (PolicyMgtResponse)

Specifies the response to set.

Return Value

The SetResponse method returns one of the following values:

- 0 if the call is successful.
- -1 if the call is unsuccessful.

Policy Server Connectivity Methods

The following methods define TCP/IP connectivity information for a PolicyMgtServer object:

- GetPorts Method—Deprecated; replaced by the GetServerPort Method
- GetServerPort Method—For non-clustered servers, retrieves an array of TCP/IP ports corresponding to the Accounting, Authentication, and Authorization ports. For clustered servers, retrieves the Policy Server port.
- GetServerAddress Method—Retrieves the TCP/IP address of the Policy Server

GetPorts Method—Deprecated

The GetPorts method is deprecated in SiteMinder v6.0 and replaced by the GetServerPort method.

GetServerAddress Method—Retrieves the Host Name or IP Address of the Policy Server

The GetServerAddress method retrieves the Host Name or IP address of the Policy Server.

Syntax

The GetServerAddress method has the following format:

```
Netegrity::PolicyMgtServer->GetServerAddress()
```

Parameters

The GetServerAddress method accepts no parameters.

Return Value

The GetServerAddress method returns one of the following values:

- A string representing the Policy Server host name or IP address
- **undef** if the call is unsuccessful

GetServerPort Method—Retrieves TCP Port for Policy Server or Server Cluster

The GetServerPort method retrieves one of the following:

- For a clustered server, retrieves an array that contains the Policy Server port.
- For a non-clustered server, retrieves an array of the Accounting, Authentication, and Authorization ports, in that order.

Syntax

The GetServerPort method has the following format:

```
Netegrity::PolicyMgtServer->GetServerPort()
```

Parameters

The GetServerPort method accepts no parameters:

Return Value

The GetServerPort method returns one of the following values:

- An array of host ports
- **undef** if the call is unsuccessful

Remarks

The single-process Policy Server introduced in SiteMinder v6.0 combines the previously separate Authentication, Authorization, and Accounting processes into one combined process whose requests go through one TCP port. As a result, the ports numbers retrieved in the array are all the same.

Realm Methods

The following methods act on PolicyMgtRealm objects:

- Agent Method—Sets or retrieves the agent for the realm
- AuthScheme Method—Sets or retrieves the authentication scheme for the realm
- AzUserDir Method—Sets or retrieves the authorization user directory for the realm
- CreateChildRealm—Creates a top-level child realm under the realm
- CreateRule Method—Creates a rule within the realm
- DeleteChildRealm—Method Deletes a top-level child realm
- DeleteRule Method—Deletes an existing rule within the realm
- Description Method—Sets or retrieves the description of the realm
- Flush Method—Flushes the realm from the resource cache
- GetAllChildRealms Method—Retrieves an array of all top-level child realms under the realm
- GetAllRules Method—Retrieves an array of all rules associated with the realm
- GetChildRealm Method—Retrieves a top-level child realm
- GetDomain Method—Retrieves the domain associated with the realm
- GetRule Method—Retrieves a rule in the realm
- IdleTimeout Method—Sets or retrieves the idle timeout before re-authentication
- MaxTimeout Method—Sets or retrieves the maximum timeout before re-authentication
- Name Method—Sets or retrieves the realm name
- ProcessAuEvents Method—Sets or retrieves the authentication event flag in the realm
- ProcessAzEvents Method—Sets or retrieves the authorization event flag in the realm

- ProtectResource Method—Sets or retrieves the default resource protection flag
- RegScheme Method—Sets or retrieves the registration scheme for the realm
- ResourceFilter Method—Sets or retrieves the realm resource filter
- SyncAudit Method—Sets or retrieves the synchronous auditing flag

Agent Method—Sets or Retrieves the Agent for the Realm

The Agent method sets or retrieves the agent for the realm.

Syntax

The Agent method has the following format:

```
Netegrity::PolicyMgtRealm->Agent([agent])
```

Parameters

The Agent method accepts the following parameters:

agent (PolicyMgtAgent)

(Optional) Specifies the agent to set for the realm.

Return Value

The Agent method returns one of the following values:

- A new or existing PolicyMgtAgent object for the realm
- **undef** if the call is unsuccessful

AuthScheme Method—Sets or Retrieves the Authentication Scheme for the Realm

The AuthScheme method sets or retrieves the authentication scheme for the realm.

Syntax

The AuthScheme method has the following format:

```
Netegrity::PolicyMgtRealm->AuthScheme([authScheme])
```

Parameters

The AuthScheme method accepts the following parameter:

authScheme (PolicyMgtAuthScheme)

(Optional) Specifies the authentication scheme to set for the realm.

Return Value

The AuthScheme method returns one of the following values:

- A New or existing PolicyMgtAuthScheme object for the realm
- **undef** if the call is unsuccessful

AzUserDir Method—Sets or Retrieves the Authorization User Directory for the Realm

The AzUserDir method sets or retrieves the authorization user directory for the realm.

Syntax

The AzUserDir method has the following format:

```
Netegrity::PolicyMgtRealm->AzUserDir([dir])
```

Parameters

The AzUserDir method accepts the following parameter:

dir (PolicyMgtUserDirectory)

(Optional) Specifies the authorization user directory to set for the realm.

Return Value

The AzUserDir method returns one of the following values:

- A new or existing PolicyMgtUserDir object for the realm
- **undef** if none exists, or if the call is unsuccessful

CreateChildRealm Method—Creates and Configures a Child Realm

The CreateChildRealm method creates and configures a realm directly under the realm on which this method was called.

Syntax

The CreateChildRealm method has the following format:

```
Netegrity::PolicyMgtRealm->CreateChildRealm(realmName, agent, authScheme [,
realmDesc] [, resFilter] [, procAuthEvents] [, procAzEvents] [, protectAll] [,
maxTimeout] [, idleTimeout] [, syncAudit] [, azUserDir] [, regScheme])
```

Parameters

The CreateChildRealm method accepts the following parameters:

realmName (string)

Specifies the name of the realm.

agent (PolicyMgtAgent)

Specifies the agent or agent group for the realm.

authScheme (PolicyMgtAuthScheme)

Specifies the authentication scheme to associate with the realm.

realmDesc (string)

(Optional) Specifies the realm description.

resFilter (string)

(Optional) Specifies the resource filter for the realm.

procAuthEvents (int)

(Optional) Specifies a flag for processing authentication events: 1 to enable, or 0 to disable. The default is enabled.

procAzEvents (int)

(Optional) Specifies a flag for processing authorization events: 1 to enable, or 0 to disable. The default is enabled.

protectAll (int)

(Optional) Specifies a flag for activating default resource protection:1 to enable, or 0 to disable. The default is enabled.

maxTimeout (int)

(Optional) Specifies the maximum time, in seconds, a user can access the realm before re-authentication is required. The default is 7200 (2 hours).

idleTimeout (int)

(Optional) Specifies the maximum time a user can remain inactive in the realm before re-authentication is required. The default is 3600 (1 hour).

syncAudit (int)

(Optional) Specifies a flag for enabling synchronous auditing: 1 to enable, or 0 to disable. When this flag is enabled, SiteMinder logs Policy Server and agent actions before it allows access to resources. The default is enabled.

azUserDir (PolicyMgtUserDir)

(Optional) Specifies the directory where users in the realm will be authorized. The default is the default directory.

regScheme (PolicyMgtRegScheme)

(Optional) Specifies the registration scheme used to register new users accessing resources in the realm.

Return Value

The CreateChildRealm method returns one of the following values:

- A PolicyMgtRealm object
- **undef** if the call is unsuccessful

Remarks

This method creates a realm that is configured for non-persistent sessions. To configure the realm for SiteMinder 5.0 persistent sessions, edit the realm in the Administrative UI.

Note: The Policy Management API only manipulates realms that are direct descendants of the object whose method has been called, as follows:

- For a realm under a domain. You can only manipulate the top-level realms in a domain object.
- For a realm under a realm. You can only manipulate realms that are directly under the parent realm.

CreateRule Method—Creates and Configures a Rule under the Realm

The CreateRule method creates and configures a rule under the realm.

Syntax

The CreateRule method has the following format:

```
Netegrity::PolicyMgtRealm->CreateRule( ruleName [, ruleDesc] [, action] [, resource]
[, allowAccess] [, regexMatch] [, activeExpr] [, isEnabled] )
```

Parameters

The CreateRule method accepts the following parameters:

ruleName (string)

Specifies the name of the rule.

ruleDesc (string)

(Optional) Specifies the description of the rule.

action (string)

(Optional) Specifies the type of action that the rule will execute. One of the following actions:

For action type Web Agent actions, use one or more of the following HTTP actions. Use commas to separate multiple actions:

- GET. Retrieves a resource for viewing through HTTP.
- POST. Posts user-supplied information through HTTP.
- PUT. Supports legacy HTTP actions.

For action type Authentication events:

- OnAuthAccept. Occurs when a user successfully authenticates.
- OnAuthAttempt. Occurs when a user fails to authenticate because no user name was supplied.
- OnAuthChallenge. May be used in custom authentication schemes to trigger a response.
- OnAuthReject. Occurs when a user fails to authenticate.
- OnAuthUserNotFound. Used to trigger Active Responses.

For action type Authorization events:

- OnAccessAccept. Occurs when SiteMinder successfully authorizes a user to access the resource.
- OnAccessReject. Occurs when SiteMinder rejects a user because the user is not authorized to access the resource.

resource (string)

(Optional) Specifies the resource protected by the rule. This value doesn't apply to action type Authentication events.

allowAccess (int)

(Optional) Specifies a flag to allow or deny access to the resource protected by the rule: 1 allows access, or 0 denies access. This flag applies only to **action** values of type GET, PUT, and/or POST. The default is 1.

regexMatch (int)

(Optional) Specifies a flag to allow regular expression pattern matching in the **resource** field : 1 allows regular expression matching, and 0 denies regular expression matching. This flag doesn't apply to action type Authentication events. The default is 0.

activeExpr (string)

(Optional) Specifies the active expression associated with the rule.

isEnabled (int)

(Optional) Specifies a flag to enable or disable the rule:1 to enable, or 0 to disable. The default is enabled.

Return Value

The CreateRule method returns one of the following values:

- A PolicyMgtRule object
- **undef** if the call is unsuccessful

DeleteChildRealm Method—Deletes a Top-level Realm within the Realm

The DeleteChildRealm method deletes a top-level realm within the realm.

Syntax

The DeleteChildRealm method has the following format:

```
Netegrity::PolicyMgtRealm->DeleteChildRealm(realm)
```

Parameters

The DeleteChildRealm method accepts the following parameter:

realm (PolicyMgtRealm)

Specifies the child realm to delete.

Return Value

The DeleteChildRealm method returns one of the following values:

- 0 on success, or is the realm was not found
- -1 if the call is unsuccessful

DeleteRule Method—Deletes an Existing Rule within the Realm

The DeleteRule method deletes an existing rule within the realm.

Syntax

The DeleteRule method has the following format:

```
Netegrity::PolicyMgtRealm->DeleteRule(rule)
```

Parameters

The DeleteRule method accepts the following parameter:

rule (PolicyMgtRule)

Specifies the rule to delete.

Return Value

The DeleteRule method returns one of the following values:

- 0 on success
- -1 if the call is unsuccessful, or if the rule is not part of the realm being used to delete the rule

Description Method—Sets or Retrieves the Description of the Realm

The Description method sets or retrieves the description of the realm.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtRealm->Description([realmDesc])
```

Parameters

The Description method accepts the following parameter:

realmDesc (string)

(Optional) Specifies the description to assign to the realm.

Return Value

The Description method returns one of the following values:

- A new or existing realm description.
- An empty string if the call is unsuccessful

Flush Method—Flushes the Realm from the Resource Cache

The Flush method flushes the realm from the resource cache.

Syntax

The Flush method has the following format:

```
Netegrity::PolicyMgtRealm->Flush()
```

Parameters

The Flush method accepts no parameters.

Return Value

The Flush method returns one of the following values:

- 0 on success
- -1 if the call is unsuccessful

GetAllChildRealms Method—Retrieves All Top-level Realms within the Realm

The GetAllChildRealms method retrieves all top-level realms within the realm. Returns only the children.

Syntax

The GetAllChildRealms method has the following format:

```
Netegrity::PolicyMgtRealm->GetAllChildRealms()
```

Parameters

The GetAllChildRealms method accepts no parameters.

Return Value

The GetAllChildRealms method returns one of the following values:

- An array of PolicyMgtRealm objects
- **undef** if the call is unsuccessful

GetAllRules Method—Retrieves the Rules Associated with the Realm

The GetAllRules method retrieves the rules associated with the realm.

Syntax

The GetAllRules method has the following format:

```
Netegrity::PolicyMgtRealm->GetAllRules()
```

Parameters

The GetAllRules method accepts no parameters.

Return Value

The GetAllRules method returns one of the following values:

- An array of PolicyMgtRule objects
- **undef** if the call is unsuccessful

GetChildRealm Method—Retrieves a Top-level Child Realm under the Realm

The GetChildRealm method retrieves a top-level child realm under the realm. This method only searches child realms.

Syntax

The GetChildRealm method has the following format:

```
Netegrity::PolicyMgtRealm->GetChildRealm(realmName)
```

Parameters

The GetChildRealm method accepts the following parameter:

realmName (string)

Specifies the realm to check for child realms.

Return Value

The GetChildRealm method returns one of the following values:

- A PolicyMgtRealm object
- **undef** if the call is unsuccessful, or if the realm does not exist

GetDomain Method—Retrieves the Domain Associated with the Realm

The GetDomain method retrieves the domain associated with the realm.

Syntax

The GetDomain method has the following format:

```
Netegrity::PolicyMgtRealm->GetDomain()
```

Parameters

The GetDomain method accepts parameters.

Return Value

The GetDomain method returns one of the following values:

- Existing PolicyMgtDomain object for the realm
- **undef** if the call is unsuccessful

GetRule Method—Retrieves an Existing Rule in the Realm

The GetRule method retrieves an existing rule in the realm.

Syntax

The GetRule method has the following format:

```
Netegrity::PolicyMgtRealm->GetRule(ruleName)
```

Parameters

The GetRule method accepts the following parameter:

ruleName (string)

Specifies the name of the rule to retrieve.

Return Value

The GetRule method returns one of the following values:

- A PolicyMgtRule object
- **undef** if the call is unsuccessful, or if he specified rule does not exist

IdleTimeout Method—Sets or Retrieves the Maximum Time a User Can Remain Inactive in the Realm

The IdleTimeout method sets or retrieves the maximum time a user can remain inactive in the realm before re-authentication is required.

Syntax

The IdleTimeout method has the following format:

```
Netegrity::PolicyMgtRealm->IdleTimeout([idleTimeout])
```

Parameters

The IdleTimeout method accepts the following parameter:

idleTimeout (type)

(Optional) Specifies the idle timeout value, in seconds.

Return Value

The IdleTimeout method returns one of the following values:

- The existing timeout value if no argument is specified
- The new timeout value if *idleTimeout* is specified
- -1 if the call is unsuccessful

MaxTimeout Method—Sets or Retrieves the Maximum Time a User Can Access the Realm

The MaxTimeout method sets or retrieves the maximum time a user can access the realm before re-authentication is required.

Syntax

The MaxTimeout method has the following format:

```
Netegrity::PolicyMgtRealm->MaxTimeout([maxTimeout])
```

Parameters

The MaxTimeout method accepts the following parameter:

maxTimeout (int)

(Optional) Specifies the maximum timeout value, in seconds.

Return Value

The MaxTimeout method returns one of the following values:

- The existing maximum timeout value if no argument is specified.
- The new maximum timeout value if *maxTimeout* is specified.
- -1 if the call is unsuccessful

Name Method—Sets or Retrieves the Realm Name

The Name method sets or retrieves the realm name.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtRealm->Name([realmName])
```

Parameters

The Name method accepts the following parameter:

realmName (string)

(Optional) Specifies the name to assign to the realm.

Return Value

The Name method returns one of the following values:

- The new or existing realm name
- **undef** if the call is unsuccessful

ProcessAuEvents Method—Sets or Retrieves the Authentication Event Flag in the Realm

The ProcessAuEvents method sets or retrieves the authentication event flag in the realm.

Authentication event processing affects performance. If no rules in the realm are triggered by authentication events, set this flag to 0.

Syntax

The ProcessAuEvents method has the following format:

```
Netegrity::PolicyMgtRealm->ProcessAuEvents([authFlag])
```

Parameters

The ProcessAuEvents method accepts the following parameter:

authFlag (int)

(Optional) Specifies whether authentication events are processed:

- 1 to enable even processing
- 0 to disable event processing

Return Value

The ProcessAuEvents method returns one of the following values:

- 1 if authentication events are to be processed
- 0 if authentication events are not to be processed
- -1 if the call is unsuccessful

ProcessAzEvents Method—Sets or Retrieves the Authorization Event Flag in the Realm

The ProcessAzEvents method sets or retrieves the authorization event flag in the realm.

Syntax

The ProcessAzEvents method has the following format:

```
Netegrity::PolicyMgtRealm->ProcessAzEvents([azFlag])
```

Parameters

The ProcessAzEvents method accepts the following parameter:

azFlag (int)

(Optional) Specifies whether to enable authorization event processing:

- 1 enables event processing
- 0 disables event processing

Return Value

The ProcessAzEvents method returns one of the following values:

- 1 if authorization events are to be processed
- 0 if authorization events are not to be processed
- -1 if the call is unsuccessful

Remarks

Authorization event processing affects performance. If no rules in the realm are triggered by authorization events, set this flag to 0.

ProtectResource Method—Sets or Retrieves the Current Resource Protection Flag Example

The ProtectResource method sets or retrieves the current resource protection flag.

Syntax

The ProtectResource method has the following format:

```
Netegrity::PolicyMgtRealm->ProtectResource([protectFlag])
```

Parameters

The ProtectResource method accepts the following parameter:

protectFlag (int)

(Optional) Specifies whether enable resource protection:

- 1 protects the resource
- 0 makes the resource unprotected

Return Value

The ProtectResource method returns one of the following values:

- The existing resource protection state (0 or 1) if no argument is specified
- The new resource protection state if a flag value is passed to the method
- **undef** if the call is unsuccessful

RegScheme Method—Sets or Retrieves the Registration Scheme for the Realm

The RegScheme method sets or retrieves the registration scheme for the realm.

Syntax

The RegScheme method has the following format:

```
Netegrity::PolicyMgtRealm->RegScheme([regScheme])
```

Parameters

The RegScheme method accepts the following parameter:

regScheme (PolicyMgtRegScheme)

(Optional) Specifies the registration scheme to set.

Return Value

The RegScheme method returns one of the following values:

- A PolicyMgtRegScheme object
- **undef** if the call is unsuccessful, or if no registration scheme exists

SessionDrift Method--Sets or Retrieves the Session Drift

The SessionDrift method sets or retrieves the session drift of the realm, that is, the validation period (in seconds) if enabled on a persistent realm.

Syntax

The SessionDrift method has the following format:

```
Netegrity::PolicyMgtRealm->SessionDrift([SessionDrift])
```

Parameters

The SessionDrift method accepts the following parameter:

SessionDrift (int)

(Optional) Specifies the new value, or returns the current value when not specified.

Return Value

The SessionDrift method returns one of the following values:

- The new or existing session drift value
- An Sm_PolicyApi_Status_t error code if unsuccessful

  Note: -1 (Sm_PolicyApi_Failure) is a valid return value, indicating that the session drift is not enabled

ResourceFilter Method—Sets or Retrieves the Realm Resource Filter

The ResourceFilter method sets or retrieves the realm resource filter.

Syntax

The ResourceFilter method has the following format:

```
Netegrity::PolicyMgtRealm->ResourceFilter([rFilter])
```

Parameters

The ResourceFilter method accepts the following parameter:

rFilter (string)

(Optional) Specifies the realm resource filter to set.

Return Value

The ResourceFilter method returns one of the following values:

- The new or existing realm filter
- **undef** if the call is unsuccessful

SyncAudit Method—Sets or Retrieves the Synchronous Auditing Flag

The SyncAudit method sets or retrieves the synchronous auditing flag. When this flag is enabled, SiteMinder logs Policy Server and agent actions before it allows access to resources.

Syntax

The SyncAudit method has the following format:

```
Netegrity::PolicyMgtRealm->SyncAudit([syncFlag])
```

Parameters

The SyncAudit method accepts the following parameter:

syncFlag (int)

(Optional) Specifies whether synchronous auditing is enabled:

- 1 enables synchronous auditing
- 0 disables synchronous auditing

Return Value

The SyncAudit method returns one of the following values:

- Existing synchronous auditing value (0 or 1) if no argument is specified
- New synchronous auditing value if a flag argument is passed to the method
- -1 if the call is unsuccessful

Registration Scheme Methods

The following methods act on PolicyMgtRegScheme objects:

- Description Method—Sets or retrieves the registration scheme description
- EnableLogging Method—Enables or disables registration scheme logging
- Name Method—Sets or retrieves the registration scheme name
- TemplatePath Method—Sets or retrieves the path of the registration scheme template
- UserDirectory Method—Sets or retrieves the user directory for the registration scheme
- WelcomePageURL Method—Sets or retrieves the welcome page URL for the registration scheme

Description Method—Sets or Retrieves the Registration Scheme Description

The Description method sets or retrieves the registration scheme description.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtRegScheme->Description([regDesc])
```

Parameters

The Description method accepts the following parameter:

regDesc (string)

(Optional) Specifies the description of the registration scheme.

Return Value

The Description method returns one of the following values:

- The new or existing description of the registration scheme
- An empty string if the call is unsuccessful

EnableLogging Method—Enables or Disables Registration Scheme Logging

The EnableLogging method enables or disables registration scheme logging.

Syntax

The EnableLogging method has the following format:

```
Netegrity::PolicyMgtRegScheme->EnableLogging([logFlag])
```

Parameters

The EnableLogging method accepts the following parameter:

logFlag (int)

(Optional) Specifies whether registration scheme logging is enabled:

- 1 enables logging
- 0 disables logging

Return Value

The EnableLogging method returns one of the following values:

- 1 if logging is enabled
- 0 if logging is disabled
- -1 if the call is unsuccessful

Name Method—Sets or Retrieves the Registration Scheme Name

The Name method Sets or retrieves the registration scheme name.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtRegScheme->Name([regName])
```

Parameters

The Name method accepts the following parameters:

regName (string)

(Optional) Specifies the registration scheme name.

Return Value

The Name method returns one of the following values:

- The new or existing registration scheme name
- **undef** if the call is unsuccessful

TemplatePath Method—Sets or Retrieves the Path of the Registration Scheme

The TemplatePath method sets or retrieves the path of the registration scheme template.

Syntax

The TemplatePath method has the following format:

```
Netegrity::PolicyMgtRegScheme->TemplatePath([path])
```

Parameters

The TemplatePath method accepts the following parameters:

path (string)

(Optional) Specifies the path of the registration scheme template.

Return Value

The TemplatePath method returns one of the following values:

- The new or existing template path
- **undef** if the call is unsuccessful

UserDirectory Method—Sets or Retrieves the User Directory for the Registration Scheme

The UserDirectory method sets or retrieves the user directory for the registration scheme.

Syntax

The UserDirectory method has the following format:

```
Netegrity::PolicyMgtRegScheme->UserDirectory([userDir])
```

Parameters

The UserDirectory method accepts the following parameters:

userDir (PolicyMgtUserDir)

(Optional) Specifies the user directory for the registration scheme.

Return Value

The UserDirectory method returns one of the following values:

- A PolicyMgtUserDir object
- **undef** if the call is unsuccessful, or if no user directory exists

WelcomePageURL Method—Sets or Retrieves the Welcome Page URL for the Registration Scheme

The WelcomePageURL method sets or retrieves the welcome page URL for the registration scheme.

Syntax

The WelcomePageURL method has the following format:

```
Netegrity::PolicyMgtRegScheme->WelcomePageURL([URL])
```

Parameters

The WelcomePageURL method accepts the following parameter:

URL (string)

(Optional) Specifies the welcome page URL for the registration scheme. Users are redirected to this page after successfully registering.
Format: http://my.acme.com/hr/welcome.htm

Return Value

The WelcomePageURL method returns one of the following values:

- The new or existing URL
- **undef** if the call is unsuccessful

Response Methods

The following methods act on PolicyMgtResponse objects:

- CreateActiveAttribute Method—Creates an Active Response attribute
- CreateAttribute Method—Creates a Static response attribute
- CreateVariableAttribute Method—Creates a Variable Definition response attribute
- DeleteAttribute Method—Deletes a response attribute in the response
- Description Method—Sets or retrieves the response description
- GetAllAttributes Method—Retrieves an array of response attributes
- Name Method—Sets or retrieves the response name

CreateAttribute Method—Creates a Static Response Attribute for the Response

The CreateAttribute method creates a Static response attribute for the response.

Syntax

The CreateAttribute method has the following format:

```
Netegrity::PolicyMgtResponse->CreateAttribute(attrName, varValue [, TTL])
```

Parameters

The CreateAttribute method accepts the following parameters:

attrName (string)

Specifies the name of the attribute to create. Valid attribute names vary with the type of agent associated with the response.

Agent type is specified in the SiteMinder Response Dialog, which is displayed when you create a response. To see the list of attributes associated with a given agent type, select the agent type in the SiteMinder Response Dialog, click Create, then view the choices in the Attribute field of the SiteMinder Response Attribute Editor.

For example, if you are creating a response with a SiteMinder Web Agent type, you can create any of the following response attributes:

- WebAgent-HTTP-Header-Variable
- WebAgent-HTTP-Cookie-Variable
- WebAgent-OnAccept-Redirect
- WebAgent-OnAccept-Text
- WebAgent-OnAuthAccept-Session-Idle-Timeout
- WebAgent-OnAuthAccept-Session-Max-Timeout
- WebAgent-OnReject-Redirect
- WebAgent-OnReject-Text

varValue (string)

Specifies the value of the static attribute. This value appears in the Value column of the SiteMinder Response Dialog. The value represents either a variable or cookie value or a name/value pair. If you need to specify a name as well as a value, use the form name=value. For example, the attribute WebAgent-HTTP-Header-Variable requires a name/value pair. If the name is show_content and the value is yes, you would assign show_content=yes to varValue.

TTL (int)

(Optional) Specifies the amount of time in seconds that can elapse before the value of the response attribute is recalculated.

Return Value

The CreateAttribute method returns one of the following values:

- A PolicyMgtResponseAttr object
- **undef** if the call is unsuccessful

Remarks

You cannot create response attributes of type User Attribute or DN Attribute with the Command Line Interface.

See also the descriptions of the PolicyMgtResponse->CreateActiveAttribute method and the PolicyMgtResponse->CreateVariableAttribute method.

DeleteAttribute Method—Deletes a Response Attribute in the Response

The DeleteAttribute method deletes a response attribute in the response.

Syntax

The DeleteAttribute method has the following format:

```
Netegrity::PolicyMgtResponse->DeleteAttribute(respAttr)
```

Parameters

The DeleteAttribute method accepts the following parameter:

respAttr (PolicyMgtResponseAttr)

Specifies the response attribute to delete.

Return Value

The DeleteAttribute method returns one of the following values:

- 0 on success
- -1 if the call is unsuccessful

Description Method—Sets or Retrieves the Response Description

The Description method sets or retrieves the response description.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtResponse->Description([resDesc])
```

Parameters

The Description method accepts the following parameter:

resDesc (string)

(Optional) Specifies the response description.

Return Value

The Description method returns one of the following values:

- The new or existing response description
- An empty string if the call is unsuccessful

GetAllAttributes Method—Retrieves a List of Configured Response Attributes

The GetAllAttributes method retrieves a list of configured response attributes.

Syntax

The GetAllAttributes method has the following format:

```
Netegrity::PolicyMgtResponse->GetAllAttributes()
```

Parameters

The GetAllAttributes method accepts no parameters:

Return Value

The GetAllAttributes method returns one of the following values:

- An array of PolicyMgtResponseAttr objects
- **undef** if the call is unsuccessful

Name Method—Sets or Retrieves the Response Name

The Name method sets or retrieves the response name.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtResponse->Name([resName])
```

Parameters

The Name method accepts the following parameter:

resName (string)

(Optional) Specifies the response name.

Return Value

The Name method returns one of the following values:

- A new or existing response name
- **undef** if the call is unsuccessful

Response Attribute Methods

The following methods act on PolicyMgtResponseAttr objects:

- GetActiveExpr Method—Retrieves the active expression that is defined for the response attribute
- GetAgentTypeAttrName Method—Retrieves the name of the agent type attribute associated with this response attribute
- GetTTL Method—Retrieves the Time To Live (TTL) setting
- GetValue Method—Retrieves the response attribute value
- GetVariable Method—Retrieves the variable object used in the response attribute's active expression

GetAgentTypeAttrName Method—Retrieves the Name of the Agent Type Attribute

The GetAgentTypeAttrName method retrieves the name of the agent type attribute associated with this response attribute.

Syntax

The GetAgentTypeAttrName method has the following format:

```
Netegrity::PolicyMgtResponseAttr->GetAgentTypeAttrName()
```

Parameters

The GetAgentTypeAttrName method accepts no parameters.

Return Value

The GetAgentTypeAttrName method returns one of the following values:

- The agent type attribute name (for example, WebAgent-OnReject-Redirect).
- **undef** if the call is unsuccessful

GetTTL Method—Retrieves the Time To Live (TTL) Setting

The GetTTL method retrieves the Time To Live (TTL) setting.

Syntax

The GetTTL method has the following format:

```
Netegrity::PolicyMgtResponseAttr->GetTTL()
```

Parameters

The GetTTL method accepts no parameters.

Return Value

The GetTTL method returns one of the following values:

- The existing TTL setting
- **undef** if the call is unsuccessful

GetValue Method—Retrieves the Response Attribute Value

The GetValue method retrieves the response attribute value.

Syntax

The GetValue method has the following format:

```
Netegrity::PolicyMgtResponseAttr->GetValue()
```

Parameters

The GetValue method accepts no parameters.

Return Value

The GetValue method returns one of the following values:

- The existing value of the response attribute
- **undef** if the call is unsuccessful

Rule Methods

The following methods act on PolicyMgtRule objects:

- AccessType Method—Sets or retrieves the flag that allows or denies access to the resource protected by the rule
- Action Method—Sets or retrieves the action for the rule
- ActiveExpr Method—Sets or retrieves the active expression for the rule
- Agent Method—Sets or retrieves an agent object or an agent group object associated with the rule
- Description Method—Sets or retrieves the description of the rule
- IsEnabled Method—Enables or disables the rule
- RegexMatch Method—Sets or retrieves the flag that determines whether the rule should perform regular expression pattern matching
- Resource Method—Sets or retrieves the resource protected by the rule

AccessType Method—Sets or Retrieves the Flag that Allows or Denies Access to the Resource Protected by the Rule

The AccessType method sets or retrieves the flag that allows or denies access to the resource protected by the rule.

Syntax

The AccessType method has the following format:

```
Netegrity::PolicyMgtRule->AccessType([allowAccess])
```

Parameters

The AccessType method accepts the following parameter:

allowAccess (int)

(Optional) Specifies whether the rule allows access to the resource:

- 1 if the rule allows access to the resource
- 0 if the rule denies access to the resource

Return Value

The AccessType method returns one of the following values:

- 1 if the rule allows access to the resource
- 0 if the rule denies access to the resource
- -1 if the call is unsuccessful

Action Method—Sets or Retrieves the Action for the Rule

The Action method sets or retrieves the action for the rule.

Syntax

The Action method has the following format:

```
Netegrity::PolicyMgtRule->Action([action])
```

Parameters

The Action method accepts the following parameter:

action (string)

(Optional) Specifies the action to perform, as follows:

For action type Web Agent actions, use one or more of the following HTTP actions. Use commas to separate multiple actions:

- GET. Retrieves a resource for viewing through HTTP.
- POST. Posts user-supplied information through HTTP.
- PUT. Supports legacy HTTP actions.

For action type Authentication events:

- OnAuthAccept. Occurs when a user successfully authenticates.
- OnAuthAttempt. Occurs when a user fails to authenticate because no user name was supplied.
- OnAuthChallenge. May be used in custom authentication schemes to trigger a response.
- OnAuthReject. Occurs when a user fails to authenticate.
- OnAuthUserNotFound. Used to trigger Active Responses.

For action type Authorization events:

- OnAccessAccept. Occurs when SiteMinder successfully authorizes a user to access the resource.
- OnAccessReject. Occurs when SiteMinder rejects a user because the user is not authorized to access the resource.

Return Value

The Action method returns one of the following values:

- The new or the existing rule action
- **undef** if the call is unsuccessful

ActiveExpr Method—Sets or Retrieves the Active Expression for the Rule

The ActiveExpr method sets or retrieves the active expression for the rule.

Syntax

The ActiveExpr method has the following format:

```
Netegrity::PolicyMgtRule->ActiveExpr([expr])
```

Parameters

The ActiveExpr method accepts the following parameters:

expr (string)

(Optional) Specifies the active expression to execute.

Return Value

The ActiveExpr method returns one of the following values:

- The new or the existing active expression
- **undef** if the call is unsuccessful

Agent Method—Sets or Retrieves an Agent Object or an Agent Group Object Associated with the Global Rule

The Agent method sets or retrieves an agent object or an agent group object associated with the global rule.

Syntax

The Agent method has the following format:

```
Netegrity::PolicyMgtRule->Agent(agentObject)
```

Parameters

The Agent method accepts the following parameter:

agentObject (*objectType*)

Specifies the agent object or agent group object to associate with the rule. objectType can be either PolicyMgtAgent or PolicyMgtGroup.

Return Value

The Agent method returns a new or existing PolicyMgtAgent object or PolicyMgtGroup object.

Remarks

After the rule is created, the agent associated with the rule can be changed only within the same agent type (such as Web Agent).

Note: Rules that have domain scope are associated with agents indirectly, through a realm.

Description Method—Sets or Retrieves the Description of the Rule

The Description method sets or retrieves the description of the rule.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtRule->Description([ruleDesc])
```

Parameters

The Description method accepts the following parameter:

ruleDesc (string)

(Optional) Specifies the description of the rule.

Return Value

The Description method returns one of the following values:

- A new or existing rule description
- An empty string if the call is unsuccessful

IsEnabled Method—Enables or Disables the Rule

The IsEnabled method enables or disables the rule.

Syntax

The IsEnabled method has the following format:

```
Netegrity::PolicyMgtRule->IsEnabled([enableFlag])
```

Parameters

The IsEnabled method accepts the following parameter:

enableFlag (type)

(Optional) Specifies whether to enable the rule:

- 1 enables the rule
- 0 disables the rule

Return Value

The IsEnabled method returns one of the following values:

- 1 if the rule is enabled
- 0 if the rule is disabled
- -1 if the call is unsuccessful

Name Method—Sets or Retrieves the Rule Name

The Name method sets or retrieves the rule name.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtRule->Name([ruleName])
```

Parameters

The Name method accepts the following parameter:

ruleName (string)

Specifies the rule name.

Return Value

The Name method returns one of the following values:

- The new or existing rule name
- **undef** if the call is unsuccessful

RegexMatch Method—Determines whether Regular Expression Pattern Matching Is Enabled

The RegexMatch method sets or retrieves the flag that determines whether regular expression pattern matching is enabled for resource-matching operations.

Syntax

The RegexMatch method has the following format:

```
Netegrity::PolicyMgtRule->RegexMatch([enableFlag])
```

Parameters

The RegexMatch method accepts the following parameters:

enableFlag (int)

(Optional) Specifies whether to allow regular expression pattern matching:

- 1 allows pattern matching
- 0 disallows pattern matching

Return Value

The RegexMatch method returns one of the following values:

- 1 if regular expression pattern matching is enabled
- 0 if regular expression pattern matching is disabled
- -1 if the call is unsuccessful

Resource Method—Sets or Retrieves the Resource Protected by the Rule

The Resource method sets or retrieves the resource protected by the rule.

Syntax

The Resource method has the following format:

```
Netegrity::PolicyMgtRule->Resource()
```

Parameters

The Resource method accepts no parameters.

Return Value

The Resource method returns one of the following values:

- The protected resource if the call is successful
- **undef** if the call is unsuccessful

SAML 2.0 Affiliation Methods

The following methods act on PolicyMgtSAMLAffiliation objects:

- GetAffiliatedSAMLAuthSchemes Method—Retrieves all the SAML 2.0 authentication schemes associated with this SAML affiliation
- GetAffiliatedSAMLServiceProviders Method—Retrieves all the SAML 2.0 Service Providers associated with this SAML affiliation
- Property Method—Sets or retrieves the specified SAML 2.0 metadata property for this SAML 2.0 affiliation
- Save Method—Saves any SAML 2.0 metadata properties modified through one or more calls to the Property method.

GetAffiliatedSAMLAuthSchemes Method—Retrieves the SAML 2.0 Authentication Schemes Associated with This SAML Affiliation

The GetAffiliatedSAMLAuthSchemes method retrieves all the SAML 2.0 authentication schemes associated with this SAML affiliation.

Syntax

The GetAffiliatedSAMLAuthSchemes method has the following format:

```
Netegrity::PolicyMgtSAMLAffiliation->GetAffiliatedSAMLAuthSchemes()
```

Parameters

The GetAffiliatedSAMLAuthSchemes method accepts no parameters.

Return Value

The GetAffiliatedSAMLAuthSchemes method returns one of the following values:

- An array of PolicyMgtAuthScheme objects based on the SAML 2.0 Template
- **undef** if the call is unsuccessful

GetAffiliatedSAMLServiceProviders Method—Retrieves the SAML 2.0 Service Providers Associated with this SAML Affiliation

The GetAffiliatedSAMLServiceProviders method Retrieves all the SAML 2.0 Service Providers associated with this SAML affiliation.

Syntax

The GetAffiliatedSAMLServiceProviders method has the following format:

```
Netegrity::PolicyMgtSAMLAffiliation->GetAffiliatedSAMLServiceProviders()
```

Parameters

The GetAffiliatedSAMLServiceProviders method accepts no parameters.

Return Value

The GetAffiliatedSAMLServiceProviders method returns one of the following values:

- An array of PolicyMgtSAMLServiceProvider objects
- **undef** if the call is unsuccessful

Property Method—Sets or Retrieves the Specified SAML 2.0 Metadata Property

The Property method sets or retrieves the specified SAML 2.0 metadata property for this SAML 2.0 affiliation.

Syntax

The Property method has the following format:

```
Netegrity::PolicyMgtSAMLAffiliation->Property(name [, value])
```

Parameters

The Property method accepts the following parameters:

name (string)

Specifies the property to set or retrieve.

value (string)

(Optional) Specifies the value of the property being set.

Return Value

The Property method returns one of the following values:

- The new or existing property value
- **undef** if the call is unsuccessful

Remarks

For a list of affiliation metadata properties, see the description of the PolicyMgtSession->CreateSAMLAffiliation (see page 441) method.

Note: After modifying one or more existing affiliation properties with this method, call PolicyMgtSAMLAffiliation->Save (see page 392) to write the changes to the policy store.

Save Method—Saves the Changes to the SAML 2.0 Metadata Properties of this SAML 2.0 Affiliation

The Save method saves the changes you made to the SAML 2.0 metadata properties of this SAML 2.0 affiliation.

Syntax

The Save method has the following format:

```
Netegrity::PolicyMgtSAMLAffiliation->Save()
```

Parameters

The Save method accepts no parameters.

Return Value

The Save method returns one of the following values:

- 0 on success
- -1 if the call is unsuccessful
- -4 if the user has insufficient privileges to save the changes
- -10 if the path and class are empty

Remarks

To modify an affiliation property, call the PolicyMgtSAMLAffiliation->Property method.

SAML 2.0 Indexed Endpoint Methods

The following methods act on PolicyMgtSAMLSPACS objects:

- GetACSIndex Method—Retrieves Index Value of Assertion Consumer Service Object
- GetACSBinding Method—Retrieves Protocol Binding of Assertion Consumer Service Object
- GetACSURL Method—Retrieves URL Value of Assertion Consumer Service Object
- GetIsDefault Method—Retrieves IsDefault Value for Assertion Consumer Service Object

GetACSIndex Method—Retrieves Index Value of Assertion Consumer Service Object

The GetACSIndex method retrieves the index value of a SAML Service Provider Assertion Consumer Service object.

Syntax

The GetACSIndex method has the following format:

```
Netegrity::PolicyMgtSAMLSPACS->GetACSIndex()
```

Parameters

The GetACSIndex method accepts no parameters.

Return Value

The GetACSIndex method returns one of the following values:

- Assertion_Consumer_Service_object_index_value
- **undef** if the call is unsuccessful

GetACSBinding Method—Retrieves Protocol Binding of Assertion Consumer Service Object

The GetACSBinding method retrieves the protocol binding of a SAML Service Provider Assertion Consumer Service object.

Syntax

The GetACSBinding method has the following format:

```
Netegrity::PolicyMgtSAMLSPACS->GetACSBinding()
```

Parameters

The GetACSBinding method accepts no parameters.

Return Value

The GetACSBinding method returns one of the following values:

- Assertion_Consumer_Service_object_protocol_binding
- **undef** if the call is unsuccessful

GetACSURL Method—Retrieves URL Value of Assertion Consumer Service Object

The GetACSURL method retrieves the URL value of a SAML Service Provider Assertion Consumer Service object.

Syntax

The GetACSURL method has the following format:

```
Netegrity::PolicyMgtSAMLSPACS->GetACSURL()
```

Parameters

The GetACSURL method accepts no parameters.

Return Value

The GetACSURL method returns one of the following values:

- Assertion_Consumer_Service_object_URL_value
- **undef** if the call is unsuccessful

GetIsDefault Method—Retrieves IsDefault Value for Assertion Consumer Service Object

The GetIsDefault method retrieves the value of IsDefault for the SAML Service Provider Assertion Consumer Service object.

Syntax

The GetIsDefault method has the following format:

```
Netegrity::PolicyMgtSAMLSPACS->GetIsDefault()
```

Parameters

The GetIsDefault method accepts no parameters.

Return Value

The GetIsDefault method returns one of the following values:

- Assertion_Consumer_Service_object_IsDefault_value
- **undef** if the call is unsuccessful

SAML 2.0 Requester Attribute Methods

The following methods act on PolicyMgtSAMLRequesterAttr objects:

- GetAttrNameFormat Method—Retrieves SAML Requester Attribute Name Format
- GetLocalName Method—Retrieves SAML Requester Attribute's Local Name
- GetName Method—Retrieves SAML Requester Attribute's Name

GetAttrNameFormat Method—Retrieves SAML Requester Attribute's Name Format

The GetAttrNameFormat method retrieves a SAML Requester attribute's name format.

Syntax

The GetAttrNameFormat method has the following format:

```
Netegrity::PolicyMgtSAMLRequesterAttr->GetAttrNameFormat()
```

Parameters

The GetAttrNameFormat method accepts no parameters.

Return Value

The GetAttrNameFormat method returns the following value:

- SAML_Requester_attribute_name_format

GetLocalName Method—Retrieves SAML Requester Attribute's Local Name

The GetLocalName method retrieves a SAMLRequester attribute's local name.

Syntax

The GetLocalName method has the following format:

```
Netegrity::PolicyMgtSAMLRequesterAttr->GetLocalName()
```

Parameters

The GetLocalName method accepts no parameters.

Return Value

The GetLocalName method returns one of the following values:

- SAML_Requester_attribute_local_name
- **undef** if the call is unsuccessful

GetName Method—Retrieves SAML Requester Attribute's Name

The GetName method retrieves a SAML Requester attribute's name.

Syntax

The GetName method has the following format:

```
Netegrity::PolicyMgtSAMLRequesterAttr->GetName()
```

Parameters

The GetName method accepts no parameters.

Return Value

The GetName method returns one of the following values:

- SAML_Requester_attribute_name
- **undef** if the call is unsuccessful

SAML 2.0 Service Provider Methods

The following methods act on PolicyMgtSAMLServiceProvider objects:

- AddAssertionConsumerService Method—Adds an Assertion Consumer Service to a SAML Service Provider Object
- AddAttribute Method—Adds an Attribute to the SAML 2.0 Service Provider
- AddUser Method—Adds a User to the SAML Service Provider
- CreateIPConfigHostName Method—Creates an IP Configuration Object for the Service Provider
- CreateIPConfigRange—Creates an IP Configuration Object for the Service Provider
- CreateIPConfigSingleHost—Creates an IP Configuration Object for the Service Provider
- CreateIPConfigSubnetMask—Creates an IP Configuration Object for the Service Provider
- DeleteIPConfig—Deletes Specified IP Configuration Object
- GetAllAttributes Method—Retrieves All Attributes for SAML 2.0 Service Provider
- GetAllIPConfigs Method—Retrieves All IP Configuration Objects
- GetAllAssertionConsumerServices Method—Retrieves All Assertion Consumer Services
- GetAllUsers Method—Retrieves All Users
- Property Method—Sets or Retrieves Metadata Property
- RemoveAssertionConsumer Method—Removes Assertion Consumer Service
- RemoveAttribute Method—Removes Specified Attribute
- RemoveUser Method—Removes Specified User
- Save Method—Saves Changes Made to Metadata Properties

AddAssertionConsumerService Method—Adds an Assertion Consumer Service to a SAML Service Provider Object

The AddAssertionConsumerService method adds an Assertion Consumer Service to a SAML Service Provider object.

Syntax

The AddAssertionConsumerService method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->AddAssertionConsumerService(index,
protocolBinding, URL)
```

Parameters

The AddAssertionConsumerService method accepts the following parameters:

index (int)

Specifies the Assertion Consumer Service Indexed Endpoint index value.

protocolBinding (string)

Specifies the protocol binding of the Assertion Consumer Service, which is one of the following:

- SAMLSP_HTTP_Post
- SAMLSP_ACS_PROTOCOLBINDING_HTTP_Artifact

URL (string)

Specifies the URL of the Indexed Endpoint.

Return Value

The AddAssertionConsumerService method returns one of the following values:

- A PolicyMgtSAMLSPACS object
- **undef** if the call is unsuccessful

AddAttribute Method—Adds an Attribute to the SAML 2.0 Service Provider

The AddAttribute method adds an attribute to the SAML 2.0 Service Provider.

Syntax

The AddAttribute method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->AddAttribute(attrNameFormat, value,
nEncrypted, nMode)
```

Parameters

The AddAttribute method accepts the following parameters:

attrNameFormat (int)

Specifies one of the following attribute formats, as defined in the SAML 2.0 standard:

- SAMLSP_UNSPECIFIED (Value=0)
- SAMLSP_URI (Value=1)
- SAMLSP_BASIC (Value=2)

value (string)

Specifies the value specification for the attribute. This value specification appears in the Name Value Pair column of the SiteMinder SAML Service Provider Properties Dialog. The format of the value specification depends upon the kind of attribute you are adding -- Static, User Attribute, or DN Attribute:

- Static attributes:

  variableName=value

- User attributes:

  variableName=<%userattr="AttrName"%>

- DN attributes:

  variableName=<#dn="DNSpec" attr="AttrName"#>

  To allow SiteMinder to retrieve DN attributes from a nested group, begin DNSpec with an exclamation mark (!) -- for example:

  dn="!ou=People,o=security.com"

nEncrypted (int)

Specifies whether the attribute is encrypted. If non-zero, the attribute is encrypted after being included in the assertion.

nMode (int)

Specifies the retrieval mode of this attribute, which is one of the following:

- SAMLSP_SSO
- SAMLSP_Attribute

Return Value

The AddAttribute method returns one of the following values:

- A PolicyMgtSAMLSPAttr object
- **undef** if the call is unsuccessful

Remarks

A SAML 2.0 attribute contains information about a principal who is trying to access a resource on the Service Provider -- for example, the principal's user DN.

The defined attribute is included in an attribute statement for all SAML 2.0 assertions that are produced for this Service Provider.

AddUser Method—Adds a User to the SAML 2.0 Service Provider

The AddUser method adds a user to the SAML Service Provider. Assertions can be generated for the users associated with a Service Provider.

Syntax

The AddUser method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->AddUser(user)
```

Parameters

The AddUser method accepts the following parameter:

user (PolicyMgtUser)

Specifies the user to add.

Return Value

The AddUser method returns one of the following values:

- 0 on success
- -1 if the call is unsuccessful

CreateIPConfigHostName Method—Creates an IP Configuration Object for the Service Provider

The CreateIPConfigHostName method creates an IP configuration object for the Service Provider, based on the specified host name.

Syntax

The CreateIPConfigHostName method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->CreateIPConfigHostName(hostName)
```

Parameters

The CreateIPConfigHostName method accepts the following parameters:

hostName (string)

Specifies the host name where assertions must originate.

Return Value

The CreateIPConfigHostName method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** if the call is unsuccessful

Remarks

This method creates an IP address restriction for the assertion generation policy. With this address restriction, only assertions generated from the specified host will be accepted.

CreateIPConfigRange Method—Creates an IP Configuration Object for the Service Provider

The CreateIPConfigRange method creates an IP configuration object for the Service Provider, based on the specified range of IP addresses.

Syntax

The CreateIPConfigRange method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->CreateIPConfigRange(ipAddr1, ipAddr2)
```

Parameters

The CreateIPConfigRange method accepts the following parameters:

ipAddr1 (string)

Specifies the first IP address in the range of valid IP addresses.

ipAddr2 (string)

Specifies the last IP address in the range of valid IP addresses.

Return Value

The CreateIPConfigRange method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** if the call is unsuccessful

Remarks

This method creates an IP address restriction for the assertion generation policy. With this address restriction, only assertions generated from the specified range of IP addresses will be accepted.

CreateIPConfigSingleHost Method—Creates an IP Configuration Object for the Service Provider

The CreateIPConfigSingleHost method creates an IP configuration object for the Service Provider, based on the specified IP address.

Syntax

The CreateIPConfigSingleHost method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->CreateIPConfigSingleHost(ipAddr)
```

Parameters

The CreateIPConfigSingleHost method accepts the following parameter:

ipAddr (string)

Specifies the IP address where assertions must originate.

Return Value

The CreateIPConfigSingleHost method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** if the call is unsuccessful

Remarks

This method creates an IP address restriction for the assertion generation policy. With this address restriction, only assertions generated from the specified IP address will be accepted.

CreateIPConfigSubnetMask Method—Creates an IP Configuration Object for the Service Provider

The CreateIPConfigSubnetMask method creates an IP configuration object for the Service Provider, based on the specified IP address and subnet mask.

Syntax

The CreateIPConfigSubnetMask method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->CreateIPConfigSubnetMask(ipAddr,
subnetMask)
```

Parameters

The CreateIPConfigSubnetMask method accepts the following parameters:

ipAddr (string)

Specifies the IP address used to derive the subnet address.

subnetMask (unsigned long)

Specifies the subnet mask used to derive the subnet address.

Return Value

The CreateIPConfigSubnetMask method returns one of the following values:

- A PolicyMgtIPConfig object
- **undef** if the call is unsuccessful

Remarks

This method creates an IP address restriction for the assertion generation policy. With this address restriction, only assertions generated from the subnet address will be accepted. The subnet address is derived from the passed IP address and subnet mask. For information about defining the subnet mask value, see the description of the PolicyMgtPolicy->CreateIPConfigSubnetMask (see page 346) method.

DeleteIPConfig Method—Deletes Specified IP Configuration Object

The DeleteIPConfig method deletes the specified IP configuration object.

Syntax

The DeleteIPConfig method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->DeleteIPConfig(IPConfig)
```

Parameters

The DeleteIPConfig method accepts the following parameter:

IPConfig (PolicyMgtIPConfig object)

Specifies the IP configuration object to delete.

Return Value

The DeleteIPConfig method returns one of the following values:

- value = 0

  Specifies that the method is successful.

- value = -1

  Specifies that the method is unsuccessful.

GetAllAttributes Method—Retrieves All Attributes for SAML 2.0 Service Provider

The GetAllAttributes method retrieves all attributes defined for the SAML 2.0 Service Provider.

Syntax

The GetAllAttributes method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->GetAllAttributes()
```

Parameters

The GetAllAttributes method accepts no parameters.

Return Value

The GetAllAttributes method returns one of the following values:

- PolicyMgtSAMLSPAttr (array)
- **undef** if the call is unsuccessful

GetAllIPConfigs Method—Retrieves All IP Configuration Objects

The GetAllIPConfigs method retrieves all IP configuration objects for the SAML 2.0 Service Provider.

Syntax

The GetAllIPConfigs method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->GetAllIPConfigs()
```

Parameters

The GetAllIPConfigs method accepts no parameters.

Return Value

The GetAllIPConfigs method returns one of the following values:

- PolicyMgtIPConfig (array)
- **undef** if no IP configuration objects are found

GetAllAssertionConsumerServices Method—Retrieves All Assertion Consumer Services

The GetAllAssertionConsumerServices method retrieves all Assertion Consumer Services from the SAML 2.0 Service Provider object.

Syntax

The GetAllAssertionConsumerServices method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->GetAllAssertionConsumerServices()
```

Parameters

The GetAllAssertionConsumerServices method accepts no parameters.

Return Value

The GetAllAssertionConsumerServices method returns one of the following values:

- PolicyMgtSAMLSPACS (array)
- **undef** if the call is unsuccessful

GetAllUsers Method—Retrieves All Users

The GetAllUsers method retrieves all users associated with the SAML 2.0 Service Provider. If a user directory is specified, only users who belong to the specified directory are returned.

Syntax

The GetAllUsers method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->GetAllUsers([userDir])
```

Parameters

The GetAllUsers method accepts the following parameter:

userDir (PolicyMgtUserDir object)

(Optional) Specifies the user directory to which all retrieved users must belong.

Return Value

The GetAllUsers method returns one of the following values:

- PolicyMgtUser (array)
- **undef** if an error occurs or no users are found

Property Method—Sets or Retrieves Metadata Property

The Property method sets or retrieves the specified SAML 2.0 metadata property for this Service Provider.

Note: After modifying one or more Service Provider properties using this method, call the PolicyMgtSAMLServiceProvider->Save method to write the changes to the policy store.

Syntax

The Property method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->Property(name[, value])
```

Parameters

The Property method accepts the following parameters:

name (string)

Specifies the property to set or retrieve.

Note: For a complete list of Service Provider metadata properties, see the method PolicyMgtAffDomain->CreateSAMLServiceProvider (see page 160).

value (string)

(Optional) Specifies a new value for the property.

Return Value

The Property method returns one of the following values:

- property_value

  Specifies the property's new or existing value.

- **undef**

  Specifies that the call is unsuccessful.

RemoveAssertionConsumer Method—Removes Assertion Consumer Service

The RemoveAssertionConsumer method removes an existing Assertion Consumer Service from a SAML 2.0 Service Provider.

Syntax

The RemoveAssertionConsumer method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->RemoveAssertionConsumer(pSAMLSPACS)
```

Parameters

The RemoveAssertionConsumer method accepts the following parameter:

pSAMLSPACS

Specifies the Assertion Consumer Service to remove.

Return Value

The RemoveAssertionConsumer method returns one of the following values:

- value = 0

  Specifies that the method is successful.

- value = -1

  Specifies that the method is unsuccessful.

RemoveAttribute Method—Removes Specified Attribute

The RemoveAttribute method removes the specified attribute from the SAML 2.0 Service Provider.

Syntax

The RemoveAttribute method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->RemoveAttribute(SAMLSPAttr)
```

Parameters

The RemoveAttribute method accepts the following parameter:

SAMLSPAttr (PolicyMgtSAMLSPAttr object)

Specifies the attribute to remove.

Return Value

The RemoveAttribute method returns one of the following values:

- value = 0

  Specifies that the method is successful.
- value = -1

  Specifies that the method is unsuccessful.

RemoveUser Method—Removes Specified User

The RemoveUser method removes the specified user from the SAML 2.0 Service Provider.

Syntax

The RemoveUser method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->RemoveUser(user)
```

Parameters

The RemoveUser method accepts the following parameter:

user (PolicyMgtUser object)

Specifies the user to remove.

Return Value

The RemoveUser method returns one of the following values:

- value = 0

  Specifies that the method is successful.

- value = -1

  Specifies that the method is unsuccessful.

Save Method—Saves Changes Made to Metadata Properties

The Save method saves any changes made to the SAML 2.0 metadata properties of the Service Provider. Call this method once after making all changes to the SAML 2.0 Service Provider. You must call this method for the changes to take effect. To modify a metadata property, call the PolicyMgtSAMLServiceProvider->Property method.

Syntax

The Save method has the following format:

```
Netegrity::PolicyMgtSAMLServiceProvider->Save()
```

Parameters

The Save method accepts no parameters.

Return Value

The Save method returns one of the following values:

- value = 0

  Specifies that the method is successful.

- value = -1

  Specifies that the method is unsuccessful.

- value = -4

  Specifies that the user does not have the privileges required to change metadata properties.

- value = -10

  Specifies that the path and class are empty.

SAML 2.0 Service Provider Attribute Methods

The following methods act on PolicyMgtSAMLSPAttr objects:

- GetAttrNameFormat Method—Retrieves Format of Attribute Names
- GetValue Method—Retrieves Service Provider Attribute Value

GetAttrNameFormat Method—Retrieves Format of Attribute Names

The GetAttrNameFormat method retrieves the format of attribute names used with the SAML 2.0 Service Provider. For more information about SAML 2.0 attributes, see the method PolicyMgtSAMLServiceProvider->AddAttribute (see page 398).

Syntax

The GetAttrNameFormat method has the following format:

```
Netegrity::PolicyMgtSAMLSPAttr->GetAttrNameFormat()
```

Parameters

The GetAttrNameFormat method accepts no parameters.

Return Value

The GetAttrNameFormat method returns one of the following values:

- SAMLSP_UNSPECIFIED (value = 0)
- SAMLSP_URI (value = 1)
- SAMLSP_BASIC (value = 2)

GetValue Method—Retrieves Service Provider Attribute Value

The GetValue method retrieves the value of the SAML 2.0 Service Provider attribute. To retrieve all attributes associated with a Service Provider, call the method PolicyMgtSAMLServiceProvider->GetAllAttributes. For more information about SAML 2.0 attributes, see the method PolicyMgtSAMLServiceProvider->AddAttribute (see page 398).

Syntax

The GetValue method has the following format:

```
Netegrity::PolicyMgtSAMLSPAttr->GetValue()
```

Parameters

The GetValue method accepts no parameters.

Return Value

The GetValue method returns one of the following values:

- Service_Provider_attribute_value
- **undef** if the call is unsuccessful

Session Methods

The following methods act on PolicyMgtSession objects:

- AddAttributeToSAMLScheme Method—Adds New Attribute to Authentication Scheme
- AddTrustedHost Method—Creates or Modifies Trusted Host Object
- CreateAdmin Method—Creates System-Level Administrator
- CreateAffDomain Method—Creates Affiliate Domain
- CreateAgent Method—Creates SiteMinder Agent
- CreateAgentConfig Method—Creates Agent Configuration Object
- CreateAgentGroup Method—Creates Agent Group
- CreateAuthAzMap Method—Creates Directory Mapping Object
- CreateAuthScheme Method—Creates Authentication Scheme
- CreateCustomCertMap Method—Creates Custom Certificate Map
- CreateDataManager Method—Creates Data Manager Object
- CreateDomain Method—Creates Policy Domain Object
- CreateExactCertMap Method—Creates Certificate Map Matching User Directory Attributes
- CreateGlobalPolicy Method—Creates Global Policy
- CreateGlobalResponse Method—Creates Global Response
- CreateGlobalRule Method—Creates Global Rule
- CreateHostConfig Method—Creates Host Configuration Object
- CreateODBCQueryScheme Method—Creates ODBC Query Scheme

- CreatePwdPolicy Method—Creates Password Policy
- CreateRegScheme Method—Creates Registration Scheme
- CreateSAMLAffiliation Method—Creates SAML 2.0 Affiliation Object
- CreateSAMLAuthScheme Method—Creates SAML Authentication Scheme Object
- CreateSingleCertMap Method—Creates Single-Attribute Certificate Map
- CreateTrustedHost Method—Creates Trusted Host Object
- CreateUserDir Method—Creates User Directory Object
- CreateWSFEDAuthScheme Method—Creates WS-Federation Authentication Scheme
- DeleteAdmin Method—Deletes Administrator
- DeleteAffDomain Method—Deletes Affiliate Domain
- DeleteAgent Method—Deletes Agent
- DeleteAgentConfig Method—Deletes Agent Configuration Object
- DeleteAuthAzMap Method—Deletes Authentication and Authorization Map
- DeleteAuthScheme Method—Deletes Authentication Scheme
- DeleteCertMap Method—Deletes Certificate Map
- DeleteDomain Method—Deletes Policy Domain
- DeleteGlobalPolicy Method—Deletes Global Policy
- DeleteGlobalResponse Method—Deletes Global Response
- DeleteGlobalRule Method—Deletes Global Rule
- DeleteGroup Method—Deletes Agent Group
- DeleteHostConfig Method—Deletes Host Configuration Object
- DeleteODBCQueryScheme Method—Deletes ODBC Query Scheme
- DeletePwdPolicy Method—Deletes Password Policy
- DeleteRegScheme Method—Deletes Registration Scheme
- DeleteSAMLAffiliation Method—Deletes SAML Affiliation Object
- DeleteTrustedHost Method—Deletes Trusted Host
- DeleteUserDir Method—Deletes User Directory
- GetAdmin Method—Retrieves Administrator
- GetAffDomain Method—Retrieves Affiliate Domain
- GetAgent Method—Retrieves Agent

- GetAgentConfig Method—Retrieves Agent Configuration Object
- GetAgentGroup Method—Retrieves Agent Group
- GetAgentType Method—Retrieves Agent Type
- GetAllAdmins Method—Retrieves List of All Administrators
- GetAllAffDomains Method—Retrieves List of All Affiliate Domains
- GetAllAgentConfigs Method—Retrieves List of All Agent Configuration Objects
- GetAllAgentGroups Method—Retrieves List of All Agent Group Objects
- GetAllAgents Method—Retrieves List of All Agents
- GetAllAuthAzMaps Method—Retrieves List of All AuthAz Maps
- GetAllAuthSchemes Method—Retrieves List of Authentication Schemes
- GetAllCertMaps Method—Retrieves List of Certificate Mapping Objects
- GetAllDomains Method—Retrieves List of All Domains
- GetAllGlobalPolicies Method—Retrieves List of All Global Policy Objects
- GetAllGlobalResponses Method—Retrieves List of All Global Response Objects
- GetAllGlobalRules Method—Retrieves List of All Global Rule Objects
- GetAllHostConfigs Method—Retrieves List of All Host Configuration Objects
- GetAllODBCQuerySchemes Method—Retrieves List of All ODBC Query Schemes
- GetAllPwdPolicies Method—Retrieves List of All Password Policies
- GetAllRegSchemes Method—Retrieves List of All Registration Schemes
- GetAllSAMLAffiliations Method—Retrieves List of All SAML 2.0 Affiliations
- GetAllSAMLSchemeAttributes Method—Retrieves List of All Requester Attributes
- GetAllTrustedHosts Method—Retrieves List of All Trusted Host Objects
- GetAllUserDirs Method—Retrieves List of All User Directories
- GetAllVariableTypes Method—Retrieves List of All Variable Type Objects
- GetAuthScheme Method—Retrieves Authentication Scheme Object
- GetCertMap Method—Retrieves Certificate Mapping Object
- GetDomain Method—Retrieves Domain Object
- GetGlobalPolicy Method—Retrieves Global Policy Object
- GetGlobalResponse Method—Retrieves Global Response Object
- GetGlobalRule Method—Retrieves Global Rule Object

- GetHostConfig Method—Retrieves Host Configuration Object
- GetODBCQueryScheme Method—Retrieves ODBC Query Scheme Object
- GetPwdPolicy Method—Retrieves Password Policy Object
- GetRegScheme Method—Retrieves Registration Scheme Object
- GetSAMLAffiliation Method—Retrieves SAML 2.0 Affiliation Object
- GetSAMLAffiliationById Method—Retrieves SAML 2.0 Affiliation Object by ID
- GetSharedSecretPolicy Method—Retrieves Shared Secret Policy Object
- GetTrustedHost Method—Retrieves Trusted Host Object
- GetUserDir Method—Retrieves User Directory Object
- GetVariableType Method—Retrieves Variable Type Object
- RemoveAttributeFromSAMLScheme Method—Removes Attribute from SAML Scheme
- SAMLAuthSchemeProperties Method—Sets or Retrieves SAML Metadata Properties
- WSFEDAuthSchemeProperties Method—Sets or Retrieves WS-Federation Properties

AddAttributeToSAMLScheme Method—Adds New Attribute to Authentication Scheme

The AddAttributeToSAMLScheme method adds a new attribute to the SAML 2.0 authentication scheme definition.

Syntax

The AddAttributeToSAMLScheme method has the following format:

```
Netegrity::PolicyMgtSession->AddAttributeToSAMLScheme(scheme, AttrNameFormat,
LocalName, Name)
```

Parameters

The AddAttributeToSAMLScheme method accepts the following parameters:

scheme (PolicyMgtAuthScheme object)

Specifies the SAML 2.0 authentication scheme.

AttrNameFormat (int)

Specifies the attribute type:

- SAMLSP_UNSPECIFIED
- SAMLSP_URI
- SAMLSP_BASIC

LocalName (string)

Specifies the attribute's name as used locally.

Name (string)

Specifies the attribute's name as defined on the Attribute Authority.

Return Value

The AddAttributeToSAMLScheme method returns one of the following values:

- PolicyMgtSAMLRequesterAttr (object)
- **undef** if the call is unsuccessful

AddTrustedHost Method—Creates or Modifies Trusted Host Object

The AddTrustedHost method creates or modifies a trusted host object in the policy store.

Syntax

The AddTrustedHost method has the following format:

```
Netegrity::PolicyMgtSession->AddTrustedHost(trustedHostName[,
trustedHostDescription][, trustedHostIpAddress][, sharedSecret])
```

Parameters

The AddTrustedHost method accepts the following parameters:

trustedHostName (string)

Specifies the name of the trusted host.

trustedHostDescription (string)

(Optional) Specifies the description of the trusted host.

trustedHostIpAddress (string)

(Optional) Specifies the IP address of the trusted host.

sharedSecret (string)

(Optional) Specifies the shared secret.

Note: You must also define the shared secret in the host configuration file by running the SiteMinder tool smreghost with the -sh option. If you do not use the -sh option to specify the shared secret, SiteMinder automatically generates one.

Return Value

The AddTrustedHost method returns one of the following values:

- PolicyMgtTrustedHost (object)
- **undef** if the trusted host name already exists

Remarks

You can use the AddTrustedHost method to register the trusted host without first configuring a connection between the Policy Server and the Agent. When you use this method to register the trusted host, you must also run the SiteMinder tool smreghost to define the shared secret in the host configuration file. (The host configuration file is named SmHost.conf by default.) Run smreghost with the -sh option and the shared secret. To retrieve the shared secret in clear text, call the method PolicyMgtTrustedHost->GetSecret.

Alternately, you can create the trusted host by calling the method CreateTrustedHost and run smreghost without the -sh option. In this case, SiteMinder automatically creates and configures the trusted host during installation.

Important! SiteMinder generates a random 128-byte ASCII shared secret. When you create the shared secret, it can be any string value. To create a strong shared secret, we strongly recommend that you call the AddTrustedHost method with the *sharedSecret* parameter set to an empty string. This results in the automatic generation of a shared secret that is random, long, and hard-to-guess.

CreateAdmin Method—Creates System-Level Administrator

The CreateAdmin method creates and configures a system-level administrator.

Syntax

The CreateAdmin method has the following format:

```
Netegrity::PolicyMgtSession->CreateAdmin(adminName[, adminDesc][, adminPwd][,
userDir][, authScheme])
```

Parameters

The CreateAdmin method accepts the following parameters:

adminName (string)

Specifies the administrator's name.

adminDesc (string)

(Optional) Specifies the administrator's description.

adminPwd (string)

(Optional) Specifies the administrator's password.

userDir (PolicyMgtUserDir object)

(Optional) Specifies the user directory if the administrator is stored in an external directory.

authScheme (PolicyMgtAuthScheme object)

(Optional) Specifies the authentication scheme to use if the administrator is stored in an external directory.

Note: This parameter is required if an external user directory is specified.

Return Value

The CreateAdmin method returns one of the following values:

- PolicyMgtAdmin (object)
- **undef** if the call is unsuccessful or the administrator name already exists

Remarks

The Policy Management API does not allow you to create an administrator for a particular domain. However, you can add an existing administrator to a particular domain by calling the method AddAdmin. To create an administrator with domain privileges, use the Administrative UI.

CreateAffDomain Method—Creates Affiliate Domain

The CreateAffDomain method creates an affiliate domain.

Syntax

The CreateAffDomain method has the following format:

```
Netegrity::PolicyMgtSession->CreateAffDomain(domName[, domDesc])
```

Parameters

The CreateAffDomain method accepts the following parameters:

domName (string)

Specifies the name of the affiliate domain.

domDesc (string)

(Optional) Specifies the description of the affiliate domain.

Return Value

The CreateAffDomain method returns one of the following values:

- PolicyMgtAffDomain (object)
- **undef** if the call is unsuccessful or the affiliate domain name already exists

Remarks

To implement affiliate domains, you need legacy federation.

CreateAgent Method—Creates SiteMinder Agent

The CreateAgent method creates and configures a SiteMinder agent.

Syntax

The CreateAgent method has the following format:

```
Netegrity::PolicyMgtSession->CreateAgent(agentName, agentType[, agentDesc][,
agentIP][, agentSecret][, realmHintAttrID])
```

Parameters

The CreateAgent method accepts the following parameters:

agentName (string)

Specifies the name of the agent.

agentType (PolicyMgtAgentType object)

Specifies the type of agent.

agentDesc (string)

(Optional) Specifies the description of the agent.

agentIP (string)

(Optional) Specifies the agent's IP address.

Note: This parameter is required for RADIUS agents.

agentSecret (string)

(Optional) Specifies the shared secret.

Note: To create a v4.x agent, specify the shared secret. To create a v5.x agent, omit this parameter.

realmHintAttrID (int)

(Optional) Specifies the realm hint attribute ID.

Note: This parameter only applies to RADIUS agents.

Return Value

The CreateAgent method returns one of the following values:

- PolicyMgtAgent (object)
- **undef** if the call is unsuccessful or the SiteMinder agent name already exists

CreateAgentConfig Method—Creates Agent Configuration Object

The CreateAgentConfig method creates an agent configuration object.

Syntax

The CreateAgentConfig method has the following format:

```
Netegrity::PolicyMgtSession->CreateAgentConfig(agentConfigName[,
AgentConfigDesc])
```

Parameters

The CreateAgentConfig method accepts the following parameters:

agentConfigName (string)

Specifies the name of the agent configuration.

AgentConfigDesc (string)

(Optional) Specifies the description of the agent configuration.

Return Value

The CreateAgentConfig method returns one of the following values:

- PolicyMgtAgentConfig (object)
- **undef** if the call is unsuccessful or the agent configuration name already exists

CreateAgentGroup Method—Creates Agent Group

The CreateAgentGroup method creates an agent group.

Syntax

The CreateAgentGroup method has the following format:

```
Netegrity::PolicyMgtSession->CreateAgentGroup(agentGroupName, agentType[,
groupDesc])
```

Parameters

The CreateAgentGroup method accepts the following parameters:

agentGroupName (string)

Specifies the name of the agent group.

agentType (PolicyMgtAgentType object)

Specifies the type of agent associated with the agent group.

Note: To retrieve the agent type for this method, call the method PolicyMgtSession->GetAgentType.

groupDesc (string)

(Optional) Specifies the description of the agent group.

Return Value

The CreateAgentGroup method returns one of the following values:

- PolicyMgtGroup (object)
- **undef** if the agent group name already exists

CreateAuthAzMap Method—Creates Directory Mapping Object

The CreateAuthAzMap method creates an authentication and authorization directory mapping object.

Syntax

The CreateAuthAzMap method has the following format:

```
Netegrity::PolicyMgtSession->CreateAuthAzMap(authDir, azDir, mapType)
```

Parameters

The CreateAuthAzMap method accepts the following parameters:

authDir (PolicyMgtUserDir object)

Specifies the user directory to use when authenticating the user.

azDir (PolicyMgtUserDir object)

Specifies the user directory to use when authorizing the user.

mapType (int)

Specifies the type of directory mapping.

- AUTHAZMAPTYPE_DN (value = 1)

  Specifies mapping based on a DN.

- AUTHAZMAPTYPE_UNIVERSALID (value = 2)

  Specifies mapping based on a universal identifier.

- AUTHAZMAPTYPE_ATTR (value = 3)

  Specifies mapping based on an attribute in the user directory.

Return Value

The CreateAuthAzMap method returns one of the following values:

- PolicyMgtAuthAzMap (object)
- **undef** if the call is unsuccessful

Remarks

SiteMinder uses the same user directory to authenticate and authorize users. In addition, SiteMinder allows you to specify one user directory for authentication and another user directory for authorization. This feature is called directory mapping. Directory mapping is especially useful, when authentication information is stored in a central directory, but authorization information is stored in multiple directories, each one associated with a particular application.

CreateAuthScheme Method—Creates Authentication Scheme

The CreateAuthScheme method creates and configures an authentication scheme.

Syntax

The CreateAuthScheme method has the following format:

```
Netegrity::PolicyMgtSession->CreateAuthScheme(schemeName, schemeTemplate[,
schemeDesc][, protLevel][, schemeLib][, schemeParam][, secret][, isTemplate][,
isUsedByAdmin][, saveCreds][, isRadius][, ignorePwd])
```

Parameters

The CreateAuthScheme method accepts the following parameters:

schemeName (string)

Specifies the authentication scheme's name.

schemeTemplate (PolicyMgtAuthScheme object)

Specifies the template on which to base the authentication scheme.

Note: To view a list of templates, see the method PolicyMgtSession->GetAuthScheme (see page 481).

schemeDesc (string)

(Optional) Specifies the authentication scheme's description.

protLevel (int)

(Optional) Specifies the authentication scheme's protection level.

Range: 1-1000

Note: The higher the protection level value, the more secure the authentication scheme.

schemeLib (string)

(Optional) Specifies the name of the custom library to use in place of the default library shipped with each type of authentication scheme.

schemeParam (string)

(Optional) Specifies a parameter string to pass to the authentication scheme.

Note: For help constructing the parameter string, navigate to the Scheme Type Setup tab on the Authentication Scheme Properties dialog in the Administrative UI. Select the authentication scheme type, type the values in the fields, and observe the result on the Advanced tab.

secret (string)

(Optional) Specifies the authentication scheme's shared secret.

isTemplate (int)

(Optional) Specifies whether the authentication scheme is a template for other authentication schemes.

Default: A zero (0) value specifies that the authentication scheme is not a template.

Note: This parameter is deprecated as of CA SiteMinder v6.0 SP3.

isUsedByAdmin (int)

(Optional) Specifies whether the authentication scheme can be used to authenticate administrators.

saveCreds (int)

(Optional) Specifies whether to save user credentials.

isRadius (int)

(Optional) Specifies whether the authentication scheme type is RADIUS.

ignorePwd (int)

(Optional) Specifies whether to ignore password policies.

Return Value

The CreateAuthScheme method returns one of the following values:

- PolicyMgtAuthScheme (object)
- **undef** if the call is unsuccessful or the authentication scheme name already exists

CreateCustomCertMap Method—Creates Custom Certificate Map

The CreateCustomCertMap method creates a custom certificate map. The custom certificate map associates user attribute names defined in the certificate's Subject DN with the corresponding user attribute names in the user directory. For authentication to succeed, the values of the mapped user attribute pairs must match. Use the *AttributeMap* parameter to define the attribute names that are mapped.

Syntax

The CreateCustomCertMap method has the following format:

```
Netegrity::PolicyMgtSession->CreateCustomCertMap(IssuerDN, AttributeMap[,
DirectoryType])
```

Parameters

The CreateCustomCertMap method accepts the following parameters:

IssuerDN (string)

Specifies the certificate issuer's distinguished name.

AttributeMap (string)

Specifies an expression that maps attribute names in the certificate's Subject DN to attribute names in the user directory.

Syntax: UserAttrName1=%{CertAttrName1},UserAttrName2=%{CertAttrName2}, . . . UserAttrName#=%{CertAttrName#}

Example:

Certificate's Subject DN contains: CN=John Smith, UID=JSMITH, OU=Development, O=CompanyA

AttributeMap contains: CN=%{UID}, OU=%{OU}, O=%{O}

Matching user DN in the user directory: CN=JSMITH, OU=Development, O=CompanyA

DirectoryType (int)

(Optional) Specifies the type of user directory specified as the authentication directory:

- Sm_PolicyApi_DirType_LDAP

  Note: This is the default.

- Sm_PolicyApi_DirType_WinNT
- Sm_PolicyApi_DirType_ODBC

Return Value

The CreateCustomCertMap method returns one of the following values:

- PolicyMgtCertMap (object)
- **undef** if the call is unsuccessful

Remarks

When a certificate map is created, the following flags are set to false, the default value:

- certificate_required_flag
- use_distributionpoints_flag
- verify_signature_flag

- check_certificate_revocation_list_flag
- cache_certificate_revocation_list_entries_flag

For information on changing the value of these flags, see the method PolicyMgtSession->CreateExactCertMap (see page 426).

CreateDomain Method—Creates Policy Domain Object

The CreateDomain method creates a policy domain object.

Syntax

The CreateDomain method has the following format:

```
Netegrity::PolicyMgtSession->CreateDomain(domName[, domDesc][,
globalPoliciesApply])
```

Parameters

The CreateDomain method accepts the following parameters:

domName (string)

Specifies the name of the domain.

domDesc (string)

(Optional) Specifies the description of the domain.

globalPoliciesApply (int)

(Optional) Specifies whether the domain can accept global policies:

- value = 1 (default)

  Specifies that the domain can accept global policies.

- value = 0

  Specifies that the domain cannot accept global policies.

Return Value

The CreateDomain method returns one of the following values:

- PolicyMgtDomain (object)
- **undef** if the call is unsuccessful or the policy domain name already exists

CreateExactCertMap Method—Creates Certificate Map Matching User Directory Attributes

The CreateExactCertMap method creates a certificate map object whose Subject DN attributes match the corresponding user directory attributes exactly. When the certificate map object is created, the following flags are set to FALSE:

- Certificate required

  Note: To change the value of this flag, call the method PolicyMgtCertMap->CertRequired.

- Use distribution points

  Note: To change the value of this flag, call the method PolicyMgtCertMap->UseDistributionPoints.

- Verify signature

  Note: To change the value of this flag, call the method PolicyMgtCertMap->VerifySignature.

- Check Certificate Revocation List (CRL)

  Note: To change the value of this flag, call the method PolicyMgtCertMap->EnableCRL.

- Cache CRL entries

  Note: To change the value of this flag, call the method PolicyMgtCertMap->CacheCRL.

Syntax

The CreateExactCertMap method has the following format:

```
Netegrity::PolicyMgtSession->CreateExactCertMap(IssuerDN[, DirectoryType])
```

Parameters

The CreateExactCertMap method accepts the following parameters:

IssuerDN (string)

Specifies the distinguished name of the certificate issuer.

DirectoryType (int)

(Optional) Specifies one of the following user directory types used for authentication:

- Sm_PolicyApi_DirType_LDAP (default)
- Sm_PolicyApi_DirType_WinNT
- Sm_PolicyApi_DirType_ODBC

Return Value

The CreateExactCertMap method returns one of the following values:

- PolicyMgtCertMap (object)
- **undef** if the call is unsuccessful

CreateGlobalPolicy Method—Creates Global Policy

The CreateGlobalPolicy method creates a policy that has a global scope.

Syntax

The CreateGlobalPolicy method has the following format:

```
Netegrity::PolicyMgtSession->CreateGlobalPolicy(policyName[, enableFlag][,
activeExpr][, policyDesc])
```

Parameters

The CreateGlobalPolicy method accepts the following parameters:

policyName (string)

Specifies the global policy's name.

enableFlag (type)

(Optional) Specifies whether to enable the global policy:

- value = 1 (default)

  Specifies that the global policy is enabled.

- value = 0

  Specifies that the global policy is disabled.

activeExpr (string)

(Optional) Specifies ...

policyDesc (string)

(Optional) Specifies the global policy's description.

Return Value

The CreateGlobalPolicy method returns one of the following values:

- PolicyMgtPolicy (object)
- **undef** if the call is unsuccessful

CreateGlobalResponse Method—Creates Global Response

The CreateGlobalResponse method creates a response that has a global scope.

Syntax

The CreateGlobalResponse method has the following format:

```
Netegrity::PolicyMgtSession->CreateGlobalResponse(respName, agentType[, respDesc])
```

Parameters

The CreateGlobalResponse method accepts the following parameters:

respName (string)

Specifies the global response's name.

agentType (PolicyMgtAgentType object)

Specifies the type of agent associated with the global response.

Note: To retrieve the agent type object, call the method PolicyMgtSession->GetAgentType.

respDesc (string)

(Optional) Specifies the global response's description.

Return Value

The CreateGlobalResponse method returns one of the following values:

- PolicyMgtResponse (object)
- **undef** if the call is unsuccessful

CreateGlobalResponseGroup Method--Creates a Domain-Specific Rule Group

The CreateGlobalResponseGroup method creates a rule group that is specific to a particular domain.

Syntax

The CreateGlobalResponseGroup method has the following format:

```
Netegrity::PolicyMgtSession->CreateGlobalResponseGroup(groupName, agentType,
domain)
```

Parameters

The CreateGlobalResponseGroup method accepts the following parameters:

groupName (string)

Specifies the global rule group name.

agentType (PolicyMgtAgentType)

Specifies the type of agent.

domain (PolicyMgtDomain)

Specifies the domain for which the rule group applies.

Return Value

The CreateGlobalResponseGroup method returns one of the following values:

- PolicyMgtGroup (object)
- **undef** if the call is unsuccessful

CreateGlobalRule Method—Creates Global Rule

The CreateGlobalRule method creates a rule that has a global scope.

Syntax

The CreateGlobalRule method has the following format:

```
Netegrity::PolicyMgtSession->CreateGlobalRule(ruleName, resource, event, agent[,
ruleDesc][, allowAccess][, regexMatch][, activeExpr][, isEnabled])
```

Parameters

The CreateGlobalRule method accepts the following parameters:

ruleName (string)

Specifies the global rule's name.

resource (string)

Specifies the filter for the resource that the global rule is protecting.

event (string)

Specifies the type of event that the global rule is executing.

agent (PolicyMgtAgent | PolicyMgtGroup)

Specifies the agent or agent group associated with the global rule.

ruleDesc (string)

(Optional) Specifies the global rule's description.

allowAccess (int)

(Optional) Specifies whether to allow or deny access to the resource protected by the rule:

- value = 1 (default)

  Specifies allowing access.

- value = 0

  Specifies denying access.

regexMatch (int)

(Optional) Specifies whether to perform regular expression pattern matching:

- value = 1

  Specifies performing regular expression pattern matching.

- value = 0 (default)

  Specifies *not* performing regular expression pattern matching.

activeExpr (string)

(Optional) Specifies the global rule's active expression.

isEnabled (int)

(Optional) Specifies whether to enable or disable the global rule:

- value = 1 (default)

  Specifies that the global rule is enabled.

- value = 0

  Specifies that the global rule is disabled.

Return Value

The CreateGlobalRule method returns one of the following values:

- PolicyMgtRule (object)
- **undef** if the call is unsuccessful

CreateGlobalRuleGroup Method--Creates a Domain-Specific Rule Group

The CreateGlobalRuleGroup method creates a rule group that is specific to a particular domain.

Syntax

The CreateGlobalRuleGroup method has the following format:

```
Netegrity::PolicyMgtSession->CreateGlobalRuleGroup(groupName, agentType, domain)
```

Parameters

The CreateGlobalRuleGroup method accepts the following parameters:

groupName (string)

Specifies the global rule group name.

agentType (PolicyMgtAgentType)

Specifies the type of agent.

domain (PolicyMgtDomian)

Specifies the domain for which the rule group applies.

Return Value

The CreateGlobalRuleGroup method returns one of the following values:

- PolicyMgtGroup (object)
- **undef** if the call is unsuccessful

CreateHostConfig Method—Creates Host Configuration Object

The CreateHostConfig method creates a host configuration object.

Syntax

The CreateHostConfig method has the following format:

```
Netegrity::PolicyMgtSession->CreateHostConfig(hostConfigName[, hostConfDesc][,
enableFailover][, maxSocketsPerPort][, minSocketsPerPort][, newSocketstep][,
requestTimeout])
```

Parameters

The CreateHostConfig method accepts the following parameters:

hostConfigName (string)

Specifies the name of the host configuration object.

hostConfDesc (string)

(Optional) Specifies the description of the host configuration object.

enableFailover (int)

(Optional) Specifies whether to use failover or round-robin communication between the Policy Server and the agent:

- value = 1

  Specifies failover communication.

- value = 0

  Specifies round-robin communication.

maxSocketsPerPort (int)

(Optional) Specifies the maximum number of TCP/IP sockets that can be opened between an agent and the Policy Server.

minSocketsPerPort (int)

(Optional) Specifies the minimum number of TCP/IP sockets that can be opened between an agent and the Policy Server.

newSocketstep (int)

(Optional) Specifies how many sockets to open when additional sockets are required.

requestTimeout (int)

(Optional) Specifies how long, in seconds, an agent can wait for a response from the Policy Server.

Return Value

The CreateHostConfig method returns one of the following values:

- PolicyMgtHostConfig (object)
- **undef** if the call is unsuccessful or the host configuration name already exists

CreateODBCQueryScheme Method—Creates ODBC Query Scheme

The CreateODBCQueryScheme method creates and configures an ODBC query scheme. ODBC query schemes are also called SQL query schemes.

Note: Create a unique data source for each ODBC query scheme.

Syntax

The CreateODBCQueryScheme method has the following format:

```
Netegrity::PolicyMgtSession->CreateODBCQueryScheme(schemeName[, schemeDesc][,
queryEnumerate][, queryGetObjInfo][, queryLookup][, queryInitUser][,
queryAuthenticateUser][, queryGetUserProp][, querySetUserProp][,
queryGetUserProps][, queryLookupUser][, queryGetGroups][, queryIsGroupMember][,
queryGetGroupProp][, querySetGroupProp][, queryGetGroupProps][,
queryLookupGroup][, querySetPassword])
```

Parameters

The CreateODBCQueryScheme method accepts the following parameters:

schemeName (string)

Specifies the ODBC query scheme's name.

schemeDesc (string)

(Optional) Specifies the ODBC query scheme's description.

queryEnumerate (string)

(Optional) Specifies a query that lists the names of user objects in the directory.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QueryEnumerate (see page 289).

queryGetObjInfo (string)

(Optional) Specifies a query that fetches the object's class.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QueryGetObjInfo (see page 292).

queryLookup (string)

(Optional) Specifies a query that returns objects based on the value of an attribute in a group table.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QueryLookup (see page 296).

queryInitUser (string)

(Optional) Specifies a query that determines if a user with a given name exists in the database.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QueryInitUser (see page 295).

queryAuthenticateUser (string)

(Optional) Specifies a query that retrieves the user's password.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QueryAuthenticateUser (see page 288).

queryGetUserProp (string)

(Optional) Specifies a query that retrieves the value of a user property.

Note: The property must be listed in the *queryGetUserProps* parameter string. For more information, see the method PolicyMgtODBCQueryScheme->QueryGetUserProp (see page 293).

querySetUserProp (string)

(Optional) Specifies a query that sets the value of a user property.

Note: The property must be listed in the *queryGetUserProps* parameter string. For more information, see the method PolicyMgtODBCQueryScheme->QuerySetUserProp (see page 300).

queryGetUserProps (string)

(Optional) Specifies a comma-separated list of user attributes that reside in the same table as the user name.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QueryGetUserProps (see page 294).

queryLookupUser (string)

(Optional) Specifies a query that retrieves a user name through an attribute of the user table.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QueryLookupUser (see page 298).

queryGetGroups (string)

(Optional) Specifies a query that retrieves the names of the groups to which the user belongs.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QueryGetGroups (see page 291).

queryIsGroupMember (string)

(Optional) Specifies a query that determines whether a particular user is a member of a group.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QueryIsGroupMember (see page 295).

queryGetGroupProp (string)

(Optional) Specifies a query that returns the value of a group property.

Note: The property must be listed in the *queryGetGroupProps* parameter string. For more information, see the method PolicyMgtODBCQueryScheme->QueryGetGroupProp (see page 290).

querySetGroupProp (string)

(Optional) Specifies a query that sets the value of a group property.

Note: The property must be listed in the *queryGetGroupProps* parameter string. For more information, see the method PolicyMgtODBCQueryScheme->QuerySetGroupProp (see page 298).

queryGetGroupProps (string)

(Optional) Specifies a comma-separated list of group attributes.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QueryGetGroupProps (see page 291).

queryLookupGroup (string)

(Optional) Specifies a query that retrieves a group name through an attribute of the group table.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QueryLookupGroup (see page 297).

querySetPassword (string)

(Optional) Specifies a query that changes a user's password.

Note: For more information, see the method PolicyMgtODBCQueryScheme->QuerySetPassword (see page 299).

Return Value

The CreateODBCQueryScheme method returns one of the following values:

- PolicyMgtODBCQueryScheme (object)
- **undef** if the call is unsuccessful or the ODBC query scheme name already exists

CreatePwdPolicy Method—Creates Password Policy

The CreatePwdPolicy method creates and configures a password policy.

Syntax

The CreatePwdPolicy method has the following format:

```
Negetrity::PolicyMgtSession->CreatePwdPolicy(pwdPolName, userDir[, pwdPolDesc][,
enabledFlag][, entireDirFlag][, path][, class][, allowNestedGroups][,
maxLoginFailures][, maxLoginInactive][, expDelay][, expWarningDays][, dicName][,
dicMatchLength][, userwait][, pwdSvcRedirect][maxPwdLength][, minPwdLength][,
maxPwdRepeatChar][, minPwdAlphaNum][, minPwdAlpha][, minPwdNonAlpha][,
minPwdNonPrint][, minPwdNum][, minPwdPunc][, pwdReuseCount][, pwdReuseDelay][,
pwdPctDiff][, pwdIgnoreSeq][, profileAttrMatch])
```

Parameters

The CreatePwdPolicy method accepts the following parameters:

pwdPolName (string)

Specifies the name of the password policy.

userDir (PolicyMgtUserDir object)

Specifies the user directory to which the password policy applies.

pwdPolDesc (string)

(Optional) Specifies the description of the password policy.

enabledFlag (int)

(Optional) Specifies whether the password policy is enabled.

entireDirFlag (int)

(Optional) Specifies whether the password policy applies to the entire LDAP directory or only part of the directory.

- value = 1

  Specifies that the password policy applies to the entire LDAP directory.

- value = 0

  Specifies that the password policy only applies to part of the LDAP directory.

Note: For part of the LDAP directory, specify the directory path in the *path* parameter and the class in the *class* parameter.

path (string)

(Optional) Specifies the part of the directory to which the password policy applies.

Note: Include this parameter when the *entireDirFlag* parameter is set to 0.

class (string)

(Optional) Specifies the class to which the password policy applies.

Note: Include this parameter when the *entireDirFlag* parameter is set to 0.

allowNestedGroups (int)

(Optional) Specifies whether the password policy is associated with the nested groups in the LDAP directory.

Note: Include this parameter when the *entireDirFlag* parameter is set to 0.

maxLoginFailures (int)

(Optional) Specifies the maximum number of login failures allowed before the user's account is disabled.

maxLoginInactive (int)

(Optional) Specifies the maximum number of days of inactivity allowed before the user's password expires.

expDelay (int)

(Optional) Specifies the number of days a password can be unchanged before it expires.

expWarningDays (int)

(Optional) Specifies the number of days in advance to notify the user that the password is due to expire.

dicName (string)

(Optional) Specifies the location of the dictionary file that lists the words that cannot be used in a password.

dicMatchLength (int)

(Optional) Specifies the minimum number of letters required for dictionary checking.

userwait (int)

(Optional) Specifies the number of minutes an account is disabled before the account is enabled and the user is allowed to attempt logging in again.

pwdSvcRedirect (string)

(Optional) Specifies the URL where the user is redirected when an invalid password is entered.

Note: This must be the URL of the Password Services CGI.

maxPwdLength (int)

(Optional) Specifies the maximum length of a user password.

Note: This value must be greater than the value specified by the parameter *minPwdLength*.

minPwdLength (int)

(Optional) Specifies the minimum length of a user password.

maxPwdRepeatChar (int)

(Optional) Specifies the maximum number of identical characters that can appear consecutively in a password.

minPwdAlphaNum (int)

(Optional) Specifies the minimum number of alphanumeric characters (A-Z, a-z, 0-9) that a password must contain.

minPwdAlpha (int)

(Optional) Specifies the minimum number of alphabetic characters (A-Z, a-z) that a password must contain.

minPwdNonAlpha (int)

(Optional) Specifies the minimum number of non-alphanumeric characters that a password must contain.

Note: The following are examples of non-alphanumeric characters: "@", "$", and "*".

minPwdNonPrint (int)

(Optional) Specifies the minimum number of non-printable characters that a password must contain.

Note: Non-printable characters are not displayed on a computer screen.

minPwdNum (int)

(Optional) Specifies the minimum number of numeric characters (0-9) that a password must contain.

minPwdPunc (int)

(Optional) Specifies the minimum number of punctuation marks that a password must contain.

Note: Punctuation marks include periods, commas, exclamation points, slashes, hyphens, and dashes.

pwdReuseCount (int)

(Optional) Specifies the number of new passwords that must be used before an old one can be reused.

pwdReuseDelay (int)

(Optional) Specifies the number of days a user must wait before reusing a password.

pwdPctDiff (int)

(Optional) Specifies the percentage of characters contained in a new password that must differ from the characters in the previous password.

Note: A value of 100 specifies that the new password cannot contain any of the characters in the previous password. For more information, see the parameter *pwdIgnoreSeq*.

pwdIgnoreSeq (int)

(Optional) Specifies whether character position is ignored when the new password is compared to the previous password and the percentage of characters that are different is calculated.

- value = 1

  Specifies that character sequence is ignored.

- value = 0

  Specifies that character sequence is considered.

  Example: If the character "c" is in both the new and previous passwords, but its position in each password is different, then it is considered to be two different characters when the percentage is calculated.

profileAttrMatch (int)

(Optional) Specifies that the minimum character sequence that SiteMinder checks when checking the password against attributes in the user's directory entry.

Return Value

The CreatePwdPolicy method returns one of the following values:

- PolicyMgtPwdPolicy (object)
- **undef** if the call is unsuccessful or the password policy name already exists

CreateRegScheme Method—Creates Registration Scheme

The CreateRegScheme method creates and configures a registration scheme.

Syntax

The CreateRegScheme method has the following format:

```
Netegrity::PolicyMgtSession->CreateRegScheme(regName, userDir[, regDesc][,
welcomeURL][, templatePath][, enableLogging])
```

Parameters

The CreateRegScheme method accepts the following parameters:

regName (string)

Specifies the registration scheme's name.

userDir (string)

Specifies the user directory associated with the registration scheme.

regDesc (string)

(Optional) Specifies the registration scheme's description.

welcomeURL (string)

(Optional) Specifies the URL for the welcome page.

Note: Users are redirected to this page after successfully registering.

Example: http://my.acme.com/hr/welcome.htm

templatePath (string)

(Optional) Specifies the path where the registration templates are located.

Note: For more information about the *templePath* parameter, see Remarks.

enableLogging (int)

(Optional) Specifies whether to enable logging.

- value = 1

  Specifies enabling logging.

- value = 0 (default)

  Specifies disabling logging.

Return Value

The CreateRegScheme method returns one of the following values:

- PolicyMgtRegScheme (object)
- **undef** if the call is unsuccessful or the registration scheme name already exists

Remarks

When you install a SiteMinder Web Agent, the registration templates are installed in the samples/selfreg subdirectory of the Web Agent installation directory by default. During SiteMinder installation, the virtual directory /siteminderagent is created and pointed to the samples directory in the Web Agent installation directory. Therefore, when using the default directory, specify *templePath* as follows: /siteminderagent/selfreg (without the final slash).

If you are using SSL for registration, you must provide the absolute path for the registration templates. The default paths are as follows:

- Windows platforms: install-dir\Netegrity\Siteminder Web Agent\Samples\SelfReg\
- Solaris/Hpux platforms: install-dir/netegrity/siteminder/webagent/samples/selfreg/

CreateSAMLAffiliation Method—Creates SAML 2.0 Affiliation Object

The CreateSAMLAffiliation method creates a SAML 2.0 affiliation object. A SAML 2.0 affiliation is a set of entities that share a single federated namespace of unique Name IDs for principals. To modify the properties of an existing SAML affiliation, call the method PolicyMgtSAMLAffiliation->Property.

Syntax

The CreateSAMLAffiliation method has the following format:

```
Netegrity::PolicyMgtSession->CreateSAMLAffiliation(propsHash_ref)
```

Parameters

The CreateSAMLAffiliation method accepts the following parameter:

propsHash_ref (hash)

Specifies a reference to a hashtable of metadata properties for the SAML 2.0 affiliation.

Example: \%myhash

Return Value

The CreateSAMLAffiliation method returns one of the following values:

- PolicyMgtSAMLAffiliation (object)
- **undef** if the call is unsuccessful

Remarks

The SAML 2.0 affiliation properties are grouped as follows:

General Properties

SAML_NAME

SAML_DESCRIPTION

SAML_KEY_AFFILIATION_ID

SAML_MAJOR_VERSION

SAML_MINOR_VERSION

SAML_OID

Name ID Properties

SAML_SP_NAMEID_FORMAT

SAML_SP_NAMEID_TYPE

SAML_SP_NAMEID_STATIC

SAML_SP_NAMEID_ATTRNAME

SAML_SP_NAMEID_DNSPEC

User Properties

SAML_IDP_XPATH

SAML_IDP_LDAP_SEARCH_SPEC

SAML_IDP_ODBC_SEARCH_SPEC

SAML_IDP_WINNT_SEARCH_SPEC

SAML_IDP_CUSTOM_SEARCH_SPEC

SAML_IDP_AD_SEARCH_SPEC

For more information, see the SAML 2.0 Property Reference in this guide.

CreateSAMLAuthScheme Method—Creates SAML Authentication Scheme Object

The CreateSAMLAuthScheme method creates a SAML 2.0 authentication scheme object with its properties set to specified values. There are two types of properties associated with the object: authentication scheme properties and metadata properties.

Authentication Scheme Properties

The authentication scheme properties are based on the SAML 2.0 template and have the following initial values:

- Library

  Default: smauthsaml

- Is_Template

  Default: FALSE

- Is_Used_by_Administrator

  Default: FALSE

- Save_Credentials

  Default: FALSE

- Is_RADIUS

  Default: FALSE

- Ignore_Password_Check

  Default: TRUE

- Protection_Level

  Default: 5

Note: You can modify the default protection level by calling the CreateSAMLAuthScheme method with the optional *protLevel* parameter set to a new value.

Metadata Properties

The metadata properties are the properties of the Identity Provider associated with the SAML 2.0 authentication scheme and are stored with the authentication scheme. To specify them, pass the reference to the hashtable of metadata properties to the CreateSAMLAuthScheme method in the *propsHash_ref* parameter. To update the metadata properties of an existing SAML 2.0 authentication scheme, call the method PolicyMgtSession->SAMLAuthSchemeProperties.

Syntax

The CreateSAMLAuthScheme method has the following format:

```
Netegrity::PolicyMgtSession->CreateSAMLAuthScheme(schemeName, propsHash_ref[,
schemeDesc][, protLevel])
```

Parameters

The CreateSAMLAuthScheme method accepts the following parameters:

schemeName (string)

  Specifies the name of the authentication scheme.

propsHash_ref (hash)

  Specifies a reference to a hashtable of metadata properties to associate with the SAML 2.0 authentication scheme.

  Example: \%myhash

  Note: For a complete list of metadata properties, see Remarks.

schemeDesc (string)

  (Optional) Specifies the description of the authentication scheme.

protLevel (int)

  (Optional) Specifies the protection level of the authentication scheme.

Return Value

The CreateSAMLAuthScheme method returns one of the following values:

- PolicyMgtAuthScheme (object)
- **undef** if the call is unsuccessful or the SAML authentication scheme name already exists

Remarks

The metadata properties associated with the SAML 2.0 authentication scheme are listed following.

General Properties

SAML_NAME

SAML_DESCRIPTION

Scheme Setup Properties

SAML_IDP_SPID

SAML_KEY_IDPID

SAML_MAJOR_VERSION

SAML_MINOR_VERSION

SAML_SKEWTIME

SAML_DISABLE_SIGNATURE_PROCESSING

SAML_DSIG_VERINFO_ISSUER_DN

SAML_DSIG_VERINFO_SERIAL_NUMBER

User Properties

SAML_IDP_XPATH

SAML_IDP_LDAP_SEARCH_SPEC

SAML_IDP_ODBC_SEARCH_SPEC

SAML_IDP_WINNT_SEARCH_SPEC

SAML_IDP_CUSTOM_SEARCH_SPEC

SAML_IDP_AD_SEARCH_SPEC

SAML_AFFILIATION

SSO Properties

SAML_IDP_SSO_REDIRECT_MODE

SAML_IDP_SSO_DEFAULT_SERVICE

SAML_AUDIENCE

SAML_IDP_SSO_TARGET

SAML_ENABLE_SSO_ARTIFACT_BINDING

SAML_KEY_IDP_SOURCEID

SAML_IDP_ARTIFACT_RESOLUTION_DEFAULT_SERVICE

SAML_IDP_BACKCHANNEL_AUTH_TYPE

SAML_IDP_SPNAME

SAML_IDP_PASSWORD

SAML_ENABLE_SSO_POST_BINDING

SAML_IDP_SSO_ENFORCE_SINGLE_USE_POLICY

SAML_SSOECPPROFILE

SAML_IDP_SIGN_AUTHNREQUESTS

SLO Properties

SAML_SLO_REDIRECT_BINDING

SAML_SLO_SERVICE_VALIDITY_DURATION

SAML_SLO_SERVICE_URL

SAML_SLO_SERVICE_RESPONSE_URL

SAML_SLO_SERVICE_CONFIRM_URL

Encryption Properties

SAML_IDP_REQUIRE_ENCRYPTED_ASSERTION

SAML_IDP_REQUIRE_ENCRYPTED_NAMEID

Attribute Properties

SAML_IDP_SAMLREQ_ENABLE

SAML_IDP_SAMLREQ_REQUIRE_SIGNED_ASSERTION

SAML_IDP_SAMLREQ_ATTRIBUTE_SERVICE

SAML_IDP_SAMLREQ_GET_ALL_ATTRIBUTES

NameID Properties

SAML_IDP_SAMLREQ_NAMEID_FORMAT

SAML_IDP_SAMLREQ_NAMEID_TYPE

SAML_IDP_SAMLREQ_NAMEID_STATIC

SAML_IDP_SAMLREQ_NAMEID_ATTR_NAME

SAML_IDP_SAMLREQ_NAMEID_DN_SPEC

SAML_IDP_SAMLREQ_NAMEID_ALLOW_NESTED

Advanced Properties

SAML_SP_PLUGIN_CLASS

SAML_SP_PLUGIN_PARAMS

SAML_IDP_REDIRECT_URL_USER_NOT_FOUND

SAML_IDP_REDIRECT_MODE_USER_NOT_FOUND

SAML_IDP_REDIRECT_URL_FAILURE

SAML_IDP_REDIRECT_MODE_FAILURE

SAML_IDP_REDIRECT_URL_INVALID

SAML_IDP_REDIRECT_MODE_INVALID

CreateSingleCertMap Method—Creates Single-Attribute Certificate Map

The CreateSingleCertMap method creates a certificate map between a single attribute in the certificate's Subject DN and the corresponding user attribute in the user directory. For authentication to succeed, the attribute's value in the Subject DN must match the value of the corresponding user attribute in the user directory.

Syntax

The CreateSingleCertMap Method method has the following format:

```
Netegrity::PolicyMgtSession->CreateSingleCertMap(IssuerDN, Attribute[,
DirectoryType])
```

Parameters

The CreateSingleCertMap Method method accepts the following parameters:

IssuerDN (string)

Specifies the distinguished name of the certificate issuer.

Attribute (string)

Specifies the name of the attribute whose values in the certificate's Subject DN and in the user directory must match.

Syntax: %{*attribute_name*}

Example: %{uid}

DirectoryType (int)

(Optional) Specifies the type of the user directory specified for authentication.

- Sm_PolicyApi_DirType_LDAP (default)
- Sm_PolicyApi_DirType_WinNT
- Sm_PolicyApi_DirType_ODBC

Return Value

The CreateSingleCertMap Method method returns one of the following values:

- PolicyMgtCertMap (object)
- **undef** if the call is unsuccessful

Remarks

When a certificate map is created, the following flags are set to false, the default value:

- certificate_required_flag
- use_distributionpoints_flag
- verify_signature_flag
- check_certificate_revocation_list_flag
- cache_certificate_revocation_list_entries_flag

For information on changing the value of these flags, see the method PolicyMgtSession->CreateExactCertMap (see page 426).

CreateTrustedHost Method—Creates Trusted Host Object

The CreateSAMLAuthScheme method creates a trusted host object in the policy store, registers the trusted host with the Policy Server, and if registration is successful, creates the local registration file. Use this method when the Policy Server is connected to the agent host. When there is no connection between the Policy Server and the agent host, call the method PolicyMgtSession->AddTrustedHost instead.

Syntax

The CreateTrustedHost method has the following format:

```
Netegrity::PolicyMgtSession->CreateTrustedHost(trustedHostName [,ipAddress][,
adminName][, adminPassword][, hostConfigName][, registrationDataFileName])
```

Parameters

The CreateTrustedHost method accepts the following parameters:

trustedHostName (string)

Specifies the name of the trusted host.

ipAddress (string)

(Optional) Specifies the IP address of the Policy Server.

adminName (string)

(Optional) Specifies the name of a Policy Server administrator.

adminPassword (string)

(Optional) Specifies the administrator's password.

hostConfigName (string)

(Optional) Specifies the name of the host configuration object.

registrationDataFileName (string)

(Optional) Specifies the name of the file where registration data is written when the host is successfully registered with the Policy Server.

Note: This filename is specified by calling the Agent API method Connect. The file is stored and managed by SiteMinder.

Return Value

The CreateTrustedHost method returns one of the following values:

- PolicyMgtTrustedHost (object)
- **undef** if the call is unsuccessful or if the trusted host name already exists

CreateUserDir Method—Creates User Directory Object

The CreateUserDir method creates and configures a user directory object.

Syntax

The CreateUserDir method has the following format:

```
Netegrity::PolicyMgtSession->CreateUserDir(dirName, namespace, server[,
ODBCQueryScheme][, domDesc][, searchRoot][, usrLookStart][, usrLookEnd][,
username][, password][, searchResults][, searchScope][, searchTimeout][,
secureConn][, requireCreds][, disabledAttr][, UIDAttr][, anonID][, pwdData][,
pwdAttr][, emailAttr][, chalRespAttr])
```

Parameters

The CreateUserDir method accepts the following parameters:

dirName (string)

Specifies the user directory object's name.

namespace (string)

Specifies the user directory's namespace:

- LDAP
- AD
- ODBC
- WinNT
- Custom

server (string)

Specifies one of the following directory-dependent values:

LDAP and AD

Specifies the IP address and port number of the LDAP server.

Syntax: IP_address:port_number

Note: The default port number is 389.

ODBC

Specifies the data source name.

WinNT

Specifies the domain name.

Custom

Specifies the name of the library that corresponds to the custom directory.

ODBCQueryScheme (PolicyMgtODBCQueryScheme object)

(Optional) Specifies a set of queries that SiteMinder uses to query the ODBC directory.

Note: If the user directory is not an ODBC directory, this parameter's value is undef.

domDesc (string)

(Optional) Specifies the description of the user directory.

searchRoot (string)

(Optional) Specifies one of the following directory-dependent values:

LDAP

Specifies the location in the LDAP tree that is the starting point for the directory connection, for example, the organization (o) or organizational unit (ou). This location, called the search root, is the point where the Policy Server starts the search for a user.

Note: For more information about this parameter, see the parameter *searchScope*.

Custom

Specifies a string of parameters to pass to the custom library.

usrLookStart (string)

(Optional) Specifies the start value for a user DN lookup in an LDAP directory.

usrLookEnd (string)

(Optional) Specifies the end value for a user DN lookup in an LDAP directory.

Note: Specifying values for the user DN lookup starting point and endpoint allows users to enter part of the DN string when authenticating. In the following example, the user only needs to specify the string "JSmith" and not the whole DN string when logging in:

- DN = "uid=JSmith,ou=marketing,o=myorg.org"
- starting_point = "uid="
- endpoint = ",ou=marketing,o=myorg.org"
- login = "JSmith"

username (string)

(Optional) Specifies the user name needed for accessing the user directory.

Note: When using this parameter, set *requireCreds* to 1.

password (string)

(Optional) Specifies the password required for accessing the user directory.

Note: When using this parameter, set *requireCreds* to 1.

searchResults (int)

(Optional) Specifies the maximum number of results to return from a search of an LDAP or custom directory.

searchScope (int)

(Optional) Specifies how many levels SiteMinder searches when looking for users or user groups in an LDAP directory:

- USERDIR_SCOPE_SUBTREE

  Specifies searching the root and all levels below.

- USERDIR_SCOPE_ONELEVEL

  Specifies searching the root and one level below.

Note: For more information, see the *searchRoot* parameter.

searchTimeout (int)

(Optional) Specifies the maximum time, in seconds, allowed for searching an LDAP or custom directory.

secureConn (int)

(Optional) Specifies whether an LDAP or custom user directory connection is secured by SSL:

- value = 1

  Specifies a connection secured by SSL.

- value = 0 (default)

  Specifies a connection that is not secure.

Note: When this flag is enabled, SiteMinder authentication is secure and transmissions are encrypted. Enable this flag when using SSL.

requireCreds (int)

(Optional) Specifies whether user credentials are required for authentication:

- value = 1

  Specifies that credentials are required.

- value = 0 (default)

  Specifies that credentials are not required.

disabledAttr (string)

(Optional) Specifies the name of the user directory attribute that contains the user's disabled state.

Note: This parameter applies to LDAP and ODBC directories and some custom directories.

UIDAttr (string)

(Optional) Specifies the name of the user directory's universal ID attribute.

Note: The universal ID is different from the user's login ID and is used to look up user information. This parameter applies to LDAP, ODBC, and WinNT directories and to some custom directories.

anonID (string)

(Optional) Specifies the name of the user directory's anonymous user DN attribute.

Note: The DN, which is defined in the anonymous authentication scheme, gives anonymous users access to resources protected by the anonymous authentication scheme. This parameter applies to LDAP directories and some custom directories.

pwdData (string)

(Optional) Specifies the name of the user directory's password data attribute.

Note: This parameter applies to LDAP and ODBC directories and some custom directories.

pwdAttr (string)

(Optional) Specifies the name of the user directory's password attribute.

Note: This parameter applies to LDAP and ODBC directories and some custom directories.

emailAttr (string)

Note: This optional parameter is reserved for future use.

chalRespAttr (string)

(Optional) Specifies the name of the user directory's challenge/response attribute.

Example: The challenge/response can be a hint that SiteMinder sends the user when the user forgets the password.

Note: This parameter applies to LDAP directories and some custom directories.

Return Value

The CreateUserDir method returns one of the following values:

- PolicyMgtUserDir (object)
- **undef** if the call is unsuccessful

CreateWSFEDAuthScheme Method—Creates WS-Federation Authentication Scheme

The CreateWSFEDAuthScheme method creates an instance of a WS-Federation authentication scheme and sets the authentication scheme's properties.

Syntax

The CreateWSFEDAuthScheme method has the following format:

```
Netegrity::PolicyMgtSession->CreateWSFEDAuthScheme(name, propsHash[, desc][,
level])
```

Parameters

The CreateWSFEDAuthScheme method accepts the following parameters:

name (string)

Specifies the name of the WS-Federation authentication scheme.

propsHash (hashtable)

Specifies a reference to the hashtable of WS-Federation authentication scheme properties to set.

Note: For a complete list of WS-Federation authentication scheme properties, see Remarks.

desc (string)

(Optional) Specifies a description of the authentication scheme.

level (int)

(Optional) Specifies the authentication scheme level.

Return Value

The CreateWSFEDAuthScheme method returns one of the following values:

- PolicyMgtAuthScheme (object)
- **undef** if the call is unsuccessful

Remarks

The WS-Federation authentication scheme properties are grouped in the FSS Administrative UI as follows:

General Properties

WSFED_NAME

WSFED_DESCRIPTION

Scheme Setup Tab

WSFED_KEY_APID

WSFED_RPID

WSFED_SKEW_TIME

WSFED_DISABLE_SIGNATURE_PROCESSING

WSFED_DSIG_VERINFO_ALIAS

Additional Configuration, Users Tab

WSFED_AP_XPATH

WSFED_AP_LDAP_SEARCH_SPEC

WSFED_AP_ODBC_SEARCH_SPEC

WSFED_AP_WINNT_SEARCH_SPEC

WSFED_AP_CUSTOM_SEARCH_SPEC

WSFED_AP_ADD_SEARCH_SPEC

Additional Configuration, SSO Tab

WSFED_AP_SSO_REDIRECT_MODE

WSFED_AP_SSO_DEFAULT_SERVICE

WSFED_AP_SSO_TARGET

WSFED_ENFORCE_SINGLE_USE_POLICY

Additional Configuration, Signout Tab

WSFED_AP_SLO_ENABLED

WSFED_AP_SIGNOUT_URL

Additional Configuration, Advanced Tab

WSFED_AP_PLUGIN_CLASS

WSFED_AP_PLUGIN_PARAMS

WSFED_AP_USER_NOT_FOUND_REDIRECT_URL

WSFED_AP_USER_NOT_FOUND_REDIRECT_MODE

WSFED_AP_FAILURE_REDIRECT_URL

WSFED_AP_FAILURE_REDIRECT_MODE

WSFED_AP_INVALID_REDIRECT_URL

WSFED_AP_INVALID_REDIRECT_MODE

DeleteAdmin Method—Deletes Administrator

The DeleteAdmin method deletes an administrator from the policy store.

Syntax

The DeleteAdmin method has the following format:

```
Netegrity::PolicyMgtSession->DeleteAdmin(admin)
```

Parameters

The DeleteAdmin method accepts the following parameter:

admin (PolicyMgtAdmin object)

Specifies the administrator object to delete.

Return Value

The DeleteAdmin method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the administrator is not found.
- value = -1

  Specifies that the method is unsuccessful.

Remarks

To remove an administrator from a particular domain, see the method PolicyMgtAffDomain->RemoveAdmin (see page 174).

DeleteAffDomain Method—Deletes Affiliate Domain

The DeleteAffDomain method deletes an affiliate domain.

Syntax

The DeleteAffDomain method has the following format:

```
Netegrity::PolicyMgtSession->DeleteAffDomain(affDomain)
```

Parameters

The DeleteAffDomain method accepts the following parameter:

affDomain (PolicyMgtAffDomain object)

Specifies the affiliate domain object to delete.

Return Value

The DeleteAffDomain method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the affiliate domain is not found.

- value = -1

  Specifies that the method is unsuccessful.

DeleteAgent Method—Deletes Agent

The DeleteAgent method deletes an agent.

Syntax

The DeleteAgent method has the following format:

```
Netegrity::PolicyMgtSession->DeleteAgent(agent)
```

Parameters

The DeleteAgent method accepts the following parameter:

agent (PolicyMgtAgent object)

Specifies the agent object to delete.

Return Value

The DeleteAgent method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the agent is not found.

- value = -1

  Specifies that the method is unsuccessful.

DeleteAgentConfig Method—Deletes Agent Configuration Object

The DeleteAgentConfig method deletes an agent configuration object.

Syntax

The DeleteAgentConfig method has the following format:

```
Netegrity::PolicyMgtSession->DeleteAgentConfig(AgentConfig)
```

Parameters

The DeleteAgentConfig method accepts the following parameter:

AgentConfig (PolicyMgtAgentConfig object)

Specifies the agent configuration object to delete.

Return Value

The DeleteAgentConfig method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the agent configuration object was not found.

- value = -1

  Specifies that the method is unsuccessful.

DeleteAuthAzMap Method—Deletes Authentication and Authorization Map

The DeleteAuthAzMap method deletes an authentication and authorization map.

Syntax

The DeleteAuthAzMap method has the following format:

```
Netegrity::PolicyMgtSession->DeleteAuthAzMap(map)
```

Parameters

The DeleteAuthAzMap method accepts the following parameter:

map (PolicyMgtAuthAzMap object)

Specifies the authentication and authorization map object to delete.

Return Value

The DeleteAuthAzMap method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the authentication and authorization map is not found.

- value = -1

  Specifies that the method is unsuccessful.

DeleteAuthScheme Method—Deletes Authentication Scheme

The DeleteAuthScheme method deletes an authentication scheme.

Syntax

The DeleteAuthScheme method has the following format:

```
Netegrity::PolicyMgtSession->DeleteAuthScheme(authScheme)
```

Parameters

The DeleteAuthScheme method accepts the following parameter:

authScheme (PolicyMgtAuthScheme object)

  Specifies the authentication scheme object to delete.

Return Value

The DeleteAuthScheme method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the authentication scheme is not found.

- value = -1

  Specifies that the method is unsuccessful.

DeleteCertMap Method—Deletes Certificate Map

The DeleteCertMap method deletes a certificate map.

Syntax

The DeleteCertMap method has the following format:

```
Netegrity::PolicyMgtSession->DeleteCertMap(map)
```

Parameters

The DeleteCertMap method accepts the following parameter:

map (PolicyMgtCertMap object)

Specifies the certificate map object to delete.

Return Value

The DeleteCertMap method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the certificate map is not found.

- value = -1

  Specifies that the method is unsuccessful.

DeleteDomain Method—Deletes Policy Domain

The DeleteDomain method deletes a policy domain.

Syntax

The DeleteDomain method has the following format:

```
Netegrity::PolicyMgtSession->DeleteDomain(domain)
```

Parameters

The DeleteDomain method accepts the following parameter:

domain (PolicyMgtDomain object)

Specifies the domain object to delete.

Return Value

The DeleteDomain method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the domain is not found.
- value = -1

  Specifies that the method is unsuccessful.

DeleteGlobalPolicy Method—Deletes Global Policy

The DeleteGlobalPolicy method deletes a global policy.

Syntax

The DeleteGlobalPolicy method has the following format:

```
Netegrity::PolicyMgtSession->DeleteGlobalPolicy(policy)
```

Parameters

The DeleteGlobalPolicy method accepts the following parameter:

policy (PolicyMgtPolicy object)

  Specifies the global policy object to delete.

Return Value

The DeleteGlobalPolicy method returns one of the following values:

- value = 0

  Specifies that the method is successful.
- value = -1

  Specifies that the method is unsuccessful.

DeleteGlobalResponse Method—Deletes Global Response

The DeleteGlobalResponse method deletes a global response.

Syntax

The DeleteGlobalResponse method has the following format:

```
Netegrity::PolicyMgtSession->DeleteGlobalResponse(response)
```

Parameters

The DeleteGlobalResponse method accepts the following parameter:

response (PolicyMgtResponse object)

Specifies the global response object to delete.

Return Value

The DeleteGlobalResponse method returns one of the following values:

- value = 0

  Specifies that the method is successful.

- value = -1

  Specifies that the method is unsuccessful.

DeleteGlobalRule Method—Deletes Global Rule

The DeleteGlobalRule method deletes a global rule.

Syntax

The DeleteGlobalRule method has the following format:

```
Netegrity::PolicyMgtSession->DeleteGlobalRule(rule)
```

Parameters

The DeleteGlobalRule method accepts the following parameter:

rule (PolicyMgtRule object)

Specifies the global rule object to delete.

Return Value

The DeleteGlobalRule method returns one of the following values:

- value = 0

  Specifies that the method is successful.

- value = -1

  Specifies that the method is unsuccessful.

DeleteGroup Method—Deletes Agent Group

The DeleteGroup method deletes an agent group.

Syntax

The DeleteGroup method has the following format:

```
Netegrity::PolicyMgtSession->DeleteGroup(group)
```

Parameters

The DeleteGroup method accepts the following parameter:

group (PolicyMgtGroup object)

Specifies the agent group object to delete.

Return Value

The DeleteGroup method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the agent group is not found.
- **undef**

  Specifies that the method is unsuccessful.

DeleteHostConfig Method—Deletes Host Configuration Object

The DeleteHostConfig method deletes a host configuration object.

Syntax

The DeleteHostConfig method has the following format:

```
Netegrity::PolicyMgtSession->DeleteHostConfig(HostConfig)
```

Parameters

The DeleteHostConfig method accepts the following parameter:

HostConfig (PolicyMgtHostConfig object)

Specifies the host configuration object to delete.

Return Value

The DeleteHostConfig method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the host configuration object is not found.

- value = -1

  Specifies that the method is unsuccessful.

DeleteODBCQueryScheme Method—Deletes ODBC Query Scheme

The DeleteODBCQueryScheme method deletes an ODBC query scheme.

Syntax

The DeleteODBCQueryScheme method has the following format:

```
Netegrity::PolicyMgtSession->DeleteODBCQueryScheme(scheme)
```

Parameters

The DeleteODBCQueryScheme method accepts the following parameter:

scheme (PolicyMgtODBCQueryScheme object)

Specifies the ODBC query scheme object to delete.

Return Value

The DeleteODBCQueryScheme method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the ODBC query scheme is not found.

- value = -1

  Specifies that the method is unsuccessful.

DeletePwdPolicy Method—Deletes Password Policy

The DeletePwdPolicy method deletes a password policy.

Syntax

The DeletePwdPolicy method has the following format:

```
Netegrity::PolicyMgtSession->DeletePwdPolicy(pwdPolicy)
```

Parameters

The DeletePwdPolicy method accepts the following parameter:

pwdPolicy (PolicyMgtPwdPolicy object)

Specifies the password policy object to delete.

Return Value

The DeletePwdPolicy method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the password policy is not found.

- value = -1

  Specifies that the method is unsuccessful.

DeleteRegScheme Method—Deletes Registration Scheme

The DeleteRegScheme method deletes a registration scheme.

Syntax

The DeleteRegScheme method has the following format:

```
Netegrity::PolicyMgtSession->DeleteRegScheme(regScheme)
```

Parameters

The DeleteRegScheme method accepts the following parameter:

regScheme (PolicyMgtRegScheme object)

Specifies the registration scheme object to delete.

Return Value

The DeleteRegScheme method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the registration scheme is not found.

- value = -1

  Specifies that the method is unsuccessful.

DeleteSAMLAffiliation Method—Deletes SAML Affiliation

The DeleteSAMLAffiliation method deletes a SAML 2.0 affiliation object.

Syntax

The DeleteSAMLAffiliation method has the following format:

```
Netegrity::PolicyMgtSession->DeleteSAMLAffiliation(SAMLAffil)
```

Parameters

The DeleteSAMLAffiliation method accepts the following parameter:

SAMLAffil (PolicyMgtSAMLAffiliation object)

Specifies the SAML 2.0 affiliation object to delete.

Return Value

The DeleteSAMLAffiliation method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the SAML affiliation object is not found.

- value = -1

  Specifies that the method is unsuccessful.

DeleteTrustedHost Method—Deletes Trusted Host

The DeleteTrustedHost method deletes a trusted host.

Syntax

The DeleteTrustedHost method has the following format:

```
Netegrity::PolicyMgtSession->DeleteTrustedHost(TrustedHost)
```

Parameters

The DeleteTrustedHost method accepts the following parameter:

TrustedHost (PolicyMgtTrustedHost object)

Specifies the trusted host object to delete.

Return Value

The DeleteTrustedHost method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the trusted host is not found.

- value = -1

  Specifies that the method is unsuccessful.

DeleteUserDir Method—Deletes User Directory

The DeleteUserDir method

Syntax

The DeleteUserDir method has the following format:

```
Netegrity::PolicyMgtSession->DeleteUserDir(userdir)
```

Parameters

The DeleteUserDir method accepts the following parameter:

userdir (PolicyMgtUserDir object)

Specifies the user directory object to delete.

Return Value

The DeleteUserDir method returns one of the following values:

- value = 0

  Specifies that the method is successful or that the user directory is not found.

- value = -1

  Specifies that the method is unsuccessful.

GetAdmin Method—Retrieves Administrator

The GetAdmin method retrieves the specified administrator.

Syntax

The GetAdmin method has the following format:

```
Netegrity::PolicyMgtSession->GetAdmin(adminName)
```

Parameters

The GetAdmin method accepts the following parameter:

adminName (string)

Specifies the name of the administrator to retrieve.

Return Value

The GetAdmin method returns one of the following values:

- PolicyMgtAdmin object if the call is successful
- **undef** if the call is unsuccessful or the specified administrator does not exist

GetAffDomain Method—Retrieves Affiliate Domain

The GetAffDomain method retrieves the specified affiliate domain.

Syntax

The GetAffDomain method has the following format:

```
Netegrity::PolicyMgtSession->GetAffDomain(domName)
```

Parameters

The GetAffDomain method accepts the following parameter:

domName (string)

Specifies the name of the affiliate domain to retrieve.

Return Value

The GetAffDomain method returns one of the following values:

- PolicyMgtAffDomain object
- **undef** if the call is unsuccessful or the specified affiliate domain does not exist

GetAgent Method—Retrieves Agent

The GetAgent method retrieves the specified agent.

Syntax

The GetAgent method has the following format:

```
Netegrity::PolicyMgtSession->GetAgent(agentName)
```

Parameters

The GetAgent method accepts the following parameter:

agentName (string)

Specifies the name of the agent to retrieve.

Return Value

The GetAgent method returns one of the following values:

- PolicyMgtAgent object if the call is successful
- **undef** if the call is unsuccessful or the specified agent does not exist

GetAgentConfig Method—Retrieves Agent Configuration Object

The GetAgentConfig method retrieves the specified agent configuration object.

Syntax

The GetAgentConfig method has the following format:

```
Netegrity::PolicyMgtSession->GetAgentConfig(acName)
```

Parameters

The GetAgentConfig method accepts the following parameter:

acName (string)

Specifies the name of the agent configuration object to retrieve.

Return Value

The GetAgentConfig method returns one of the following values:

- PolicyMgtAgentConfig object if the call is successful
- **undef** if the call is unsuccessful or the specified agent configuration object does not exist

GetAgentGroup Method—Retrieves Agent Group

The GetAgentGroup method retrieves the specified agent group.

Syntax

The GetAgentGroup method has the following format:

```
Netegrity::PolicyMgtSession->GetAgentGroup(agentGroup)
```

Parameters

The GetAgentGroup method accepts the following parameter:

agentGroup (string)

Specifies the name of the agent group to retrieve.

Return Value

The GetAgentGroup method returns one of the following values:

- PolicyMgtGroup object
- **undef** if the call is unsuccessful or the specified agent group does not exist

GetAgentType Method—Retrieves Agent Type

The GetAgentType method retrieves the specified agent type.

Syntax

The GetAgentType method has the following format:

```
Netegrity::PolicyMgtSession->GetAgentType(agentTypeName)
```

Parameters

The GetAgentType method accepts the following parameter:

agentTypeName (string)

Specifies one of the following pre-defined agent types to retrieve:

- 3Com (RADIUS agent)
- Affiliate Agent (SiteMinder Affiliate agent)
- AffiliateMinder (AffiliateMinder agent)
- Ascend (RADIUS agent)
- Bay Networks (RADIUS agent)
- CheckPoint (RADIUS agent)
- Cisco (RADIUS agent)
- EJB Agent (SiteMinder EJB agent)
- Generic RADIUS (RADIUS agent)
- Livingston (RADIUS agent)
- Network Associates Sniffer (RADIUS agent)
- Servlet Agent (SiteMinder Servlet agent)
- Shiva (RADIUS agent)
- TeleBit (RADIUS agent)
- U.S. Robotics (RADIUS agent)
- Web Agent (SiteMinder Web agent)

Return Value

The GetAgentType method returns one of the following values:

- PolicyMgtAgentType object if the call is successful
- **undef** if the call is unsuccessful or the specified agent type does not exist

GetAllAdmins Method—Retrieves List of All Administrators

The GetAllAdmins method retrieves a list of all administrators configured on the Policy Server.

Syntax

The GetAllAdmins method has the following format:

```
Netegrity::PolicyMgtSession->GetAllAdmins()
```

Parameters

The GetAllAdmins method accepts no parameters.

Return Value

The GetAllAdmins method returns one of the following values:

- PolicyMgtAdmin (array)
- **undef** if the call is unsuccessful or no administrators exist

GetAllAffDomains Method—Retrieves List of All Affiliate Domains

The GetAllAffDomains method retrieves a list of all configured affiliate domains.

Syntax

The GetAllAffDomains method has the following format:

```
Netegrity::PolicyMgtSession->GetAllAffDomains()
```

Parameters

The GetAllAffDomains method accepts no parameters.

Return Value

The GetAllAffDomains method returns one of the following values:

- PolicyMgtAffDomain (array)
- **undef** if the call is unsuccessful or no affiliate domains exist

GetAllAgentConfigs Method—Retrieves List of All Agent Configuration Objects

The GetAllAgentConfigs method retrieves a list of all agent configuration objects.

Syntax

The GetAllAgentConfigs method has the following format:

```
Netegrity::PolicyMgtSession->GetAllAgentConfigs()
```

Parameters

The GetAllAgentConfigs method accepts no parameters.

Return Value

The GetAllAgentConfigs method returns one of the following values:

- PolicyMgtAgentConfig (array)
- **undef** if the call is unsuccessful or no agent configuration objects exist

GetAllAgentGroups Method—Retrieves List of All Agent Group Objects

The GetAllAgentGroups method retrieves a list of all agent group objects.

Syntax

The GetAllAgentGroups method has the following format:

```
Netegrity::PolicyMgtSession->GetAllAgentGroups()
```

Parameters

The GetAllAgentGroups method accepts no parameters.

Return Value

The GetAllAgentGroups method returns one of the following values:

- PolicyMgtGroup (array)
- **undef** if the call is unsuccessful

GetAllAgents Method—Retrieves List of All Agents

The GetAllAgents method retrieves a list of all agents configured on the Policy Server.

Syntax

The GetAllAgents method has the following format:

```
Netegrity::PolicyMgtSession->GetAllAgents()
```

Parameters

The GetAllAgents method accepts no parameters.

Return Value

The GetAllAgents method returns one of the following values:

- PolicyMgtAgent (array)
- **undef** if the call is unsuccessful or no agents exist

GetAllAuthAzMaps Method—Retrieves List of All AuthAz Maps

The GetAllAuthAzMaps method retrieves a list of all authentication and authorization maps.

Syntax

The GetAllAuthAzMaps method has the following format:

```
Netegrity::PolicyMgtSession->GetAllAuthAzMaps()
```

Parameters

The GetAllAuthAzMaps method accepts no parameters.

Return Value

The GetAllAuthAzMaps method returns one of the following values:

- PolicyMgtAuthAzMap (array)
- **undef** if the call is unsuccessful or no authentication and authorization maps exist

GetAllAuthSchemes Method—Retrieves List of Authentication Schemes

The GetAllAuthSchemes method retrieves a list of all authentication schemes on the Policy Server.

Syntax

The GetAllAuthSchemes method has the following format:

```
Netegrity::PolicyMgtSession->GetAllAuthSchemes([showTemplates])
```

Parameters

The GetAllAuthSchemes method accepts the following parameter:

showTemplates (int)

(Optional) Specifies whether to include template schemes in the list of authentication schemes.

- value = 0

  Specifies not including template schemes in the list of authentication schemes.

- value = 1

  Specifies including template schemes in the list of authentication schemes.

Return Value

The GetAllAuthSchemes method returns one of the following values:

- PolicyMgtAuthScheme (array)
- **undef** if the call is unsuccessful or no authentication schemes exist

GetAllCertMaps Method—Retrieves List of Certificate Mapping Objects

The GetAllCertMaps method retrieves a list of all certificate mapping objects.

Syntax

The GetAllCertMaps method has the following format:

```
Netegrity::PolicyMgtSession->GetAllCertMaps()
```

Parameters

The GetAllCertMaps method accepts no parameters.

Return Value

The GetAllCertMaps method returns one of the following values:

- PolicyMgtCertMap (array)
- **undef** if the call is unsuccessful or no certificate mapping objects exist

GetAllDomains Method—Retrieves List of All Domains

The GetAllDomains method retrieves a list of all domains configured on the Policy Server.

Syntax

The GetAllDomains method has the following format:

```
Netegrity::PolicyMgtSession->GetAllDomains()
```

Parameters

The GetAllDomains method accepts no parameters.

Return Value

The GetAllDomains method returns one of the following values:

- PolicyMgtDomain (array)
- **undef** if the call is unsuccessful or no domains exist

GetAllGlobalPolicies Method—Retrieves List of Global Policy Objects

The GetAllGlobalPolicies method retrieves a list of all global policy objects.

Syntax

The GetAllGlobalPolicies method has the following format:

```
Netegrity::GetAllGlobalPolicies()
```

Parameters

The GetAllGlobalPolicies method accepts no parameters.

Return Value

The GetAllGlobalPolicies method returns one of the following values:

- PolicyMgtPolicy (array)
- **undef** if the call is unsuccessful

GetAllGlobalResponses Method—Retrieves List of All Global Response Objects

The GetAllGlobalResponses method retrieves a list of all global response objects.

Syntax

The GetAllGlobalResponses method has the following format:

```
Netegrity::PolicyMgtSession->GetAllGlobalResponses()
```

Parameters

The GetAllGlobalResponses method accepts no parameters.

Return Value

The GetAllGlobalResponses method returns one of the following values:

- PolicyMgtResponse (array)
- **undef** if the call is unsuccessful

GetAllGlobalRules Method—Retrieves List of All Global Rule Objects

The GetAllGlobalRules method retrieves a list of all global rule objects.

Syntax

The GetAllGlobalRules method has the following format:

```
Netegrity::PolicyMgtSession->GetAllGlobalRules()
```

Parameters

The GetAllGlobalRules method accepts no parameters.

Return Value

The GetAllGlobalRules method returns one of the following values:

- PolicyMgtRule (array)
- **undef** if the call is unsuccessful

GetAllHostConfigs Method—Retrieves List of All Host Configuration Objects

The GetAllHostConfigs method retrieves a list of all host configuration objects.

Syntax

The GetAllHostConfigs method has the following format:

```
Netegrity::PolicyMgtSession->GetAllHostConfigs()
```

Parameters

The GetAllHostConfigs method accepts no parameters.

Return Value

The GetAllHostConfigs method returns one of the following values:

- PolicyMgtHostConfig (array)
- **undef** if the call is unsuccessful or no host configuration objects exist

GetAllODBCQuerySchemes Method—Retrieves List of All ODBC Query Schemes

The GetAllODBCQuerySchemes method retrieves a list of all ODBC query schemes on the Policy Server.

Syntax

The GetAllODBCQuerySchemes method has the following format:

```
Netegrity::PolicyMgtSession->GetAllODBCQuerySchemes()
```

Parameters

The GetAllODBCQuerySchemes method accepts no parameters.

Return Value

The GetAllODBCQuerySchemes method returns one of the following values:

- PolicyMgtODBCQueryScheme (array)
- **undef** if the call is unsuccessful or no ODBC query schemes exist

GetAllPwdPolicies Method—Retrieves List of All Password Policies

The GetAllPwdPolicies method retrieves a list of all configured password policies.

Syntax

The GetAllPwdPolicies method has the following format:

```
Netegrity::PolicyMgtSession->GetAllPwdPolicies()
```

Parameters

The GetAllPwdPolicies method accepts no parameters.

Return Value

The GetAllPwdPolicies method returns one of the following values:

- PolicyMgtPwdPolicy (array)
- **undef** if the call is unsuccessful or no password policies exist

GetAllRegSchemes Method—Retrieves List of All Registration Schemes

The GetAllRegSchemes method retrieves a list of all registration schemes configured on the Policy Server.

Syntax

The GetAllRegSchemes method has the following format:

```
Netegrity::PolicyMgtSession->GetAllRegSchemes()
```

Parameters

The GetAllRegSchemes method accepts no parameters.

Return Value

The GetAllRegSchemes method returns one of the following values:

- PolicyMgtRegScheme (array)
- **undef** if the call is unsuccessful or no registration schemes exist

GetAllSAMLAffiliations Method—Retrieves List of All SAML 2.0 Affiliations

The GetAllSAMLAffiliations method retrieves a list of all SAML 2.0 affiliations.

Syntax

The GetAllSAMLAffiliations method has the following format:

```
Netegrity::PolicyMgtSession->GetAllSAMLAffiliations()
```

Parameters

The GetAllSAMLAffiliations method accepts no parameters.

Return Value

The GetAllSAMLAffiliations method returns one of the following values:

- PolicyMgtSAMLAffiliation (array)
- **undef** if the call is unsuccessful

GetAllSAMLSchemeAttributes Method—Retrieves List of All Requester Attributes

The GetAllSAMLSchemeAttributes method retrieves a list of all defined SAML 2.0 Requester attributes.

Syntax

The GetAllSAMLSchemeAttributes method has the following format:

```
Netegrity::PolicyMgtSession->GetAllSAMLSchemeAttributes(scheme)
```

Parameters

The GetAllSAMLSchemeAttributes method accepts the following parameter:

scheme (PolicyMgtAuthScheme object)

Specifies the SAML 2.0 authentication scheme object.

Return Value

The GetAllSAMLSchemeAttributes method returns one of the following values:

- PolicyMgtSAMLRequesterAttr (array)
- **undef** if the call is unsuccessful

GetAllTrustedHosts Method—Retrieves List of All Trusted Host Objects

The GetAllTrustedHosts method retrieves a list of all trusted host objects.

Syntax

The GetAllTrustedHosts method has the following format:

```
Netegrity::PolicyMgtSession->GetAllTrustedHosts()
```

Parameters

The GetAllTrustedHosts method accepts no parameters.

Return Value

The GetAllTrustedHosts method returns one of the following values:

- PolicyMgtTrustedHost (array)
- **undef** if the call is unsuccessful or no trusted host objects exist

GetAllUserDirs Method—Retrieves List of All User Directories

The GetAllUserDirs method retrieves a list of all user directories associated with the Policy Server.

Syntax

The GetAllUserDirs method has the following format:

```
Netegrity::PolicyMgtSession->GetAllUserDirs()
```

Parameters

The GetAllUserDirs method accepts no parameters.

Return Value

The GetAllUserDirs method returns one of the following values:

- PolicyMgtUserDir (array)
- **undef** if the call is unsuccessful or no user directories exist

GetAllVariableTypes Method—Retrieves List of All Variable Type Objects

The GetAllVariableTypes method retrieves a list of all variable type objects configured on the Policy Server.

Syntax

The GetAllVariableTypes method has the following format:

```
Netegrity::PolicyMgtSession->GetAllVariableTypes()
```

Parameters

The GetAllVariableTypes method accepts no parameters.

Return Value

The GetAllVariableTypes method returns one of the following values:

- PolicyMgtVariableType (array)
- **undef** if the call is unsuccessful or no variable type objects exist

GetAuthScheme Method—Retrieves Authentication Scheme Object

The GetAuthScheme method retrieves the specified authentication scheme object. Existing authentication schemes are specified by name. To create a new authentication scheme, use this method to retrieve the type of authentication scheme object or template upon which you want the new scheme to be based. Then, pass the resulting object to the PolicyMgtSession->CreateAuthScheme (see page 422) method in the *schemeTemplate* parameter. For information about creating a SAML 2.0 authentication scheme, see the method PolicyMgtSession->CreateSAMLAuthScheme (see page 442).

Syntax

The GetAuthScheme method has the following format:

```
Netegrity::PolicyMgtSession->GetAuthScheme(schemeName)
```

Parameters

The GetAuthScheme method accepts the following parameter:

schemeName (string)

Specifies one of the following:

- The name of an existing authentication scheme.
- The type of authentication scheme that you want to create:
  - Anonymous Template
  - Basic over SSL Template
  - Basic Template
  - Custom Template
  - HTML Form Template
  - Impersonation Template
  - MS Passport Template
  - RADIUS CHAP/PAP Template
  - RADIUS Server Template
  - SafeWord HTML Form Template
  - SafeWord Template
  - SAML Artifact Template
  - SAML POST Template
  - SAML 2.0 Template
  - SecurID HTML Form Template
  - SecurID Template
  - TeleID Template
  - Windows Authentication Template
  - X509 Client Cert and Basic Template
  - X509 Client Cert and Form Template
  - X509 Client Cert or Basic Template
  - X509 Client Cert or Form Template
  - X509 Client Cert Template

Return Value

The GetAuthScheme method returns one of the following values:

- PolicyMgtAuthScheme (object)
- **undef** if the call is unsuccessful or the specified authentication scheme does not exist

GetCertMap Method—Retrieves Certificate Mapping Object

The GetCertMap method retrieves the certificate mapping object specified by the certificate issuer's DN.

Syntax

The GetCertMap method has the following format:

```
Netegrity::PolicyMgtSession->GetCertMap(issuerDN)
```

Parameters

The GetCertMap method accepts the following parameter:

issuerDN (string)

Specifies the certificate issuer's DN.

Return Value

The GetCertMap method returns one of the following values:

- PolicyMgtCertMap (object)
- **undef** if the call is unsuccessful or the certificate issuer's DN does not exist

GetDomain Method—Retrieves Domain Object

The GetDomain method retrieves the specified policy domain object.

Syntax

The GetDomain method has the following format:

```
Netegrity::PolicyMgtSession->GetDomain(domName)
```

Parameters

The GetDomain method accepts the following parameter:

domName (string)

Specifies the name of the domain to retrieve.

Return Value

The GetDomain method returns one of the following values:

- PolicyMgtDomain (object)
- **undef** if the call is unsuccessful or the specified domain does not exist

GetGlobalPolicy Method—Retrieves Global Policy Object

The GetGlobalPolicy method retrieves the specified global policy object.

Syntax

The GetGlobalPolicy method has the following format:

```
Netegrity::PolicyMgtSession->GetGlobalPolicy(policyName)
```

Parameters

The GetGlobalPolicy method accepts the following parameter:

policyName (string)

Specifies the name of the global policy to retrieve.

Return Value

The GetGlobalPolicy method returns one of the following values:

- PolicyMgtPolicy (object)
- **undef** if the call is unsuccessful or the specified global policy does not exist

GetGlobalResponse Method—Retrieves Global Response Object

The GetGlobalResponse method retrieves the specified global response object.

Syntax

The GetGlobalResponse method has the following format:

```
Netegrity::PolicyMgtSession->GetGlobalResponse(responseName)
```

Parameters

The GetGlobalResponse method accepts the following parameter:

responseName (string)

Specifies the name of the global response to retrieve.

Return Value

The GetGlobalResponse method returns one of the following values:

- PolicyMgtResponse (object)
- **undef** if the call is unsuccessful or the specified global response does not exist

GetGlobalRule Method—Retrieves Global Rule Object

The GetGlobalRule method retrieves the specified global rule object.

Syntax

The GetGlobalRule method has the following format:

```
Netegrity::PolicyMgtSession->GetGlobalRule(ruleName)
```

Parameters

The GetGlobalRule method accepts the following parameter:

ruleName (string)

Specifies the name of the global rule to retrieve.

Return Value

The GetGlobalRule method returns one of the following values:

- PolicyMgtRule (object)
- **undef** if the call is unsuccessful or the specified global rule does not exist

GetHostConfig Method—Retrieves Host Configuration Object

The GetHostConfig method retrieves the specified host configuration object.

Syntax

The GetHostConfig method has the following format:

```
Netegrity::PolicyMgtSession->GetHostConfig(hcName)
```

Parameters

The GetHostConfig method accepts the following parameter:

hcName (string)

Specifies the name of the host configuration object to retrieve.

Return Value

The GetHostConfig method returns one of the following values:

- PolicyMgtHostConfig (object)
- **undef** if the call is unsuccessful or the specified host configuration object does not exist

GetODBCQueryScheme Method—Retrieves ODBC Query Scheme Object

The GetODBCQueryScheme method retrieves the specified ODBC query scheme object.

Syntax

The GetODBCQueryScheme method has the following format:

```
Netegrity::PolicyMgtSession->GetODBCQueryScheme(schemeName)
```

Parameters

The GetODBCQueryScheme method accepts the following parameter:

schemeName (string)

Specifies the ODBC query scheme to retrieve.

Return Value

The GetODBCQueryScheme method returns one of the following values:

- PolicyMgtODBCQueryScheme (object)
- **undef** if the call is unsuccessful or the specified ODBC query scheme does not exist

GetPwdPolicy Method—Retrieves Password Policy Object

The GetPwdPolicy method retrieves the specified password policy object.

Syntax

The GetPwdPolicy method has the following format:

```
Netegrity::PolicyMgtSession->GetPwdPolicy(pwdPolicyName)
```

Parameters

The GetPwdPolicy method accepts the following parameter:

pwdPolicyName (string)

Specifies the name of the password policy to retrieve.

Return Value

The GetPwdPolicy method returns one of the following values:

- PolicyMgtPwdPolicy (object)
- **undef** if the call is unsuccessful or the specified password policy does not exist

GetRegScheme Method—Retrieves Registration Scheme Object

The GetRegScheme method retrieves the specified registration scheme object.

Syntax

The GetRegScheme method has the following format:

```
Netegrity::PolicyMgtSession->GetRegScheme(schemeName)
```

Parameters

The GetRegScheme method accepts the following parameter:

schemeName (string)

Specifies the name of the registration scheme to retrieve.

Return Value

The GetRegScheme method returns one of the following values:

- PolicyMgtRegScheme (object)
- **undef** if the call is unsuccessful or the specified registration scheme does not exist

GetSAMLAffiliation Method—Retrieves SAML 2.0 Affiliation Object

The GetSAMLAffiliation method retrieves the specified SAML 2.0 affiliation object.

Syntax

The GetSAMLAffiliation method has the following format:

```
Netegrity::PolicyMgtSession->GetSAMLAffiliation(affilName)
```

Parameters

The GetSAMLAffiliation method accepts the following parameter:

affilName (string)

Specifies the name or OID of the SAML affiliation to retrieve.

Note: When an OID is specified, it can be prefixed with the "@" character.

Return Value

The GetSAMLAffiliation method returns one of the following values:

- PolicyMgtSAMLAffiliation (object)
- **undef** if the call is unsuccessful or the specified SAML affiliation does not exist

GetSAMLAffiliationById Method—Retrieves SAML 2.0 Affiliation Object by ID

The GetSAMLAffiliationById method retrieves the SAML 2.0 affiliation object specified by the affiliation ID passed to the method.

Syntax

The GetSAMLAffiliationById method has the following format:

```
Netegrity::PolicyMgtSession->GetSAMLAffiliationById(affilID)
```

Parameters

The GetSAMLAffiliationById method accepts the following parameter:

affilID (string)

Specifies the affiliation ID of the SAML affiliation to retrieve.

Return Value

The GetSAMLAffiliationById method returns one of the following values:

- PolicyMgtSAMLAffiliation (object)
- **undef** if the call is unsuccessful or the specified SAML affiliation does not exist.

GetSharedSecretPolicy Method—Retrieves Shared Secret Policy Object

The GetSharedSecretPolicy method retrieves the specified shared secret rollover policy object. Because each policy store domain can have only one shared secret rollover policy, there is no need to pass the name of the policy to this method.

Syntax

The GetSharedSecretPolicy method has the following format:

```
Netegrity::PolicyMgtSession->GetSharedSecretPolicy()
```

Parameters

The GetSharedSecretPolicy method accepts no parameters.

Return Value

The GetSharedSecretPolicy method returns the following value:

- PolicyMgtSharedSecretPolicy (object)

GetTrustedHost Method—Retrieves Trusted Host Object

The GetTrustedHost method retrieves the specified trusted host object.

Syntax

The GetTrustedHost method has the following format:

```
Netegrity::PolicyMgtSession->GetTrustedHost(thName)
```

Parameters

The GetTrustedHost method accepts the following parameter:

thName (string)

Specifies the name of the trusted host to retrieve.

Return Value

The GetTrustedHost method returns one of the following values:

- PolicyMgtTrustedHost (object)
- **undef** if the call is unsuccessful or the specified trusted host does not exist

GetUserDir Method—Retrieves User Directory Object

The GetUserDir method retrieves the specified user directory object.

Syntax

The GetUserDir method has the following format:

```
Netegrity::PolicyMgtSession->GetUserDir(dirName)
```

Parameters

The GetUserDir method accepts the following parameter:

dirName (string)

Specifies the name of the user directory to retrieve.

Return Value

The GetUserDir method returns one of the following values:

- PolicyMgtUserDir (object)
- **undef** if the call is unsuccessful or the specified user directory does not exist

GetVariableType Method—Retrieves Variable Type Object

The GetVariableType method retrieves the specified variable type object. To create a new variable object of the specified type, pass the resulting variable type object to the CreateVariable method in the *varType* parameter.

Syntax

The GetVariableType method has the following format:

```
Netegrity::PolicyMgtSession->GetVariableType(varTypeName)
```

Parameters

The GetVariableType method accepts the following parameter:

varTypeName (string)

Specifies one of the following variable type names:

SiteMinder Variable Types

Post

UserContext

RequestContext

Static

WebService

TransactionMinder Variable Types

XMLBody

XMLAgent

XMLEnvelopeHeader

Transport

SAMLAssertion

Note: Variable type names are case-sensitive and must not contain spaces.

Return Value

The GetVariableType method returns one of the following values:

- PolicyMgtVariableType (object)
- **undef** if the call is unsuccessful

Remarks

You cannot create a TransactionMinder variable with the Command Line Interface. If you have TransactionMinder and the Option Pack installed, you can create TransactionMinder variables in the Administrative UI.

RemoveAttributeFromSAMLScheme Method—Removes Attribute from SAML Scheme

The RemoveAttributeFromSAMLScheme method removes an attribute from a SAML 2.0 authentication scheme.

Syntax

The RemoveAttributeFromSAMLScheme method has the following format:

```
Netegrity::PolicyMgtSession->RemoveAttributeFromSAMLScheme(scheme,
pSAMLRequesterAttr)
```

Parameters

The RemoveAttributeFromSAMLScheme method accepts the following parameters:

scheme (PolicyMgtAuthScheme object)

Specifies the SAML 2.0 authentication scheme from which to remove the attribute.

pSAMLRequesterAttribute (string)

Specifies the attribute to remove.

Return Value

The RemoveAttributeFromSAMLScheme method returns one of the following values:

- value = 0

  Specifies that the method is successful.

- value = -1

  Specifies that the method is unsuccessful.

SAMLAuthSchemeProperties Method—Sets or Retrieves SAML Metadata Properties

The SAMLAuthSchemeProperties method sets or retrieves the SAML 2.0 metadata properties that reside in an existing SAML 2.0 authentication scheme. For a complete list of SAML 2.0 metadata properties, see the method PolicyMgtSession->CreateSAMLAuthScheme (see page 442).

Syntax

The SAMLAuthSchemeProperties method has the following format:

```
Netegrity::PolicyMgtSession->SAMLAuthSchemeProperties(scheme, propsHash_ref)
```

Parameters

The SAMLAuthSchemeProperties method accepts the following parameters:

scheme (PolicyMgtAuthScheme object)

Specifies the authentication scheme whose metadata properties are set or retrieved.

propsHash_ref (hash)

Specifies a reference to a hashtable of metadata properties to set or retrieve.

Return Value

The SAMLAuthSchemeProperties method returns one of the following values:

- value = 0

  Specifies that the method is successful.

- value = -1

  Specifies that the method is unsuccessful.

Remarks

When the hashtable is empty, the SAMLAuthSchemeProperties method retrieves all metadata properties. You can define an empty hashtable as follows:

```
%myhash=();
```

Then, you can reference the empty hashtable as follows:

```
\%myhash
```

Finally, you can pass the hashtable reference to the SAMLAuthSchemeProperties method through the *propsHash_ref* parameter.

WSFEDAuthSchemeProperties Method—Sets or Retrieves WS-Federation Properties

The WSFEDAuthSchemeProperties method sets or retrieves the WS-Federation metadata properties in an existing WS-Federation authentication scheme. For a complete list of WS-Federation metadata properties, see the method PolicyMgtSession->CreateWSFEDAuthScheme (see page 453).

Syntax

The WSFEDAuthSchemeProperties method has the following format:

```
Netegrity::PolicyMgtSession->WSFEDAuthSchemeProperties(scheme, propsHash_ref)
```

Parameters

The WSFEDAuthSchemeProperties method accepts the following parameters:

scheme (PolicyMgtAuthScheme object)

Specifies the authentication scheme whose WS-Federation metadata properties are set or retrieved.

propsHash_ref (hash)

Specifies a reference to a hashtable of metadata properties to set or retrieve.

Return Value

The WSFEDAuthSchemeProperties method returns one of the following values:

- value = 0

  Specifies that the method is successful.

- value = -1

  Specifies that the method is unsuccessful.

Remarks

When the hashtable is empty, the WSFEDAuthSchemeProperties method retrieves all metadata properties. You can define an empty hashtable as follows:

```
%myhash=();
```

Then, you can reference the empty hashtable as follows:

```
\%myhash
```

Finally, you can pass the hashtable reference to the WSFEDAuthSchemeProperties method through the *propsHash_ref* parameter.

Shared Secret Rollover Methods

The following methods act on PolicyMgtSharedSecretPolicy objects:

- Enabled Method—Sets or Retrieves Rollover Enabled Flag for Policy
- RolloverFrequency Method—Sets or Retrieves Rollover Frequency for Policy
- RolloverPeriod Method—Sets or Retrieves Rollover Period for Policy
- Save Method—Saves Shared Secret Policy Object

Enabled Method—Sets or Retrieves Rollover Enabled Flag for Policy

The Enabled method sets or retrieves the flag that specifies whether the shared secret rollover policy is enabled.

Syntax

The Enabled method has the following format:

```
Netegrity::PolicyMgtSharedSecretPolicy->Enabled([enableFlag])
```

Parameters

The Enabled method accepts the following parameter:

enableFlag (int)

(Optional) Specifies a new value for the enabled flag.

- value = 1

  Specifies enabling the shared secret rollover policy.

- value = 0

  Specifies disabling the shared secret rollover policy.

Return Value

The Enabled method returns the new or existing value for the enabled flag:

- value = 1

  Specifies that the shared secret rollover policy is enabled.

- value = 0

  Specifies that the shared secret rollover policy is disabled.

Remarks

If the shared secret rollover policy is enabled, rollover must also be enabled for any trusted host whose shared secret needs to be synchronized with the rollover policy's shared secret. To enable rollover for a trusted host object, call the method PolicyMgtTrustedHost->RolloverEnabled.

RolloverFrequency Method—Sets or Retrieves Rollover Frequency for Policy

The RolloverFrequency method sets or retrieves the rollover frequency for the rollover policy. This value determines how often the shared secret is automatically updated in the time period specified by the method PolicyMgtSharedSecretPolicy->RolloverPeriod.

Syntax

The RolloverFrequency method has the following format:

```
Netegrity::PolicyMgtSharedSecretPolicy->RolloverFrequency([rollFreq])
```

Parameters

The RolloverFrequency method accepts the following parameter:

rollFreq (int)

(Optional) Specifies a new value for the rollover frequency.

Range: *rollFreq* >= 1

Return Value

The RolloverFrequency method returns the following value:

- rollover_frequency (int)

  Specifies the new or existing value for the rollover frequency.

RolloverPeriod Method—Sets or Retrieves Rollover Period for Policy

The RolloverPeriod method sets or retrieves the rollover period for the rollover policy. The rollover period can have one of four values: hourly, daily, weekly, or monthly. The rollover period is used with the rollover frequency to specify how often the shared secret is automatically changed. For example, if the rollover frequency is two and the rollover period is weekly, then the shared secret is automatically changed every two weeks. To set the rollover frequency, call the PolicyMgtSharedSecretPolicy->RolloverFrequency method.

Syntax

The RolloverPeriod method has the following format:

```
Netegrity::PolicyMgtSharedSecretPolicy->RolloverPeriod([rollPeriod])
```

Parameters

The RolloverPeriod method accepts the following parameter:

rollPeriod (int)

(Optional) Specifies a new value for the rollover period.

- value = 0

  Specifies that the rollover period is hourly.

- value = 1

  Specifies that the rollover period is daily.

- value = 2

  Specifies that the rollover period is weekly.

- value = 3

  Specifies that the rollover period is monthly.

Return Value

The RolloverPeriod method returns one of the following values:

- rollover_period (int)

  Specifies the new or existing value for the rollover period.

  Range: 0-3

- value = -1

  Specifies that the return value is not in the 0-3 range.

Save Method—Saves Shared Secret Policy Object

The Save method saves the shared secret policy object to the policy store. Call this method once after making all changes to the shared secret policy object. You must call this method for the changes to take effect.

Syntax

The Save method has the following format:

```
Netegrity::PolicyMgtSharedSecretPolicy->Save()
```

Parameters

The Save method accepts no parameters.

Return Value

The Save method returns one of the following values:

- value = 0

  Specifies that the call is successful.

- value = -1

  Specifies that the call is unsuccessful.

Trusted Host Methods

The following methods act on PolicyMgtTrustedHost objects:

- GetDescription Method—Retrieves Description of Trusted Host
- GetIPAddress Method—Retrieves IP Address of Trusted Host
- GetName Method—Retrieves Name of Trusted Host
- GetSecret Method—Retrieves Shared Secret of Trusted Host
- RolloverEnabled Method—Sets or Retrieves Shared Secret Rollover Flag
- SetSecret Method—Sets Shared Secret of Trusted Host

GetDescription Method—Retrieves Description of Trusted Host

The GetDescription method retrieves the description of the trusted host.

Syntax

The GetDescription method has the following format:

```
Netegrity::PolicyMgtTrustedHost->GetDescription()
```

Parameters

The GetDescription method accepts no parameters.

Return Value

The GetDescription method returns the following value:

- trusted_host_description

GetIPAddress Method—Retrieves IP Address of Trusted Host

The GetIPAddress method retrieves the IP address of the trusted host.

Syntax

The GetIPAddress method has the following format:

```
Netegrity::PolicyMgtTrustedHost->GetIPAddress()
```

Parameters

The GetIPAddress method accepts no parameters.

Return Value

The GetIPAddress method returns the following value:

- trusted_host_ip_address

GetName Method—Retrieves Name of Trusted Host

The GetName method retrieves the name of the trusted host.

Syntax

The GetName method has the following format:

```
Netegrity::PolicyMgtTrustedHost->GetName()
```

Parameters

The GetName method accepts no parameters.

Return Value

The GetName method returns the following value:

- trusted_host_name

GetSecret Method—Retrieves Shared Secret of Trusted Host

The GetSecret method retrieves the shared secret of the trusted host in clear text.

Syntax

The GetSecret method has the following format:

```
Netegrity::PolicyMgtTrustedHost->GetSecret()
```

Parameters

The GetSecret method accepts no parameters.

Return Value

The GetSecret method returns one of the following values:

- trusted_host_shared_secret
- **undef** if the call is unsuccessful

RolloverEnabled Method—Sets or Retrieves Shared Secret Rollover Flag

The RolloverEnabled method sets or retrieves the shared secret rollover flag that specifies whether shared secret rollover is enabled for this trusted host.

Syntax

The RolloverEnabled method has the following format:

```
Netegrity::PolicyMgtTrustedHost->RolloverEnabled([rolloverEnabled])
```

Parameters

The RolloverEnabled method accepts the following parameter:

rolloverEnabled (int)

(Optional) Specifies a new value for the shared secret rollover flag.

- value = 1

  Specifies that shared secret rollover is enabled for this trusted host.

- value = 0

  Specifies that shared secret rollover is *not* enabled for this trusted host.

Return Value

The RolloverEnabled method returns the new or existing value for the shared secret rollover flag:

- value = 1

  Specifies that shared secret rollover is enabled for this trusted host.

- value = 0

  Specifies that shared secret rollover is *not* enabled for this trusted host.

- **undef**

  Specifies that the call is unsuccessful.

Remarks

If shared secret rollover is enabled for this trusted host, it must also be enabled in the PolicyMgtSharedSecretPolicy object in the policy store domain where the trusted host is registered. If shared secret rollover is not enabled in this object, call the method PolicyMgtSharedSecretPolicy->Enabled to enable it.

SetSecret Method—Sets Shared Secret of Trusted Host

The SetSecret method sets the shared secret of the trusted host.

Syntax

The SetSecret method has the following format:

```
Netegrity::PolicyMgtTrustedHost->SetSecret([sharedSecret])
```

Parameters

The SetSecret method accepts the following parameter:

sharedSecret (string)

(Optional) Specifies the shared secret to set for the trusted host.

Note: If no shared secret is specified, SiteMinder generates a random 128-byte ASCII shared secret for the trusted host.

Return Value

The SetSecret method returns one of the following values:

- shared_secret (string)

  Specifies the new shared secret for the trusted host.

- "" (empty string)

  Specifies that the call is unsuccessful.

Remarks

When you use this method to set the shared secret, you must also run the SiteMinder tool smreghost to define the new shared secret in the host configuration file. (The host configuration file is named SmHost.conf by default.) Run smreghost with the -sh option. For more information, see the method PolicyMgtSession->AddTrustedHost (see page 415).

Note: You can schedule shared secret rollovers, so that they happen automatically. For more information about this feature, see the *Policy Server Configuration Guide*.

User Methods

The following methods act on PolicyMgtUser objects:

- DisableByAdmin Method—Sets or Retrieves Disabled-by-Administrator Flag
- DisableInactive Method—Sets or Retrieves Disabled-by-Inactivity Flag
- DisableMaxLoginFail Method—Sets or Retrieves Disabled-by-Max-Login-Failure Flag
- DisablePwdExpired Method—Sets or Retrieves Disabled-by-Password-Expired Flag
- ForcePwdChange Method—Sets or Retrieves Force-Password-Change Flag
- GetClass Method—Retrieves User Class
- GetPath Method—Retrieves User Path
- SetPassword Method—Sets a New Password
- UserPasswordState Method—Sets or Retrieves Password State Object
- ValidatePassword Method—Validates Password

DisableByAdmin Method—Sets or Retrieves Disabled-by-Administrator Flag

The DisableByAdmin method sets or retrieves the disabled-by-administrator flag which specifies whether the user account is disabled by the administrator.

Syntax

The DisableByAdmin method has the following format:

```
Netegrity::PolicyMgtUser->DisableByAdmin([disableFlag])
```

Parameters

The DisableByAdmin method accepts the following parameter:

disableFlag (int)

(Optional) Specifies a new value for the disabled-by-administrator flag.

- value = 1

  Specifies that the user account is disabled by the administrator.

- value = 0

  Specifies that the user account is not disabled by the administrator.

  Note: The user account can be disabled for other reasons. For more information, see Remarks.

Return Value

The DisableByAdmin method returns the new or existing value for the disabled-by-administrator flag:

- value = 1

  Specifies that the user account is disabled by the administrator.

- value = 0

  Specifies that the user account is *not* disabled by the administrator.

  Note: The user account can be disabled for other reasons. For more information, see Remarks.

- value = -1

  Specifies that the call is unsuccessful.

Remarks

User accounts can be disabled for one or more of the following reasons:

- The administrator disabled the user account.
- Account inactivity exceeded the time allowed.

  For more information, see the method PolicyMgtUser->DisableInactive (see page 504).

- The number of login failures exceeded the maximum allowed.

  For more information, see the method PolicyMgtUser->DisableMaxLoginFail (see page 506).

- The password expired.

  For more information, see the method PolicyMgtUser->DisablePwdExpired (see page 507).

DisableInactive Method—Sets or Retrieves Disabled-by-Inactivity Flag

The DisableInactive method sets or retrieves the disabled-by-inactivity flag which specifies whether the user account is disabled because account inactivity exceeded the time allowed.

Syntax

The DisableInactive method has the following format:

```
Netegrity::PolicyMgtUser->DisableInactive([disableFlag])
```

Parameters

The DisableInactive method accepts the following parameter:

disableFlag (int)

(Optional) Specifies a new value for the disabled-by-inactivity flag.

- value = 1

  Specifies that the user account is disabled because of inactivity.

- value = 0

  Specifies that the user account is not disabled because of inactivity.

Note: The user account can be disabled for other reasons. For more information, see Remarks.

Return Value

The DisableInactive method returns the new or existing value for the disabled-by-inactivity flag:

- value = 1

  Specifies that the user account is disabled because of inactivity.

- value = 0

  Specifies that the user account is *not* disabled because of inactivity.

  Note: The user account can be disabled for other reasons. For more information, see Remarks.

- value = -1

  Specifies that the call is unsuccessful.

Remarks

User accounts can be disabled for one or more of the following reasons:

- The administrator disabled the user account.

  For more information, see the method PolicyMgtUser->DisableByAdmin (see page 503).

- Account inactivity exceeded the time allowed.
- The number of login failures exceeded the maximum allowed.

  For more information, see the method PolicyMgtUser->DisableMaxLoginFail (see page 506).

- The password expired.

  For more information, see the method PolicyMgtUser->DisablePwdExpired (see page 507).

DisableMaxLoginFail Method—Sets or Retrieves Disabled-by-Max-Login-Failure Flag

The DisableMaxLoginFail method sets or retrieves the disabled-by-max-login-failure flag which specifies whether the user account is disabled because the number of login failures exceeded the maximum allowed.

Syntax

The DisableMaxLoginFail method has the following format:

```
Netegrity::PolicyMgtUser->DisableMaxLoginFail([disableFlag])
```

Parameters

The DisableMaxLoginFail method accepts the following parameter:

disableFlag (int)

(Optional) Specifies a new value for the disabled-by-max-login-failure flag.

- value = 1

  Specifies that the user account is disabled because the number of login failures exceeded the maximum allowed.

- value = 0

  Specifies that the user account is not disabled because the number of login failures exceeded the maximum allowed.

  Note: The user account can be disabled for other reasons. For more information, see Remarks.

Return Value

The DisableMaxLoginFail method returns the new or existing value for the disabled-by-max-login-failure flag:

- value = 1

  Specifies that the user account is disabled because the number of login failures exceeded the maximum allowed.

- value = 0

  Specifies that the user account is *not* disabled because the number of login failures exceeded the maximum allowed.

  Note: The user account can be disabled for other reasons. For more information, see Remarks.

- value = -1

  Specifies that the call is unsuccessful.

Remarks

User accounts can be disabled for one or more of the following reasons:

- The administrator disabled the user account.

  For more information, see the method PolicyMgtUser->DisableByAdmin (see page 503).

- Account inactivity exceeded the time allowed.

  For more information, see the method PolicyMgtUser->DisableInactive (see page 504).

- The number of login failures exceeded the maximum allowed.
- The password expired.

  For more information, see the method PolicyMgtUser->DisablePwdExpired (see page 507).

DisablePwdExpired Method—Sets or Retrieves Disabled-by-Password-Expired Flag

The DisablePwdExpired method sets or retrieves the disabled-by-password-expired flag that specifies whether the user account is disabled because the password expired.

Syntax

The DisablePwdExpired method has the following format:

```
Netegrity::PolicyMgtUser->DisablePwdExpired([disableFlag])
```

Parameters

The DisablePwdExpired method accepts the following parameter:

disableFlag (int)

(Optional) Specifies a new value for the disabled-by-password-expired flag.

- value = 1

  Specifies that the user account is disabled because the password expired.

- value = 0

  Specifies that the user account is not disabled because the password expired.

- **Note:** The user account can be disabled for other reasons. For more information, see Remarks.

Return Value

The DisablePwdExpired method returns the new or existing value for the disabled-by-password-expired flag:

- value = 1

  Specifies that the user account is disabled because the password expired.

- value = 0

  Specifies that the user account is *not* disabled because the password expired.

  Note: The user account can be disabled for other reasons. For more information, see Remarks.

- value = -1

  Specifies that the call is unsuccessful.

Remarks

User accounts can be disabled for one or more of the following reasons:

- The administrator disabled the user account.

  For more information, see the method PolicyMgtUser->DisableByAdmin (see page 503).

- Account inactivity exceeded the time allowed.

  For more information, see the method PolicyMgtUser->DisableInactive (see page 504).

- The number of login failures exceeded the maximum allowed.

  For more information, see the method PolicyMgtUser->DisableMaxLoginFail (see page 506).

- The password expired.

ForcePwdChange Method—Sets or Retrieves Force-Password-Change Flag

The ForcePwdChange method sets or retrieves the force-password-change flag that specifies whether to force a password change at the next user login.

Syntax

The ForcePwdChange method has the following format:

```
Netegrity::PolicyMgtUser->ForcePwdChange([forceFlag])
```

Parameters

The ForcePwdChange method accepts the following parameter:

forceFlag (int)

(Optional) Specifies whether to force a password change at the next user login.

- value = 1

  Specifies forcing a password change at the next user login.

- value = 0

  Specifies *not* forcing a password change at the next user login.

Return Value

The ForcePwdChange method returns the new or existing value for the force-password-change flag.

- value = 1

  Specifies forcing a password change at the next user login.

- value = 0

  Specifies *not* forcing a password change at the next user login.

- value = -1

  Specifies that the call is unsuccessful.

GetClass Method—Retrieves User Class

The GetClass method retrieves the user class.

Syntax

The GetClass method has the following format:

```
Netegrity::PolicyMgtUser->GetClass()
```

Parameters

The GetClass method accepts no parameters.

Return Value

The GetClass method returns one of the following values:

- user_class

  Example: "organization"

- **undef** if the call is unsuccessful

GetPath Method—Retrieves User Path

The GetPath method retrieves the user path. The user path is the distinguished name (DN).

Syntax

The GetPath method has the following format:

```
Netegrity::PolicyMgtUser->GetPath()
```

Parameters

The GetPath method accepts no parameters.

Return Value

The GetPath method returns one of the following values:

- user_path

  Specifies the user path or distinguished name (DN).

- **undef**

  Specifies that the call is unsuccessful.

SetPassword Method—Sets a New Password

The SetPassword method sets a new password for the user.

Syntax

The SetPassword method has the following format:

```
Netegrity::PolicyMgtUser->SetPassword(newPwd[, oldPwd])
```

Parameters

The SetPassword method accepts the following parameters:

newPwd (string)

Specifies the new password.

oldPwd (string)

(Optional) Specifies the old password to change.

Note: If provided, this value must match the existing password in the user directory.

Return Value

The SetPassword method returns one of the following values:

- value = 0

  Specifies that the password change is successful.

- value = -1

  Specifies that the password change is unsuccessful.

UserPasswordState Method—Sets or Retrieves Password State Object

The UserPasswordState method sets or retrieves the password state object for the current user. Setting a new password state object updates the object's attributes with any changes that have been made. This method also clears the password history if specified by the empty-history flag.

Syntax

The UserPasswordState method has the following format:

```
Netegrity::PolicyMgtUser->UserPasswordState([pPwState][, emptyHistoryFlag])
```

Parameters

The UserPasswordState method accepts the following parameters:

pPwState (PolicyMgtUserPasswordState)

(Optional) Specifies the new password state object to set.

emptyHistoryFlag (int)

(Optional) Specifies whether to clear the password history.

- value = 0 (default)

  Specifies *not* clearing the password history.

- value = 1

  Specifies clearing the password history.

  Note: Clearing the password history sets the last-password-change-time attribute to 0. For more information, see the method PolicyMgtUserPasswordState->LastPWChangeTime (see page 534).

Return Value

The UserPasswordState method returns one of the following values:

- PolicyMgtUserPasswordState (object)
- **undef** if the call is unsuccessful

ValidatePassword Method—Validates Password

The ValidatePassword method determines whether the user's password conforms to the password policy. Call ValidatePassword before calling the method SetPassword.

Syntax

The ValidatePassword method has the following format:

```
Netegrity::PolicyMgtUser->ValidatePassword(password)
```

Parameters

The ValidatePassword method accepts the following parameters:

password (string)

Specifies the password to validate.

Return Value

The ValidatePassword method returns one of the following values:

- value = 0

  Specifies that the password is valid.

- value = -1

  Specifies that the password is *not* valid.

User Directory Methods

The following methods act on PolicyMgtUserDir objects:

- AnonymousIDAttr Method—Sets or Retrieves Anonymous DN Name
- ChalRespAttr Method—Sets or Retrieves Challenge/Response Name
- Description Method—Sets or Retrieves Description of User Directory
- DisabledAttr Method—Sets or Retrieves Name of Disabled Attribute
- EmailAttr Method—Sets or Retrieves Email Attribute Name
- EnableSecurityContext Method—Sets or Retrieves Security Context Flag
- GetContents Method—Retrieves All Users in User Directory
- GetNamespace Method—Retrieves User Directory Namespace
- IsSecure Method—Sets or Retrieves Secure Authentication Flag
- LookupEntry Method—Retrieves Users that Match Specified Pattern
- MaxResults Method—Sets or Retrieves Maximum Search Results
- Name Method—Sets or Retrieves User Directory Name
- ODBCQueryScheme Method—Sets or Retrieves ODBC Query Scheme
- Password Method—Sets or Retrieves User Password
- PwdAttr Method—Sets or Retrieves Password Attribute Name
- PwdDataAttr Method—Sets or Retrieves Password Data Attribute Name
- RequireCredentials Method—Sets or Retrieves Whether Credentials Are Required
- SearchRoot Method—Sets or Retrieves Directory Search Root
- SearchScope Method—Sets or Retrieves LDAP Directory Search Scope
- SearchTimeout Method—Sets or Retrieves Maximum Directory Search Time
- Server Method—Sets or Retrieves a Directory-Dependent Value
- UIDAttr Method—Sets or Retrieves Universal ID Attribute Name
- UserLookupEnd Method—Sets or Retrieves User DN Lookup Endpoint
- UserLookupStart Method—Sets or Retrieves User DN Lookup Starting Point
- Username Method—Sets or Retrieves Username
- ValidateEntry Method—Validates User Directory Entry

AnonymousIDAttr Method—Sets or Retrieves Anonymous DN Name

The AnonymousIDAttr method sets or retrieves the name of the user directory's anonymous user DN attribute. The DN, which is defined in the anonymous authentication scheme, gives anonymous users access to resources protected by the anonymous authentication scheme. You can use the AnonymousIDAttr method with LDAP directories and some custom directories.

Syntax

The AnonymousIDAttr method has the following format:

```
Netegrity::PolicyMgtUserDir->AnonymousIDAttr([anonIDAttr])
```

Parameters

The AnonymousIDAttr method accepts the following parameter:

anonIDAttr (string)

(Optional) Specifies a new name for the anonymous user DN attribute.

Return Value

The AnonymousIDAttr method returns one of the following values:

- anonymous_user_dn_attribute_name (string)

  Specifies the new or existing name of the anonymous user DN attribute.

- **undef**

  Specifies that the call is unsuccessful.

ChalRespAttr Method—Sets or Retrieves Challenge/Response Name

The ChalRespAttr method sets or retrieves the name of the user directory's challenge/response attribute. You can use the ChalRespAttr method with LDAP directories and some custom directories.

Syntax

The ChalRespAttr method has the following format:

```
Netegrity::PolicyMgtUserDir->ChalRespAttr([chalRespAttr])
```

Parameters

The ChalRespAttr method accepts the following parameter:

chalRespAttr (string)

(Optional) Specifies a new name for the user directory's challenge/response attribute.

Return Value

The ChalRespAttr method returns one of the following values:

- challenge_response_attribute_name (string)

  Specifies the new or existing name of the user directory's challenge/response attribute.

- **undef**

  Specifies that the call is unsuccessful.

Description Method—Sets or Retrieves Description of User Directory

The Description method sets or retrieves the description of the user directory.

Syntax

The Description method has the following format:

```
Netegrity::PolicyMgtUserDir->Description([userDirDesc])
```

Parameters

The Description method accepts the following parameter:

userDirDesc (string)

(Optional) Specifies a new description for the user directory.

Return Value

The Description method returns one of the following values:

- user_directory_description (string)

  Specifies the new or existing description of the user directory.

- "" (empty string)

  Specifies that the call is unsuccessful.

DisabledAttr Method—Sets or Retrieves Name of Disabled Attribute

The DisabledAttr method sets or retrieves the name of the user directory attribute that contains the user's disabled state. This method applies to LDAP and ODBC directories and some custom directories.

Syntax

The DisabledAttr method has the following format:

```
Netegrity::PolicyMgtUserDir->DisabledAttr([disabledAttr])
```

Parameters

The DisabledAttr method accepts the following parameter:

disabledAttr (string)

(Optional) Specifies a new name for the user directory attribute that contains the user's disabled state.

Return Value

The DisabledAttr method returns one of the following values:

- disabled_attribute_name (string)

  Specifies the new or existing name of the user directory attribute that contains the user's disabled state.

- **undef**

  Specifies that the call is unsuccessful.

EmailAttr Method—Sets or Retrieves Email Attribute Name

The EmailAttr method sets or retrieves the name of the email attribute.

Note: This method is reserved for future use.

Syntax

The EmailAttr method has the following format:

```
Netegrity::PolicyMgtUserDir->EmailAttr([emailAttr])
```

Parameters

The EmailAttr method accepts the following parameter:

emailAttr (string)

(Optional) Specifies a new name for the email attribute.

Return Value

The EmailAttr method returns one of the following values:

- email_attribute_name (string)

  Specifies the new or existing name of the email attribute.

- **undef**

  Specifies that the call is unsuccessful.

EnableSecurityContext Method—Sets or Retrieves Security Context Flag

The EnableSecurityContext method sets or retrieves the user directory flag that specifies whether security context is enabled.

Syntax

The EnableSecurityContext method has the following format:

```
Netegrity::PolicyMgtUserDir->EnableSecurityContext([securityctxflag])
```

Parameters

The EnableSecurityContext method accepts the following parameter:

securityctxflag (int)

(Optional) Specifies a new value for the user directory's security context flag :

- value = 1 (enabled)
- value = 0 (disabled)

Return Value

The EnableSecurityContext method returns the new or existing value for the security context flag:

- value = 1

  Specifies that security context is enabled.

- value = 0

  Specifies that security context is disabled.

- Sm_PolicyApi_Failure

  Specifies that the call is unsuccessful.

GetContents Method—Retrieves All Users in User Directory

The GetContents method retrieves all users in the user directory.

Syntax

The GetContents method has the following format:

```
Netegrity::PolicyMgtUserDir->GetContents()
```

Parameters

The GetContents method accepts no parameters.

Return Value

The GetContents method returns one of the following values:

- PolicyMgtUser (array)
- **undef** if the call is unsuccessful

GetNamespace Method—Retrieves User Directory Namespace

The GetNamespace method retrieves the user directory namespace.

Syntax

The GetNamespace method has the following format:

```
Netegrity::PolicyMgtUserDir->GetNamespace()
```

Parameters

The GetNamespace method accepts no parameters.

Return Value

The GetNamespace method returns one of the following values:

- user_directory_namespace
- **undef** if the call is unsuccessful

IsSecure Method—Sets or Retrieves Secure Authentication Flag

The IsSecure method sets or retrieves the flag that specifies whether SiteMinder performs secure authentication for an LDAP or custom user directory. When this flag is enabled, SiteMinder authentication is secure and transmissions are encrypted. Enable this flag when using SSL.

Syntax

The IsSecure method has the following format:

```
Netegrity::PolicyMgtUserDir->IsSecure([secureFlag])
```

Parameters

The IsSecure method accepts the following parameter:

secureFlag (int)

(Optional) Specifies whether SiteMinder performs secure authentication:

- value = 1 (secure authentication is enabled)
- value = 0 (secure authentication is disabled)

Return Value

The IsSecure method returns the new or existing value for the secure authentication flag:

- value = 1

  Specifies that secure authentication is enabled.
- value = 0

  Specifies that secure authentication is disabled.
- value = -1

  Specifies that the call is unsuccessful.

LookupEntry Method—Retrieves Users that Match Specified Pattern

The LookupEntry method retrieves the user or users in the user directory that match the specified search pattern.

Syntax

The LookupEntry method has the following format:

```
Netegrity::PolicyMgtUserDir->LookupEntry(srchPattern)
```

Parameters

The LookupEntry method accepts the following parameter:

srchPattern (string)

Specifies the pattern to match when searching for users in the user directory.

Return Value

The LookupEntry method returns one of the following values:

- PolicyMgtUser (array)
- **undef** if the call is unsuccessful

MaxResults Method—Sets or Retrieves Maximum Search Results

The MaxResults method sets or retrieves the maximum number of search results to return from a search of an LDAP or custom user directory.

Syntax

The MaxResults method has the following format:

```
Netegrity::PolicyMgtUserDir->MaxResults([nResults])
```

Parameters

The MaxResults method accepts the following parameter:

nResults (int)

(Optional) Specifies a new number for the maximum results to return from a user directory search.

Return Value

The MaxResults method returns one of the following values:

- maximum_results (int)

  Specifies the new or existing maximum number of results to return from a user directory search.

- value = -1

  Specifies that the call is unsuccessful.

Name Method—Sets or Retrieves User Directory Name

The Name method sets or retrieves the name of the user directory.

Syntax

The Name method has the following format:

```
Netegrity::PolicyMgtUserDir->Name([userDirName])
```

Parameters

The Name method accepts the following parameter:

userDirName (string)

(Optional) Specifies a new name for the user directory.

Return Value

The Name method returns one of the following values:

- user_directory_name (string)

  Specifies the new or existing name of the user directory.

- **undef**

  Specifies that the call is unsuccessful.

ODBCQueryScheme Method—Sets or Retrieves ODBC Query Scheme

The ODBCQueryScheme method sets or retrieves the ODBC query scheme for the user directory.

Syntax

The ODBCQueryScheme method has the following format:

```
Netegrity::PolicyMgtUserDir->ODBCQueryScheme([odbcScheme])
```

Parameters

The ODBCQueryScheme method accepts the following parameters:

odbcScheme (PolicyMgtODBCQueryScheme)

(Optional) Specifies a new ODBC query scheme for the user directory.

Return Value

The ODBCQueryScheme method returns one of the following values:

- odbcScheme (PolicyMgtODBCQueryScheme)
- **undef** if no scheme exists or the call is unsuccessful

Password Method—Sets or Retrieves User Password

The Password method sets or retrieves the user password for access to the user directory.

Syntax

The Password method has the following format:

```
Netegrity::PolicyMgtUserDir->Password([pwd])
```

Parameters

The Password method accepts the following parameter:

pwd (string)

(Optional) Specifies a new user password for access to the user directory.

Return Value

The Password method returns one of the following values:

- password (string)

  Specifies the new or existing user password.

- **undef**

  Specifies that the call is unsuccessful.

PwdAttr Method—Sets or Retrieves Password Attribute Name

The PwdAttr method sets or retrieves the name of the user directory's password attribute.

Syntax

The PwdAttr method has the following format:

```
Netegrity::PolicyMgtUserDir->PwdAttr([pwdAttr])
```

Parameters

The PwdAttr method accepts the following parameter:

pwdAttr (string)

(Optional) Specifies a new name for the user directory's password attribute.

Return Value

The PwdAttr method returns one of the following values:

- password_attribute_name (string)

  Specifies the new or existing name of the user directory's password attribute.

- **undef**

  Specifies that the call is unsuccessful.

PwdDataAttr Method—Sets or Retrieves Password Data Attribute Name

The PwdDataAttr method sets or retrieves the name of the user directory's password data attribute.

Syntax

The PwdDataAttr method has the following format:

```
Netegrity::PolicyMgtUserDir->PwdDataAttr([pwdDataAttr])
```

Parameters

The PwdDataAttr method accepts the following parameter:

pwdDataAttr (string)

(Optional) Specifies a new name for the user directory's password data attribute.

Return Value

The PwdDataAttr method returns one of the following values:

- password_data_attribute_name (string)

  Specifies the new or existing name of the user directory's password data attribute.

- **undef**

  Specifies that the call is unsuccessful.

RequireCredentials Method—Sets or Retrieves Whether Credentials Are Required

The RequireCredentials method sets or retrieves the flag that specifies whether SiteMinder is required to check user credentials.

Syntax

The RequireCredentials method has the following format:

```
Netegrity::PolicyMgtUserDir->RequireCredentials([credFlag])
```

Parameters

The RequireCredentials method accepts the following parameter:

credFlag (int)

(Optional) Specifies whether SiteMinder is required to check user credentials:

- value = 1 (credentials required)
- value = 0 (credentials are not required)

Return Value

The RequireCredentials method returns the new or existing value for the require credentials flag:

- value = 1

  Specifies that credentials are required.

- value = 0

  Specifies that credentials are not required.

- value = -1

  Specifies that the call is unsuccessful.

SearchRoot Method—Sets or Retrieves Directory Search Root

The SearchRoot method sets or retrieves different values for different directory types:

LDAP Directories

The SearchRoot method sets or retrieves the location in the LDAP tree that is the starting point for the directory connection, for example, the organization (o) or organizational unit (ou). This location, called the search root, is the point where the Policy Server starts the search for a user.

Custom Directories

The SearchRoot method sets or retrieves a string of parameters to pass to the custom library.

Syntax

The SearchRoot method has the following format:

```
Netegrity::PolicyMgtUserDir->SearchRoot([srchRoot])
```

Parameters

The SearchRoot method accepts the following parameter:

srchRoot (string)

Specifies a new search root for an LDAP directory or parameter string for a custom directory.

Return Value

The SearchRoot method returns one of the following values:

- search_root (string)

  Specifies the new or existing search root for an LDAP directory or parameter string for a custom directory.

- **undef**

  Specifies that the call is unsuccessful.

SearchScope Method—Sets or Retrieves LDAP Directory Search Scope

The SearchScope method sets or retrieves the search scope for an LDAP user directory. The search scope specifies how many levels SiteMinder searches for users or user groups in the LDAP directory.

Syntax

The SearchScope method has the following format:

```
Netegrity::PolicyMgtUserDir->SearchScope([searchScope])
```

Parameters

The SearchScope method accepts the following parameter:

searchScope (int)

(Optional) Specifies a new search scope for an LDAP user directory:

- USERDIR_SCOPE_SUBTREE

  Specifies searching the root and all levels below.

- USERDIR_SCOPE_ONELEVEL

  Specifies searching the root and one level below.

Return Value

The SearchScope method returns one of the following new or existing values:

- USERDIR_SCOPE_SUBTREE

  Specifies searching the root and all levels below.

- USERDIR_SCOPE_ONELEVEL

  Specifies searching the root and one level below.

- value = -1

  Specifies that the call is unsuccessful.

SearchTimeout Method—Sets or Retrieves Maximum Directory Search Time

The SearchTimeout method sets or retrieves the maximum time, in seconds, allowed for searching an LDAP or custom user directory.

Syntax

The SearchTimeout method has the following format:

```
Netegrity::PolicyMgtUserDir->SearchTimeout([maxTimeout])
```

Parameters

The SearchTimeout method accepts the following parameter:

maxTimeout (int)

(Optional) Specifies a new maximum time (in seconds) allowed for searching an LDAP or custom user directory.

Return Value

The SearchTimeout method returns one of the following values:

- maximum_time_allowed (int)

  Specifies the new or existing maximum time (in seconds) allowed for searching an LDAP or custom user directory.

- value = -1

  Specifies that the call is unsuccessful.

Server Method—Sets or Retrieves a Directory-Dependent Value

The Server method sets or retrieves a value. The type of value depends on the type of user directory, as follows:

LDAP and AD Directories

The Server method sets or retrieves the LDAP server's IP address and port number.

ODBC Directories

The Server method sets or retrieves the data source name.

WinNT Directories

The Server method sets or retrieves the domain name.

Custom Directories

The Server method sets or retrieves the library name.

Syntax

The Server method has the following format:

```
Netegrity::PolicyMgtUserDir->Server([server])
```

Parameters

The Server method accepts the following parameter:

server (string)

(Optional) Specifies a new value for one of the following types of directories:

- LDAP and AD Directories

  Specifies a new IP address and port number for the LDAP server.

  Format: IP_address:port_number

  Default port number: 389

- ODBC Directories

  Specifies a new data source name.

- WinNT Directories

  Specifies a new domain name.

- Custom Directories

  Specifies a new library name.

Return Value

The Server method returns one of the following values:

- value (string)

  Specifies the new or existing value for the user directory.

- **undef**

  Specifies that the call is unsuccessful.

UIDAttr Method—Sets or Retrieves Universal ID Attribute Name

The UIDAttr method sets or retrieves the name of the user directory's universal ID attribute. The universal ID is different from the user's login ID and is used to look up user information. This method applies to LDAP, ODBC, and WinNT directories and to some custom directories.

Syntax

The UIDAttr method has the following format:

```
Netegrity::PolicyMgtUserDir->UIDAttr([uidAttr])
```

Parameters

The UIDAttr method accepts the following parameter:

uidAttr (string)

(Optional) Specifies a new name for the universal ID attribute.

Return Value

The UIDAttr method returns one of the following values:

- uid_attribute_name (string)

  Specifies the new or existing name of the universal ID attribute.

- **undef**

  Specifies that the call is unsuccessful.

UserLookupEnd Method—Sets or Retrieves User DN Lookup Endpoint

The UserLookupEnd method sets or retrieves the endpoint for a user DN lookup in an LDAP directory.

Syntax

The UserLookupEnd method has the following format:

```
Netegrity::PolicyMgtUserDir->UserLookupEnd([lookupEnd])
```

Parameters

The UserLookupEnd method accepts the following parameter:

lookupEnd (string)

(Optional) Specifies a new value for the user DN lookup endpoint.

Return Value

The UserLookupEnd method returns one of the following values:

- user_dn_lookup_endpoint (string)

  Specifies the new or existing user DN lookup endpoint.

- **undef**

  Specifies that the call is unsuccessful.

Remarks

Specifying values for the user DN lookup starting point and endpoint allows users to enter part of the DN string when authenticating. In the following example, the user only needs to specify the string "JSmith" and not the whole DN string when logging in:

- DN = "uid=JSmith,ou=marketing,o=myorg.org"
- starting_point = "uid="
- endpoint = ",ou=marketing,o=myorg.org"
- login = "JSmith"

UserLookupStart Method—Sets or Retrieves User DN Lookup Starting Point

The UserLookupStart method sets or retrieves the starting point for a user DN lookup in an LDAP directory.

Syntax

The UserLookupStart method has the following format:

```
Netegrity::PolicyMgtUserDir->UserLookupStart([lookupStart])
```

Parameters

The UserLookupStart method accepts the following parameter:

lookupStart (string)

(Optional) Specifies a new value for the user DN lookup starting point.

Return Value

The UserLookupStart method returns one of the following values:

- user_dn_lookup_starting_point (string)

  Specifies the new or existing user DN lookup starting point.

- **undef**

  Specifies that the call is unsuccessful.

Remarks

Specifying values for the user DN lookup starting point and endpoint allows users to enter part of the DN string when authenticating. In the following example, the user only needs to specify the string "JSmith" and not the whole DN string when logging in:

- DN = "uid=JSmith,ou=marketing,o=myorg.org"
- starting_point = "uid="
- endpoint = ",ou=marketing,o=myorg.org"
- login = "JSmith"

Username Method—Sets or Retrieves Username

The Username method sets or retrieves the username required for accessing the user directory. Set the username only if the RequireCredentials method returns the value of 1.

Syntax

The Username method has the following format:

```
Netegrity::PolicyMgtUserDir->Username([username])
```

Parameters

The Username method accepts the following parameters:

username (string)

(Optional) Specifies a new name for the user.

Return Value

The Username method returns one of the following values:

- user_name (string)

  Specifies the new or existing name of the user.
- **undef**

  Specifies that the call is unsuccessful.

ValidateEntry Method—Validates User Directory Entry

The ValidateEntry method validates a user directory entry.

Syntax

The ValidateEntry method has the following format:

```
Netegrity::PolicyMgtUserDir->ValidateEntry(path)
```

Parameters

The ValidateEntry method accepts the following parameter:

path (string)

Specifies the path of the user or user group to validate.

Return Value

The ValidateEntry method returns one of the following values:

- value = 0

  Specifies that the method is successful.

- value = -1

  Specifies that the method is unsuccessful.

User Password State Methods

The following methods act on PolicyMgtUserPasswordState objects:

- DisabledTime Method—Sets or Retrieves Time Object Was Disabled
- LastPWChangeTime Method—Sets or Retrieves Time Password Last Changed
- LastLoginTime Method—Sets or Retrieves Last Login Time
- LoginFailures Method—Sets or Retrieves Number of Login Failures
- PrevLoginTime Method—Sets or Retrieves Previous Login Time

DisabledTime Method—Sets or Retrieves Time Object Was Disabled

The DisabledTime method sets or retrieves the time that the user object was disabled. The time is represented as the number of seconds that have elapsed since a particular instant in time that varies from system to system. One common representation is the number of seconds that have elapsed since 00:00:00 1/1/1970 UTC (Coordinated Universal Time).

Syntax

The DisabledTime method has the following format:

```
Netegrity::PolicyMgtUserPasswordState->DisabledTime([time])
```

Parameters

The DisabledTime method accepts the following parameter:

time (long)

(Optional) Specifies a new time for when the user object was disabled.

Return Value

The DisabledTime method returns the following value:

- time (long)

  Specifies the new or existing time that the user object was disabled.

LastPWChangeTime Method—Sets or Retrieves Time Password Last Changed

The LastPWChangeTime method sets or retrieves the time that the user's password was last changed. The time is represented as the number of seconds that have elapsed since a particular instant in time that varies from system to system. One common representation is the number of seconds that have elapsed since 00:00:00 1/1/1970 UTC (Coordinated Universal Time).

Syntax

The LastPWChangeTime method has the following format:

```
Netegrity::PolicyMgtUserPasswordState->LastPWChangeTime([time])
```

Parameters

The LastPWChangeTime method accepts the following parameter:

time (long)

Specifies a new time for when the user's password was last changed.

Return Value

The LastPWChangeTime method returns one of the following values:

- time (long)

  Specifies the new or existing time that the user's password was changed.

- value = 0

  Specifies that the user started to change the password, but did not complete the procedure.

LastLoginTime Method—Sets or Retrieves Last Login Time

The LastLoginTime method sets or retrieves the time that the user last logged in successfully. The time is represented as the number of seconds that have elapsed since a particular instant in time that varies from system to system. One common representation is the number of seconds that have elapsed since 00:00:00 1/1/1970 UTC (Coordinated Universal Time).

Syntax

The LastLoginTime method has the following format:

```
Netegrity::PolicyMgtUserPasswordState->LastLoginTime([time])
```

Parameters

The LastLoginTime method accepts the following parameter:

time (long)

(Optional) Specifies a new time for when the user last logged in successfully.

Return Value

The LastLoginTime method returns the following value:

- time (long)

  Specifies the new or existing time that the user last logged in successfully.

LoginFailures Method—Sets or Retrieves Number of Login Failures

The LoginFailures method sets or retrieves the number of times the user failed to log in since the user's last successful login.

Syntax

The LoginFailures method has the following format:

```
Netegrity::PolicyMgtUserPasswordState->LoginFailures([count])
```

Parameters

The LoginFailures method accepts the following parameter:

count (int)

(Optional) Specifies a new value for the number of login failures.

Return Value

The LoginFailures method returns one of the following values:

- count (int)

  Specifies the new or existing number of login failures since the user's last successful login.

Variable Type Methods

The following methods act on PolicyMgtVariableType objects. PolicyMgtVariableType objects are read-only:

- GetDescription Method—Retrieves Description of Variable Type Object
- GetName Method—Retrieves Name of Variable Type

GetName Method—Retrieves Name of Variable Type Object

The GetName method retrieves the name of the variable type object. The variable type object is read-only. See the PolicyMgtSession->GetVariableType (see page 491) method for the list of variable type object names that GetName can return.

Syntax

The GetName method has the following format:

```
Netegrity::PolicyMgtVariableType->GetName()
```

Parameters

The GetName method accepts no parameters.

Return Value

The GetName method returns one of the following values:

- variable_type_object_name (string)
- **undef** if the call is unsuccessful

WS-Federation Resource Partner Methods

The following methods act on PolicyMgtWSFEDResourcePartner objects:

- AddAttribute Method—Adds Attribute to Resource Partner
- AddUser Method—Adds User to Resource Partner
- CreateIPConfigHostName Method—Creates Object Based on Specified Host
- CreateIPConfigRange Method—Creates Object Based on Address Range
- CreateIPConfigSingleHost Method—Creates Object Based on Single Address
- CreateIPConfigSubnetMask Method—Creates Object Based on Subnet Address
- DeleteIPConfig Method—Deletes Specified IP Configuration Object
- GetAllAttributes Method—Retrieves All Attributes for Resource Partner
- GetAllIPConfigs Method—Retrieves All IP Configuration Objects for Service Provider
- GetAllUsers Method—Retrieves All Users Associated with Resource Partner
- Property Method—Sets or Retrieves Resource Partner Property
- RemoveUser Method—Removes Specified User from Resource Partner
- Save Method—Saves Resource Partner's Metadata

AddAttribute Method—Adds Attribute to Resource Partner

The AddAttribute method adds an attribute to the WS-Federation Resource Partner.

Syntax

The AddAttribute method has the following format:

```
Netgerity::PolicyMgtWSFEDResourcePartner->AddAttribute(attrNameFormat, value)
```

Parameters

The AddAttribute method accepts the following parameters:

attrNameFormat (int)

Specifies one of the following attribute types:

- WSFEDRP_EMAILADDRESS
- WSFEDRP_UPN
- WSFEDRP_COMMON
- WSFEDRP_GROUP
- WSFEDRP_NAMEVALUE

value (string)

Specifies an attribute value in one of the following formats:

- Static: *variableName* = *value*

  Note: The value's format must match the attribute's type, unless the type is WSFEDRP_NAMEVALUE. In this case, the value can be in any format.

- User Attribute: *variableName* = <%userattr=*"AttrName"*%>

- DN Attribute: *variableName* = <#dn=*"DNSpec"* attr=*"AttrName"*#>

  Note: To allow SiteMinder to retrieve DN attributes from a nested group, preface *DNSpec* with an exclamation point (!), as follows: dn=*"!ou=People,o=security.com"*

- Active Response

Return Value

The AddAttribute method returns one of the following values:

- PolicyMgtWSFEDRPAttr (object)
- **undef** if the call is unsuccessful

AddUser Method—Adds User to Resource Partner

The AddUser method adds a user to the WS-Federation Resource Partner.

Syntax

The AddUser method has the following format:

```
Netegrity::PolicyMgtWSFEDResourcePartner->AddUser(user)
```

Parameters

The AddUser method accepts the following parameter:

user (PolicyMgtUser object)

Specifies the user to add to the Resource Partner.

Return Value

The AddUser method returns one of the following values:

- value = 0

  Specifies that the method is successful.

- value = -1

  Specifies that the method is unsuccessful.

CreateIPConfigHostName Method—Creates Object Based on Specified Host

The CreateIPConfigHostName method creates an IP configuration object for the WS-Federation Resource Partner based on the specified host name. This method creates an IP address restriction for the assertion generation policy. With this address restriction, only assertions generated from the specified host are accepted.

Syntax

The CreateIPConfigHostName method has the following format:

```
Netegrity::PolicyMgtWSFEDResourcePartner->CreateIPConfigHostName(hostName)
```

Parameters

The CreateIPConfigHostName method accepts the following parameter:

hostName (string)

Specifies the name of the host where assertions must originate.

Return Value

The CreateIPConfigHostName method returns one of the following values:

- PolicyMgtIPConfig (object)
- **undef** if the call is unsuccessful

CreateIPConfigSingleHost Method—Creates Object Based on Single Address

The CreateIPConfigSingleHost method creates an IP configuration object for the WS-Federation Resource Partner based on the specified IP address. This method creates an IP address restriction for the assertion generation policy. With this address restriction, only assertions generated from the specified IP address are accepted.

Syntax

The method has the following format:

```
Netegrity::PolicyMgtWSFEDResourcePartner->CreateIPConfigSingleHost(ipAddr)
```

Parameters

The CreateIPConfigSingleHost method accepts the following parameter:

ipAddr (string)

Specifies the IP address where assertions must originate.

Return Value

The CreateIPConfigSingleHost method returns one of the following values:

- PolicyMgtIPConfig (object)
- **undef** if the call is unsuccessful

CreateIPConfigSubnetMask Method—Creates Object Based on Subnet Address

The CreateIPConfigSubnetMask method creates an IP configuration object for the WS-Federation Resource Partner based on the specified IP address and subnet mask. This method creates an IP address restriction for the assertion generation policy. With this address restriction, only assertions generated from the subnet address are accepted. The subnet address is derived from the specified IP address and subnet mask.

Syntax

The CreateIPConfigSubnetMask method has the following format:

```
Netegrity::PolicyMgtWSFEDResourcePartner->CreateIPConfigSubnetMask(ipAddr,
subnetMask)
```

Parameters

The CreateIPConfigSubnetMask method accepts the following parameters:

ipAddr (string)

Specifies the IP address used to derive the subnet address.

subnetMask (unsigned long)

Specifies the subnet mask used to derive the subnet address.

Note: For more information about the subnet mask, see the method PolicyMgtPolicy->CreateIPConfigSubnetMask (see page 346).

Return Value

The CreateIPConfigSubnetMask method returns one of the following values:

- PolicyMgtIPConfig (object)
- **undef** if the call is unsuccessful

DeleteIPConfig Method—Deletes Specified IP Configuration Object

The DeleteIPConfig method deletes the specified IP configuration object.

Syntax

The DeleteIPConfig method has the following format:

```
Netegrity::PolicyMgtWSFEDResourcePartner->DeleteIPConfig(IPConfig)
```

Parameters

The DeleteIPConfig method accepts the following parameter:

IPConfig (PolicyMgtIPConfig object)

Specifies the IP configuration object to delete.

Return Value

The DeleteIPConfig method returns one of the following values:

- value = 0

  Specifies that the method is successful.
- value = -1

  Specifies that the method is unsuccessful.

GetAllAttributes Method—Retrieves All Attributes for Resource Partner

The GetAllAttributes method retrieves all attributes defined for the WS-Federation Resource Partner.

Syntax

The GetAllAttributes method has the following format:

```
Netegrity::PolicyMgtWSFEDResourcePartner->GetAllAttributes()
```

Parameters

The GetAllAttributes method accepts no parameters.

Return Value

The GetAllAttributes method returns one of the following values:

- PolicyMgtWSFEDRPAttr (array)
- **undef** if the call is unsuccessful

GetAllIPConfigs Method—Retrieves All IP Configuration Objects for Service Provider

The GetAllIPConfigs method retrieves all IP configuration objects for the Service Provider.

Syntax

The GetAllIPConfigs method has the following format:

```
Netegrity::PolicyMgtWSFEDResourcePartner->GetAllIPConfigs()
```

Parameters

The GetAllIPConfigs method accepts no parameters.

Return Value

The GetAllIPConfigs method returns one of the following values:

- PolicyMgtIPConfig (array)
- **undef** if no IP configuration objects are found

GetAllUsers Method—Retrieves All Users Associated with Resource Partner

The GetAllUsers method retrieves all users associated with the WS-Federation Resource Partner. If a user directory is specified, this method only returns the users associated with the specified directory.

Syntax

The GetAllUsers method has the following format:

```
Netegrity::PolicyMgtWSFEDResourcePartner->GetAllUsers([userDir])
```

Parameters

The GetAllUsers method accepts the following parameter:

userDir (PolicyMgtUserDir object)

(Optional) Specifies only those users associated with the user directory.

Return Value

The GetAllUsers method returns one of the following values:

- PolicyMgtUser (array)
- **undef** if no users are found or an error occurs

Property Method—Sets or Retrieves Resource Partner Property

The Property method sets or retrieves the value of the specified WS-Federation Resource Partner property. For a list of metadata properties, see the WS-Federation Property Reference in this guide.

Note: After modifying one or more properties, call Save to write the changes to the policy store.

Syntax

The Property method has the following format:

```
Netegrity::PolicyMgtWSFEDResourcePartner->Property(name, [newvalue])
```

Parameters

The Property method accepts the following parameters:

name (string)

Specifies the property to set or retrieve.

newvalue (string)

(Optional) Specifies a new value for the Resource Partner property.

Return Value

The Property method returns one of the following values:

- value

  Specifies the new or existing value of the property.

- **undef**

  Specifies that the call is unsuccessful.

RemoveAtrribute Method--Removes an Attribute from a WSFED Resource Partner

The RemoveAttribute method removes an attribute from the WS-Federation Resource Partner.

Syntax

The RemoveAttribute method has the following format:

```
Netgerity::PolicyMgtWSFEDResourcePartner->RemoveAttribute(attrName)
```

Parameters

The RemoveAttribute method accepts the following parameter:

attrName (PolicyMgtWSFEDRPAttr)

Specifies the attribute to remove.

Return Value

The RemoveAttribute method returns one of the following values:

- 0 on success
- -1 on failure

RemoveUser Method—Removes Specified User from Resource Partner

The RemoveUser method removes the specified user from the WS-Federation Resource Partner.

Syntax

The RemoveUser method has the following format:

```
Netegrity::PolicyMgtWSFEDResourcePartner->RemoveUser(user)
```

Parameters

The RemoveUser method accepts the following parameter:

user (PolicyMgtUser object)

Specifies the user to remove from the Resource Partner.

Return Value

The RemoveUser method returns one of the following values:

- value = 0

  Specifies that the method is successful.
- value = -1

  Specifies that the method is unsuccessful.

Save Method—Saves Resource Partner's Metadata

The Save method writes the WS-Federation Resource Partner's metadata to the policy store. To modify the metadata, call the PolicyMgtWSFEDResourcePartner->Property method. Then, call Save to save the changes.

Syntax

The Save method has the following format:

```
Netegrity::PolicyMgtWSFEDResourcePartner->Save()
```

Parameters

The Save method accepts no parameters.

Return Value

The Save method returns one of the following values:

- value = 0

  Specifies that the method is successful.
- value = -1

  Specifies that the method is unsuccessful.

- value = -4

  Specifies that the user lacks the privileges required to save the changes.

- value = -10

  Specifies that the path and class are empty.

WS-Federation Resource Partner Attribute Methods

The following methods act on PolicyMgtWSFEDResourcePartnerAttr objects:

- GetAttrNameFormat Method—Retrieves Format of Attribute Names
- GetValue Method—Retrieves Attribute Value

GetAttrNameFormat Method—Retrieves Format of Attribute Names

The GetAttrNameFormat method retrieves the format of attribute names used with this WS-Federation Resource Partner.

Syntax

The GetAttrNameFormat method has the following format:

```
Netegrity::PolicyMgtWSFEDRPattr->GetAttrNameFormat()
```

Parameters

The GetAttrNameFormat method accepts no parameters.

Return Value

The GetAttrNameFormat method returns one of the following format values:

- WSFEDRP_EMAILADDRESS (value = 0)
- WSFEDRP_UPN (value = 1)
- WSFEDRP_COMMON (value = 2)
- WSFEDRP_GROUP (value = 3)
- WSFEDRP_NAMEVALUE (value = 4)

GetValue Method—Retrieves Attribute Value

The GetValue method retrieves the value of the WS-Federation Resource Partner attribute.

Syntax

The GetValue method has the following format:

```
Netegrity::PolicyMgtWSFEDRPAttr->GetValue()
```

Parameters

The GetValue method accepts no parameters.

Return Value

The GetValue method returns one of the following values:

- attribute_value
- **undef** if the call is unsuccessful

Chapter 7: Policy Management Operations

This section contains the following topics:

Initialize a Session

When you create a session, a number of session initialization flags are set to their default values. The following table lists the initialization methods in the PolicyMgtAPI object and their default values:

Method	Default	Description
DisableAudit()	0 (Auditing enabled)	Enables or disables: ■ Auditing of user activity, including authentication, authorization, and administration activities. (Administration activities include changes to the policy store.) ■ Monitoring of user sessions.
DisableManagement WatchDog()	0 (Watchdog enabled)	Enables or disables the SiteMinder management watchdog. The watchdog is used internally and should not be disabled.

Method	Default	Description
Disable Validation()	0 (Validation enabled)	Enables or disables validation of policy store objects.
LoadAgentType Dictionary()	0 (Pre-load dictionary disabled)	Enables or disables the pre-loading of the agent type dictionary.
PreLoadCache()	0 (Pre-load cache disabled)	Enables or disables the pre-loading of SiteMinder caches.

Note: These methods have no effect if called after CreateSession().

Example: Initialize session operations

The following example enables all the session initialization operations that are not enabled by default. If the session is successfully initialized, the script displays the initialization flags:

```
use Netegrity::PolicyMgtAPI;

$username = "adminid";
$password = "adminpwd";

print "\nInitializing and connecting to PolicyMgtAPI...\n";
$policyapi = Netegrity::PolicyMgtAPI->New();
$policyapi->PreLoadCache(1);
$policyapi->LoadAgentTypeDictionary(1);

die "ERROR: Couldn't create session\n" unless ($session != undef);

print "Initialization settings:\n";
print "  Preload cache flag: ".$policyapi->PreLoadCache()."\n";
print "  Disable validation flag: " .
                     $policyapi->DisableValidation()."\n";
print "  Load agent type dictionary flag: " .
                     $policyapi->LoadAgentTypeDictionary()."\n";
print "  Disable audit flag: ".$policyapi->DisableAudit()."\n";
print "  Disable watchdog flag: " .
                     $policyapi->DisableManagementWatchDog()."\n";
```

Create and Manage System Objects

When you initialize a connection to the Policy Server, you create a session object (PolicyMgtSession). You use the session object to create, retrieve, and delete the *system-level objects* that are listed in the System tab of the SiteMinder Administration window.

System objects (also called global objects) have global scope—that is, they are visible across all domains in the policy store. The system objects you can create from session objects include:

- Administrators
- Agents and agent types
- Authentication schemes
- Authentication and authorization maps
- Domains
- ODBC query schemes
- Password policies
- Registration schemes
- User directories

Create Agent Objects

Agent objects are system objects with global scope. The following example uses a session object to create and configure an agent, then prints out all the agent names that are configured in the policy store:

```
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");

$ip="127.0.0.1";
$secret="oursecret";

$agentType=$session->GetAgentType("Web Agent");
$session->CreateAgent("agent1",$agentType,"",$ip,$secret);

@agents=$session->GetAllAgents();
foreach $agent(@agents) {
   print $agent->Name()."\n";
}
```

Note: This example creates a v4.x agent. To create a v5.x or v6.x agent, do not specify a shared secret.

View and Modify Object Properties

After you create an object (or after an object is created in the Administrative UI), you can view and modify the individual properties of the created object by calling the object’s get and set methods.

Typically, an object’s get and set method names represent the name of the property being retrieved or modified, and the methods take a single optional argument. If you supply the argument, you set the property for the object. If you omit the argument, you retrieve the property value without altering it. The method returns the new or existing property value.

In the following example, each user directory is checked for its Maximum Results property—that is, the value that specifies the maximum number of search results to return after a directory search. If the retrieved number is not 25, the script sets the property to 25:

```
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");

$max="25";

@userdirs=$session->GetAllUserDirs();
foreach $userdir(@userdirs) {
   print "\nMax results for directory " . $userdir->Name()."\n";
   if ($userdir->MaxResults() != $max) {
      print "  Updating from " . $userdir->MaxResults()." to " .
                                                  $max . "\n";
      $userdir->MaxResults($max);
   }
   else {
      print "  Max results are correct.\n";
   }
}
```

Objects with Domain Scope

One of the system objects you can create is a domain object (PolicyMgtDomain). Although a domain object itself has global scope, the objects you create and retrieve through a domain object have *domain scope*—that is, they are visible only within the domain and cannot be shared between domains.

Domain objects are listed in the Domains tab of the SiteMinder Administration window.

Objects with domain scope include:

- Policies
- Realms
- Responses and response attributes
- Rules
- User Policies

Retrieve One Object to Create Another

Sometimes, before you can manipulate an object, you must retrieve a higher-level object. That is the case with objects having domain scope.

Here is an example of creating a policy with domain scope. Note that before the policy object can be created, you retrieve the domain object where the policy will reside:

```
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");
$domain=$session->GetDomain("engineering");
$policy=$domain->CreatePolicy("Payroll Policy");
if($policy == undef) {
   print "Couldn't create the policy.\n";
}
else {
   print "Successfully created policy " . $policy->Name() . ".\n";
}
```

Manage an Object’s Properties

You view and set properties for an object with domain scope by using the object’s get and set methods, just as you do for objects with global scope.

Sometimes, the value of an object’s property is not a string or numeric value, but another object. For example, with policies, you add rules and users to the policy, and you set responses for the rules in the policy. All of these values are provided as objects.

Example: Add objects to a policy

In the following example, the policy is configured with a user, rule and response:

```
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");
$domain=$session->GetDomain("engineering");
$policy=$domain->GetPolicy("Payroll Policy");

# Add a user to the policy
$userdir=$session->GetUserDir("Acme North Directory");

@users=$userdir->LookupEntry("uid=ppaycheck");
foreach $user(@users) {
   if ($user->GetPath()=="uid=ppaycheck,ou=HR,o=security.com") {
      $userResult=$policy->AddUser($user);
      $thisUser=$user;
   }
}
if ($userResult != 0) {
   print "Error adding user to policy.\n";
}

# Add a rule to the policy
$realm=$domain->GetRealm("HR");
$rule=$realm->GetRule("Payroll Rule");
$ruleResult=$policy->AddRule($rule);
if ($ruleResult != 0) {
   print "Error adding rule to policy.\n";
}
else
{
   # Set a response for the rule
   $response=$domain->GetResponse("Welcome to Payroll");
   $respResult=$policy->SetResponse($rule,$response);
   if ($respResult != 0) {
      print "Error adding response to policy.\n"
   }
}

print "\nAdded these objects to the policy:\n";
print "  User DN: ".$thisUser->GetPath()."\n" unless $userResult!=0;
print "  Rule: ".$rule->Name()."\n" unless $ruleResult!=0;
print "  Response: ".$response->Name()."\n" unless $respResult!=0;
```

Objects with Domain Scope or Global Scope

Some objects can be created with either domain scope or global scope. Those objects are:

- PolicyMgtPolicy
  - Domain: PolicyMgtDomain->CreatePolicy()
  - Global: PolicyMgtSession->CreateGlobalPolicy()
- PolicyMgtResponse
  - Domain: PolicyMgtDomain->CreateResponse()
  - Global: PolicyMgtSession->CreateGlobalResponse()
- PolicyMgtRule
  - Domain: PolicyMgtRealm->CreateRule()
  - Global: PolicyMgtSession->CreateGlobalRule()
- PolicyMgtGroup
  - Domain: PolicyMgtDomain->CreateResponseGroup()
  - Domain: PolicyMgtDomain->CreateRuleGroup()
  - Global: PolicyMgtSession->CreateAgentGroup()

The following table compares policy, response, and rule objects when they have domain scope and global scope:

Object	**Domain Scope**	**Global Scope**
Policy	Bound to specific users or groups of users.	Bound to all users.
	Individual users can be included in or excluded from the policy.	Users cannot be individually included or excluded.
	Uses domain-specific rules and rule groups, domain-specific responses and response groups, and global responses.	Uses only global rules and global responses.
	Can use variable expressions.	Cannot use variable expressions.
Response	Used in a domain-specific policies.	Used in global or domain-specific policies.

Object	Domain Scope	Global Scope
	Can be a member of a domain-specific response group.	Can be a member of a domain-specific response group. Global response groups are not supported.
	Can use variables-based attributes.	Cannot use variables-based attributes.
Rule	Used in domain-specific policies.	Used in global policies.
	Associated with an agent through a realm.	Associated with a specific agent or agent group. The agent or agent group is specified when the global rule is created.
	The resource filter is bound to a specific realm (realm filter plus rule filter).	The resource filter is absolute (that is, not bound to a realm).
	Fires only for resources defined within a specific domain.	Fires for resources defined within any domain that has global policy processing enabled.
	Can be defined as an access rule or an event rule.	Can be defined as an event rule only (authentication and authorization events).
	Can be a member of a domain-specific rule group.	Can be a member of a domain-specific rule group. Global rule groups are not supported.
All	Created by domain administrators in the context of the specific domain.	Created by system administrators at the system level.

Authorization Variables

An authorization variable is a dynamic object that is resolved to a value during an authorization request. The variables appear within an active expression defined for a policy or a response.

Authorization variables are used as follows:

- With policies, variables are used as authorization constraints. When a user requests access to a resource, and the resource contains an active expression that includes one or more variables, the variables are resolved to values that pertain to the user. The values are then evaluated and used in the decision about whether to authorize the user.

  For example, suppose a policy that protects a bank's credit card application form contains an active expression with a Credit Rating variable and a Salary variable. When a user attempts to access the form, the user is authorized only if his credit rating and salary meet or exceed the minimum values for these variables.

- With responses, variables are used as return values. For example, a response attribute might be configured to return a transaction's tracking number obtained from a remote Web Service.

To use authorization variables, you must have the SiteMinder Option Pack installed.

Configure a Variable for a Particular Variable Type

You create and configure a variable by calling CreateVariable() for a PolicyMgtDomain object.

One of this method's arguments is *definition*. The value of this argument can be a simple string or a set of XML elements, depending on the variable type. Here are the SiteMinder variable types and a description of the *definition* argument for each type:

- **Post**

  The *definition* argument contains the name of a field on an HTML form. In a POST action, the variable value is derived from the value assigned to the field.

- **RequestContext**

  The *definition* argument contains the following XML code:

  ```
  <RequestContextVariableDef>
     <ItemName></ItemName>
  </RequestContextVariableDef>
  ```

The variable value depends upon which of the following attribute names appears within the ItemName element:

- Action. The variable value is the type of action specified in the request (for example, GET or POST).
- Resource. The variable value is the target resource (for example, /directory_name/).
- Server. The variable value is the full server name specified in the request (for example, server.company.com).

- **Static**

  The *definition* argument contains the actual value that will be compared against the user-supplied data at runtime. For example, a Static variable of return type VAR_RETTYPE_DATE might be assigned the string value 2004-01-01. During authorization, this assigned date is compared to a user-supplied date.

- **UserContext**

  The *definition* argument contains some or all of the following XML code:

```
<UserContextVariableDef>
   <ItemName></ItemName>
   <PropertyName></PropertyName>
   <DN></DN>
   <BufferSize></BufferSize>
</UserContextVariableDef>
```

  The variable value is based on an attribute of a user directory connection (such as session ID) or on the contents of the user directory (such as user name). The name of the attribute upon which the variable value is based appears in the XML element ItemName.

  The elements PropertyName, DN, and BufferSize are only used as follows:

  - When ItemName contains DirectoryEntryProperty, elements PropertyName, DN, and BufferSize are used.
  - When ItemName contains UserProperty, elements PropertyName and BufferSize are used.

  For a complete list of the valid ItemName values, see the description of CreateVariable() in the *Policy Management API Reference* (PolicyMgtAPI.htm).

- **WebService**

  The *definition* argument contains the following basic XML structure:

```
<WebServiceVariableDefn xmlns:NeteWS=
                       "http://www.netegrity.com/2003/SM6.0";>
   <NeteWS:RemoteURL></NeteWS:RemoteURL>
   <NeteWS:SSL/>
      <NeteWS:RemoteMethod></NeteWS:RemoteMethod>
      <NeteWS:ResultQuery></NeteWS:ResultQuery>
      <NeteWS:AuthCredentials>
         <NeteWS:Username></NeteWS:Username>
         <NeteWS:Password></NeteWS:Password>
         <NeteWS:Hash></NeteWS:Hash>
      </NeteWS:AuthCredentials>
      <NeteWS:Document>
         <SOAP:Envelope xmlns:SOAP=
                "http://schemas.xmlsoap.org/soap/envelope/";>
            <SOAP:Header></SOAP:Header>
            <SOAP:Body></SOAP:Body>
         </SOAP:Envelope>
      </NeteWS:Document>
</WebServiceVariableDefn>
```

  To retrieve a variable value from a Web Service, the Policy Server sends the Web Service a SOAP request document as specified in the *definition* argument, and then extracts the variable value from the SOAP response.

The following table describes the XML elements used to configure a WebService variable:

Element	**Description**
RemoteURL	The URL to the Web Service that will resolve the WebService variable.
SSL	Specifies that the connection between the Policy Server and the Web Service should use SSL.
RemoteMethod	Set this element to POST.
ResultQuery	The return query, in XPath format. The Policy Server uses this information to search for the variable’s value in the SOAP response document.

Element	**Description**
AuthCredentials	Optionally, specify the user's Web Service credentials through the following elements: ■ Username ■ Password (use either a SHA-1 password digest or a clear-text password) Optionally, use the Hash element to specify that a hash of the password is to be included in the WS-Security password.
Document	Optionally, use this element to define a SOAP header and/or SOAP body through the following elements: ■ Envelope. The SOAP namespace is: http://schemas.xmlsoap.org/soap/envelope ■ Header. A user-defined SOAP header. A WS-Security header is automatically added to it if the user's Web Service credentials are specified. ■ Body. A user-defined SOAP body. Nested variables of type RequestContext, UserContext, Post, and Static can be used inside the header and body. Their values are resolved and substituted before the request document is sent to the remote Web Service. Specify a nested variable as follows: $*variable-name*$

Note: The XML element structures shown above are formatted for legibility. The XML string supplied through the *definition* argument should not be formatted with spaces, tabs, and return characters. For example, a RequestContext variable for a Resource attribute would be passed in *definition* as follows:

```
<RequestContextVariableDef><ItemName>Resource</ItemName></RequestContextVariableD
ef>
```

The following information is required in a call to CreateVariable():

- The user-defined variable name.
- The variable type—for example, Static or ResourceContext.

- The variable definition.
- The data type of the variable value. Valid data type values are:
  - VAR_RETTYPE_BOOLEAN
  - VAR_RETTYPE_NUMBER
  - VAR_RETTYPE_STRING
  - VAR_RETTYPE_DATE

If you have both the optional TransactionMinder product and the Option Pack installed, you can use the following types of variables:

- SAMLAssertion
- Transport
- XMLAgent
- XMLBody
- XMLEnvironment

You cannot create variables of these types with the Command Line Interface. You can only do so using the Administrative UI.

Example: Create a ResourceContext Variable

The following example creates the variable MyVar as a ResourceContext variable. The variable value is the resource that is being protected (for example, /directory_name/):

```
use Netegrity::PolicyMgtAPI;

$pmgtapi=Netegrity::PolicyMgtAPI->New();
$session=$pmgtapi->CreateSession("adminid", "adminpwd");

$dom=$session->GetDomain("MyDomain");
$varName="MyVar";
$varType=$session->GetVariableType("RequestContext");
$varDef="<RequestContextVariableDef><ItemName>Resource</ItemName>
    </RequestContextVariableDef>";

$vr=$dom->CreateVariable($varName,$varType,$varDef,
    VAR_RETTYPE_STRING);

if ($vr==undef) {
    print "Create operation failed.";
    }
else {
    print "Created variable " . $varName;
    }
```

Save Changes to Objects

When you make changes to an instance of the following objects, you must call the object's Save() method to save the changes. Call Save() once after making any changes to these objects:

- PolicyMgtAffiliate
- PolicyMgtAuthScheme
- PolicyMgtPwdPolicy
- PolicyMgtSAMLAffiliation
- PolicyMgtSAMLServiceProvider
- PolicyMgtSharedSecretPolicy

The following example changes properties for an authentication scheme object:

```
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");
$authscheme=$session->GetAuthScheme("HTML Authentication");

#Specify the attribute list
$attrList="AL=PASSWORD,SSN,age,zipcode;";
#Specify the new credentials collector
$target="http://my.server.com/siteminderagent/forms/customlogin.fcc";

#Make the changes and save them
$authscheme->ProtectionLevel(10);
$authscheme->CustomParam($attrList.$target);
$authscheme->Save();
```

Modify a Password Policy

Suppose a company has different password policies defined at different sites, and it decides to standardize certain password policy requirements. The following script can be executed for each site to set the new password policy definitions:

```
use Netegrity::PolicyMgtAPI;

$policymgtapi = Netegrity::PolicyMgtAPI->New();
$session = $policymgtapi->CreateSession("adminid", "adminpwd");

@pwdpols=$session->GetAllPwdPolicies();
foreach $pwdpol(@pwdpols) {
   $pwdpol->Description("Standardized settings 4/15/02");
   print "\n\nPwd Policy: " . $pwdpol->Name();
   print "\nUpdated the following settings:";
   print "\n  Minimum length:\t".$pwdpol->PwdMinLength(6);
   print "\n  Maximum length:\t".$pwdpol->PwdMaxLength(18);
   print "\n  Minimum digits:\t".$pwdpol->PwdMinNumbers(3);
   print "\n  Allowable failures:\t".$pwdpol->MaxLoginFailures(3);
   print "\n  Days before reuse:\t".$pwdpol->PwdReuseDelay(360);
   print "\n  Changes before reuse:\t".$pwdpol->PwdReuseCount(6);
   $pwdpol->Save();
}
```

Manage Password State

Password state refers to activities relating to a given user's password—for example, the last time the password was changed, and the last time the password was used to log in the user.

To retrieve an existing PolicyMgtUserPasswordState object for a user, or to set a new password state object with any attribute changes, call PolicyMgtUser->UserPasswordState().

The table that follows lists the password state attributes you can access for a given user, and the method used to set or retrieve an attribute value. All methods are in the object PolicyMgtUserPasswordState, unless otherwise noted.

Password State Attribute	**Method**	**Description**
Login failures	LoginFailures()	Sets or retrieves the number of times the user failed to log in since the user's last successful login.
Last login time	LastLoginTime()	Sets or retrieves the time the user last logged in successfully.
Previous login time	PrevLoginTime()	Sets or retrieves the next-to-last time the user logged in successfully.
Disabled time	DisabledTime()	Sets or retrieves the time the user object was disabled.
Password history	PolicyMgtUser-> UserPasswordState()	Optionally, clears the user's password history when setting the password state object for the user. You cannot retrieve password history or set password history entries.
Last password change time	LastPWChangeTime()	Sets or retrieves the time the user's password was last changed.

If you change a password state attribute, the change applies to the current password state object only. To apply the change to a password state object that may be subsequently retrieved, pass the current password state object in a call to PolicyMgtUser->UserPasswordState(). This method sets a new password state object containing the attribute values passed into the method.

For example, the code fragment below performs the following operations:

1. Retrieves the password state object, $passwordstate, for the current user, $user[0].
2. Sets the login failures attribute to 3 in this instance of the password state object.
3. Calls UserPasswordState() to clear the user’s password history and set a new password state object for the user with the new history and login failures attributes.

```
$passwordstate = $user[0]->UserPasswordState();
$passwordstate->LoginFailures(3);
$user[0]->UserPasswordState($passwordstate, 1);
```

Create Responses and Response Attributes

The following script creates a response and response attribute. Note that GetAgentType() needs to be called to retrieve an agent type object for the call to CreateResponse():

```
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");

$domain = $session->GetDomain("Acme North Domain");
$agenttype=$session->GetAgentType("Web Agent");
$response=$domain->CreateResponse("Welcome to Payroll",$agenttype);
if ($response==undef) {
   print "\nCouldn't create the response.";
}
else {
   $attr=$response->CreateAttribute("WebAgent-HTTP-Cookie-Variable",
                                    "cookiename=mycookie");
   if ($attr==undef) {
      print "\nCouldn't create attribute for ".$response->Name();
   }
   else {
      print "\nCreated response " . $response->Name();
      print "\nCreated attribute " . $attr->GetValue();
   }
}
```

Update Realms with a New Authentication Scheme

The following example assigns the authentication scheme HTML Authentication to all the realms in the domain Acme North Domain:

```
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");

$domain = $session->GetDomain("Acme North Domain");
$authscheme=$session->GetAuthScheme("HTML Authentication");

@realms=$domain->GetAllRealms();
print "Updating these realms to auth scheme ".$authscheme->Name();
foreach $realm(@realms) {
   $name=$realm->Name();
   if ($realm->AuthScheme($authscheme)==undef) {
      print "\n  Couldn't update realm " . $name;
   }
   else {
      print "\n  Realm ". $name;
   }
}
```

View Default Values for an Authentication Scheme Template

The following script displays the default values for the authentication scheme template HTML Form Template.

Creating an authentication scheme is not necessary. The script uses GetAuthScheme() to retrieve an authentication scheme object for the template, then prints the objects's default properties:

```
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");

#Retrieve the object for the authentication scheme template
$template=$session->GetAuthScheme("HTML Form Template");

print "\nDefault values for template " . $template->Name();
print "\n  Type:\t" . $template->Type()->Name();
print "\n  Description:\t" . $template->Description();
print "\n  Protection level:\t" . $template->ProtectionLevel();
print "\n  Library:\t" . $template->CustomLib();
print "\n  Parameter:\t" . $template->CustomParam();
print "\n  Shared secret:\t" . $template->CustomSecret();
print "\n  Is template?:\t" . $template->IsTemplate();
print "\n  Is used by admin?:\t" . $template->IsUsedByAdmin();
print "\n  Save credentials?:\t" . $template->SaveCredentials();
print "\n  Is Radius:\t" . $template->IsRadius();
print "\n  Ignore pwd ck?:\t" . $template->IgnorePwd();
```

Create an Authentication Scheme

When you create an authentication scheme, you base the scheme on an authentication scheme template. To do so, you first retrieve an existing template and then specify the template object in the call to CreateAuthScheme().

The following example creates an authentication scheme based on the template HTML Form Template and accepts all defaults:

```
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");

#Retrieve the object for the authentication scheme template
$template=$session->GetAuthScheme("HTML Form Template");

#Create the authentication scheme
$scheme=$session->CreateAuthScheme("HTML Authentication",$template);
if ($scheme == undef) {
   print "\nCouldn't create the authentication scheme.";
}
else {
   print "\nCreated authentication scheme ".$scheme->Name();
}
```

Modify the Shared Secret Rollover Policy

A shared secret is a text string known only to a trusted host and the policy store domain where the host is registered. The shared secret is used to authenticate the identity of the trusted host when it makes a secure connection to the Policy Server.

The shared secret rollover feature provides a mechanism to periodically change the shared secret automatically.

Using the Scripting Interface for Perl, you can:

- Call GetSharedSecretPolicy() to retrieve an existing shared secret rollover policy object.
- Modify the retrieved PolicyMgtSharedSecretPolicy object as follows:
  - Modify the rollover frequency. This setting specifies how often rollover should occur over a given rollover period (see the next item).
  - Modify the rollover period (hourly, daily, weekly, monthly). For example, with a rollover frequency of 3 and a daily rollover period, the shared secret is automatically changed every three days.
  - Enable or disable the rollover feature. If the shared secret rollover policy is enabled, rollover must also be enabled for any trusted host whose shared secret needs to be synchronized with the rollover policy's shared secret. You can enable rollover for a trusted host object by calling RolloverEnabled().

Write a Domain and Realm Report to a File

The following script writes the names of all policy store domains and top-level realms to a text file:

```
use Netegrity::PolicyMgtAPI;

$destFile="DomainsRealms.txt";
open(DEST,">".$destFile) || die "Open file error: $!";
print DEST "Domains and Domain Realms for Acme North Site\n";
print DEST "Printed " . scalar(localtime)."";

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");

@domains=$session->GetAllDomains();
foreach $domain(@domains) {
   print DEST "\n\nDomain " . $domain->Name() . ":";
   @realms=$domain->GetAllRealms();
   foreach $realm(@realms) {
      print DEST "\n  Realm " . $realm->Name();
   }
}

print "\nDomain and realm report written to " . $destFile."\n";
```

Disable Authentication and Authorization Event Processing

The following script checks all the rules for each realm in the policy store. If no rules in a realm are triggered by authentication events, the script then checks whether authentication event processing is enabled for the realm. If it is, the script disables authentication event processing for the realm. The script performs the same checks for authorization events.

To simplify the example, realms in the domains are known not to have child realms.

```
use Netegrity::PolicyMgtAPI;

$policyapi = Netegrity::PolicyMgtAPI->New();
$session = $policyapi->CreateSession("adminid", "adminpwd");

$auAction=0; # Initialize flag for authentication actions
$azAction=0; # Initialize flag for authorization actions
$auChange="";# Realms with a changed auth event processing property
$azChange="";# Realms with a changed az event processing property

@domains=$session->GetAllDomains();
foreach $domain(@domains) {
   @realms=$domain->GetAllRealms();
   foreach $realm(@realms) {
      @rules=$realm->GetAllRules();
      foreach $rule(@rules) {
         if ($rule->Action()=~/OnAuth./ ) {
            $auAction=1;
         }
         if ($rule->Action()=~/OnAccess./ ) {
            $azAction=1;
         }
      }
      if($auAction==0) {
         if($realm->ProcessAuEvents()==1) {
            $realm->ProcessAuEvents(0);
            $auChange=$auChange.$domain->Name().": ";
            $auChange=$auChange.$realm->Name()."\n";
         }
      }
      else { $auAction=0; }
      if($azAction==0) {
         if($realm->ProcessAzEvents()==1) {
            $realm->ProcessAzEvents(0);
            $azChange=$azChange.$domain->Name().": ";
            $azChange=$azChange.$realm->Name()."\n";
         }
      }
      else { $azAction=0; }
   }
}

if ($auChange ne "") {
   print "Stopped auth event processing for these realms:\n";
   print $auChange . "\n\n";
}
if ($auChange ne "") {
   print "Stopped az event processing for these realms:\n";
   print $azChange . "\n";
}
```

Manage Policy Server Load Distribution

To help prevent service interruptions, SiteMinder includes a failover feature. If the primary Policy Server fails and failover is enabled, a backup Policy Server takes over policy operations. Beginning with SiteMinder v6.0, failover can occur not only between Policy Servers, but between groups, or *clusters*, of Policy Servers.

The cluster functionality also improves server performance by providing dynamic load balancing between the servers in a cluster. With dynamic load balancing, policy operations are automatically distributed between the available servers in a cluster according to the performance capabilities of each server.

A host configuration can be associated with one or more Policy Servers, or with one or more clusters of Policy Servers, depending on how you define your host configuration object:

- **Clustered servers**
  - Call AddCluster() for each cluster to create in the host configuration.
  - In the PolicyMgtCluster object returned from AddCluster(), call AddServer() to populate the cluster with servers.

  Behavior: Failover occurs between clusters of servers if multiple clusters are defined. Also, requests to servers within a cluster are sent according to the improved performance-based load-balancing techniques introduced with Agent API v6.0.

- **Non-clustered servers**
  - Call AddServer() for each non-clustered server to create in the host configuration.

  Behavior: Behavior is the same as in v5.x installations—that is, you can enable failover among the servers associated with a host configuration (set EnableFailover() to 1), or you can enable round-robin behavior among the servers (set EnableFailover() to 0).

  When round-robin behavior is enabled, the improved performance-based load-balancing techniques introduced with Agent API are used.

Note: You cannot mix clustered and non-clustered servers in a host configuration.

Cluster Configuration

A cluster is stored in a host configuration object. Cluster failover occurs according to the following configuration values set in PolicyMgtHostConfig:

- **Failover threshold**. The minimum percentage of servers within a cluster that must be available for Policy Server requests. If the number of available servers falls below the threshold, failover to the next cluster occurs.

  The failover threshold percentage applies to all clusters associated with the host configuration object.

  To determine the number of servers that the percentage represents in any given cluster, multiply the threshold percentage by the number of servers in the cluster, rounding up to the next highest integer. For example:

  - With a 60-percent failover threshold for a cluster of five servers, failover to the next cluster occurs when the number of available servers in the cluster falls below 3.
  - With a 61-percent failover threshold for the same cluster, failover occurs when the number of available servers falls below 4.

  Set through: FailoverThreshold().

- **Server timeout**. The maximum time an agent will wait for a response from a server. If the wait time exceeds the server timeout value, the server is considered inactive, and failover to the next server occurs.

  If a server timeout occurs within a cluster, and the timeout causes the cluster's failover threshold to be exceeded, failover to the next cluster occurs.

  Set through: RequestTimeout().

- **Sequence of cluster failover**. When cluster failover occurs, SiteMinder sends subsequent Policy Server requests to the next cluster in the cluster sequence. Cluster sequence is determined by the order of cluster objects in the cluster array.

  Add clusters through: AddCluster(). The newly added cluster is added to the end of the cluster array.

  Retrieve the cluster array through: GetAllClusters().

  The order in which you add clusters to a host configuration object determines the failover sequence. The first cluster you add (that is, the first cluster in the cluster array) is the *primary cluster*. This is the cluster that is used as long as the number of available servers in the cluster does not fall below the failover threshold. If there are not enough available servers in the primary cluster, failover to the next cluster occurs—that is, to the second cluster that was added to the host configuration object. If that cluster also fails, failover to the third cluster added to the host configuration object occurs, and so on.

When All Clusters Fail

If the number of available servers falls below the failover threshold in all clusters within a host configuration, policy operations do not stop. Requests are sent to the first cluster in the cluster sequence that has at least one available server.

For example, suppose a host configuration has two clusters—C1 containing three servers, and C2 containing five servers. The failover threshold for the host configuration is set at 60 percent. The following table shows the minimum number of servers that must be available within each cluster:

Cluster	**Servers in Cluster**	**Percentage Failover Threshold**	**Numeric Failover Threshold (Minimum Available Servers)**
C1	3	60	1
C2	5	60	3

If the number of available servers falls below the threshold in each cluster, so that C1 has no available servers and C2 has just two, the next incoming request will be dispatched to a C2 server with the best response time. After at least two of the three C1 servers are repaired, subsequent requests are load-balanced among the available C1 servers.

Agent API v6 is backwards-compatible with Agent API v5, allowing complete interoperability between v5/v6 agents and the v5/v6 Agent APIs.

Example: Create a host configuration

The following code creates a host configuration object and adds a number of clusters and servers:

```
use Netegrity::PolicyMgtAPI;

# Initialize the Policy Management API and create a Host Config object.
$pmgtapi = Netegrity::PolicyMgtAPI->New();
$session = $pmgtapi->CreateSession("SiteMinder", "password");
$hostconf = $session->CreateHostConfig("host", "description",
                                              false, 2, 2, 1, 30);
# Add two non-cluster servers. The Az, Auth and Acct ports are
# specified.
$hostconf->AddServer("1.1.1.1", 44443, 44442, 44441);
$hostconf->AddServer("2.2.2.2", 44443, 44442, 44441);

# Add two clusters with two servers in each cluster. One Policy
# Server port number is specified.
$clusterconf1 = $hostconf->AddCluster();
$clusterconf1->AddServer("1.1.1.1", 44443);
$clusterconf1->AddServer("2.2.2.2", 44443);

$clusterconf2 = $hostconf->AddCluster();
$clusterconf2->AddServer("3.3.3.3", 44443);
$clusterconf2->AddServer("4.4.4.4", 44443);

# Print configuration of all non-cluster servers in the Host
# Config object
@servers = $hostconf->GetAllServers();
foreach $server (@servers) {
    $address = $server->GetServerAddress();
    @ports = $server->GetServerPort();
    print("Server: $address,@ports[0],@ports[1],@ports[2]\n");
}
```

```
# Print all cluster servers
@clusters = $hostconf->GetAllClusters();
foreach $cluster (@clusters) {
    $num++;
    foreach $server ($cluster->GetAllServers()) {
    $address = $server->GetServerAddress();
    $port = $server->GetServerPort();
    print("Cluster $num Server: $address,$port\n");
}
}

# Remove all clusters and non-cluster servers from host configuration
$hostconf->RemoveAllClusters();
$hostconf->RemoveAllServers();
```

Appendix A: Command Line Interface Restrictions

This appendix summarizes the operations that you cannot perform using the Command Line Interface, but that you can perform in the Administrative UI.

In the Command Line Interface, you cannot:

- Provide custom agent support for the resolved/unresolved data lists. The feature was introduced in SiteMinder v5.x.
- Create an administrator for a particular domain. Administrators you create with the Perl Scripting Interface have system-wide scope.
- Create a realm that is configured for the persistent session feature. This feature was introduced in SiteMinder v5.x.
- Create a child realm under another child realm. You can create a realm under a domain, and one level of child realms under a realm.
- Create a response attribute of type User Attribute or DN Attribute. Static response attributes can be created with a call to the CreateAttribute method, and Active Responses with a call to the CreateActiveAttribute method.
- Define periods within a week-long time grid that specify when a policy is active. When you create a policy through the Command Line Interface, the policy defaults to being active for the entire week.

Appendix B: Property References

This section contains the following topics:

SAML 2.0 Properties

This section contains an alphabetized reference of the SAML 2.0 metadata properties supported in the Perl Policy Management API.

The properties apply to one or more of the following SAML 2.0 objects:

- SAML 2.0 affiliation.

  A SAML 2.0 affiliation is a set of entities that share a single federated namespace of unique Name IDs for principals.

- SAML 2.0 authentication scheme and its associated Identity Provider definition.

  The Identity Provider creates SAML assertions for the Service Provider that configured the Identity Provider and its associated SAML 2.0 authentication scheme.

- Service Provider.

  A Service Provider provides services (such as access to applications and other resources) to principals within a federation.

  A Service Provider uses a SAML 2. 0 authentication scheme to transparently validate a principal based on the information in a SAML assertion. The assertion is supplied by the Identity Provider associated with the authentication scheme.

Reference Notes

- All property values are provided as strings.
- Unless otherwise specified, properties have the following maximum lengths:
  - URIs and URLs must be less than 1,024 characters
  - All other strings must not exceed 255 characters

SAML_AFFILIATION

Required

No

Default

None

Description

The SAML 2.0 affiliation to associate with this object.

Service Providers share the Name ID properties across the affiliation. IdentityProviders share the user disambiguation properties across the affiliation.

A Service Provider or Identity Provider can belong to only one SAML 2.0 affiliation.

If a SAML affiliation is specified, the NAMEID properties (for example, SAML_SP_NAMEID_FORMAT) are not used. SiteMinder uses the NAMEID information in the specified affiliation.

An Identity Provider is assigned to an affiliation through its associated SAML 2.0 authentication scheme.

For more information about SAML 2.0 affiliations, see the description of the CreateSAMLAffiliation method.

SAML_AUDIENCE

Required

Yes

Default

None

Description

The URI of the expected audience for a Service Provider. The audience expected by the Service Provider must match the audience specified in the assertion.

The audience might also be sent in an authentication request.

SAML_DESCRIPTION

Required

No

Default

None

Description

A brief description of the affiliation, authentication scheme, or Service Provider object.

SAML_DISABLE_SIGNATURE_PROCESSING

Required

No

Default

0

Description

Specifies whether to disable all signature validation, including signing.

It may be useful to disable signature validation during the initial setup of a provider and during debugging. During normal runtime, this property should be set to 0 (signature processing enabled).

Valid values: 0 (false) and 1 (true).

SAML_DSIG_ALGO

Required

No

Default

1

Description

Specifies the XML Federation Signature algorithm with one of the following values:

1 = RSAwithSHA1

2 = RSAwithSHA256

SAML_DSIG_VERINFO_ISSUER_DN

Required

With SAML 2.0 Authentication Schemes:

Required only if SAML_DISABLE_SIGNATURE_PROCESSING is 0 and one or both of the following are 1:

- SAML_SLO_REDIRECT_BINDING
- SAML_ENABLE_SSO_POST_BINDING

With Service Providers:

Required only if SAML_DISABLE_SIGNATURE_PROCESSING is 0 and one or both of the following are 1:

- SAML_SLO_REDIRECT_BINDING
- SAML_SP_REQUIRE_SIGNED_AUTHNREQUESTS

Default

None

Description

If the certificate of the Service Provider is not provided inline, this value is used along with SAML_DSIG_VERINFO_SERIAL_NUMBER to locate the certificate in the key store.

SAML_DSIG_VERINFO_SERIAL_NUMBER

Required

With SAML 2.0 Authentication Schemes:

Required only if SAML_DISABLE_SIGNATURE_PROCESSING is 0 and one or both of the following are 1:

- SAML_SLO_REDIRECT_BINDING
- SAML_ENABLE_SSO_POST_BINDING

With Service Providers:

Required only if SAML_DISABLE_SIGNATURE_PROCESSING is 0 and one or both of the following are 1:

- SAML_SLO_REDIRECT_BINDING
- SAML_SP_REQUIRE_SIGNED_AUTHNREQUESTS

Default

None

Description

If the certificate of the Service Provider is not provided inline, this value is used along with SAML_DSIG_VERINFO_ISSUER_DN to locate the certificate in the key store.

SAML_ENABLE_SSO_ARTIFACT_BINDING

Required

No

Default

0

Description

Specifies whether artifact binding is supported by the Service Provider and enabled by the Identity Provider.

Valid values: 0 (false) and 1 (true).

SAML_ENABLE_SSO_POST_BINDING

Required

No

Default

0

Description

Specifies whether HTTP POST binding is supported by the Service Provider and enabled by the Identity Provider.

Valid values: 0 (false) and 1 (true).

See also SAML_DSIG_VERINFO_ISSUER_DN and SAML_DSIG_VERINFO_SERIAL_NUMBER.

SAML_ENABLED

Required

No

Default

1

Description

Specifies whether the Service Provider is activated.

Valid values: 0 (false) and 1 (true).

SAML_IDP_AD_SEARCH_SPEC

Required

No

Default

None

Description

Search specification for AD directories.

If user disambiguation is being performed on a user in an AD directory, but no AD search specification has been provided for this property, the default search specification defined on the SiteMinder User Directory Properties dialog is used.

Assigning a search specification to this property is recommended for the following reasons:

- When using the default search specification, the Policy Server might duplicate login ID prefixes and suffixes that are already present in the ID extracted from the assertion.
- If you are extending the functionality of a SAML 2.0 authentication scheme with a custom message consumer plugin, the plugin will not be called in the user disambiguation phase if the Policy Server disambiguates the user with the default search specification defined on the SiteMinder User Directory Properties dialog. For more information, see SAML_IDP_PLUGIN_CLASS.

When defined for an affiliation, the search specification is shared by all Identity Providers across the affiliation.

SAML_IDP_ARTIFACT_RESOLUTION_DEFAULT_SERVICE

Required

Yes, if SAML_ENABLE_SSO_ARTIFACT_BINDING is 1

Default

None

Description

A URL specifying the default artifact resolution service for the Identity Provider.

SAML_IDP_BACKCHANNEL_AUTH_TYPE

Required

No

Default

0

Description

Specifies the type of authentication to use on the back channel. Valid values:

- 0. Basic - Uses the specified Service Provider Name and password for authentication.
- 1. Client Cert - Uses the specified Service Provider ID and password to look up the certificate in the keystore.
- 2. No Auth - No authentication is required.

SAML_IDP_CUSTOM_SEARCH_SPEC

Required

No

Default

None

Description

Search specification for custom user directories. If user disambiguation is being performed on a user in a custom directory, but no search specification is provided, the default search specification defined on the SiteMinder User Directory Properties dialog is used.

When defined for an affiliation, the search specification is shared by all Identity Providers across the affiliation.

If you are extending the functionality of a SAML 2.0 authentication scheme with a custom message consumer plugin, the plugin will not be called in the user disambiguation phase if the Policy Server disambiguates the user with the default search specification defined on the SiteMinder User Directory Properties dialog. For more information, see SAML_IDP_PLUGIN_CLASS.

SAML_IDP_LDAP_SEARCH_SPEC

Required

No

Default

None

Description

Search specification for LDAP directories.

If user disambiguation is being performed on a user in an LDAP directory, but no search specification has been provided for this property, the default search specification defined on the SiteMinder User Directory Properties dialog is used.

Assigning a search specification to this property is recommended for the following reasons:

- When using the default search specification, the Policy Server might duplicate login ID prefixes and suffixes that are already present in the ID extracted from the assertion.
- If you are extending the functionality of a SAML 2.0 authentication scheme with a custom message consumer plugin, the plugin will not be called in the user disambiguation phase if the Policy Server disambiguates the user with the default search specification defined on the SiteMinder User Directory Properties dialog. For more information, see SAML_IDP_PLUGIN_CLASS.

When defined for an affiliation, the search specification is shared by all Identity Providers across the affiliation.

SAML_IDP_ODBC_SEARCH_SPEC

Required

No

Default

None

Description

Search specification for ODBC directories.

If user disambiguation is being performed on a user in an ODBC directory, but no ODBC search specification has been provided for this property, the default search specification defined on the SiteMinder User Directory Properties dialog is used.

Assigning a search specification to this property is recommended for the following reasons:

- When using the default search specification, the Policy Server might duplicate login ID prefixes and suffixes that are already present in the ID extracted from the assertion.
- If you are extending the functionality of a SAML 2.0 authentication scheme with a custom message consumer plugin, the plugin will not be called in the user disambiguation phase if the Policy Server disambiguates the user with the default search specification defined on the SiteMinder User Directory Properties dialog. For more information, see SAML_IDP_PLUGIN_CLASS.

When defined for an affiliation, the search specification is shared by all Identity Providers across the affiliation.

SAML_IDP_PASSWORD

Required

Yes, if SAML_IDP_BACKCHANNEL_AUTH_TYPE is set to 0 or 1

Default

None

Description

The password to use for the back-channel authentication. The password is only used with the back-channel authentication types Basic and Client Cert.

SAML_IDP_PLUGIN_CLASS

Required

No

Default

None

Description

The fully qualified name of a Java class that extends the functionality of this SAML 2.0 authentication scheme. The custom functionality is provided by an implementation of the interface MessageConsumerPlugin.java.

Authentication has two phases-user disambiguation and user authentication (validation of the disambiguated user's credentials).

If a plugin is configured for the authentication scheme, it is called as follows:

- During user disambiguation, if the authentication scheme cannot disambiguate the user.

Note: The plugin is not called in this phase if a search specification is not provided for the user directory where disambiguation is to occur (for example, SAML_IDP_LDAP_SEARCH_SPEC for an LDAP directory). In this case, the Policy Server performs the disambiguation, not the authentication scheme.

- At the end of the default authentication phase, even if the user is validated successfully.

A SAML 2.0 authentication scheme can be extended by only one message consumer plugin.

SAML_IDP_PLUGIN_PARAMS

Required

No

Default

None

Description

Parameters to pass into the custom authentication scheme extension specified in SAML_IDP_PLUGIN_CLASS.

The syntax of the parameter string is determined by the custom object.

SAML_IDP_REDIRECT_MODE_FAILURE

Required

No

Default

0

Description

The redirection mode for SAML_IDP_REDIRECT_URL_FAILURE. Valid values:

- 0. 302 No Data - HTTP 302 redirection. The URL for the target resource and the reason for the authentication failure are appended to the redirection URL. The SAML 2.0 Response message passed to the authentication scheme is not included.
- 1. Http Post. - HTTP POST redirection. The SAML 2.0 Response message passed to the authentication scheme and the Identity Provider's ID are generated by an HTTP form.

SAML_IDP_REDIRECT_MODE_INVALID

Required

No

Default

0

Description

The redirection mode for SAML_IDP_REDIRECT_URL_INVALID. Valid values:

- 0. 302 No Data - HTTP 302 redirection. The URL for the target resource and the reason for the authentication failure are appended to the redirection URL. The SAML 2.0 Response message passed to the authentication scheme is not included.
- 1. Http Post. - HTTP POST redirection. The SAML 2.0 Response message passed to the authentication scheme and the Identity Provider's ID are generated by an HTTP form.

SAML_IDP_REDIRECT_MODE_USER_NOT_FOUND

Required

No

Default

0

Description

The redirection mode for SAML_IDP_REDIRECT_URL_USER_NOT_FOUND. Valid values:

- 0. 302 No Data - HTTP 302 redirection. The URL for the target resource and the reason for the authentication failure are appended to the redirection URL. The SAML 2.0 Response message passed to the authentication scheme is not included.
- 1. Http Post. - HTTP POST redirection. The SAML 2.0 Response message passed to the authentication scheme and the Identity Provider's ID are generated by an HTTP form.

SAML_IDP_REDIRECT_URL_FAILURE

Required

No

Default

None

Description

The redirection URL to use when the authentication information passed to the authentication scheme is not accepted to authenticate the user.

SAML_IDP_REDIRECT_URL_INVALID

Required

No

Default

None

Description

The redirection URL to use when the authentication information passed to the authentication scheme is not formatted according to the SAML 2.0 standard.

SAML_IDP_REDIRECT_URL_USER_NOT_FOUND

Required

No

Default

None

Description

The redirection URL to use in either of these circumstances:

- The authentication scheme cannot obtain a login ID from the SAML 2.0 Response message passed to it.
- The authentication scheme cannot find the user in the user directory.

If you are extending the functionality of a SAML 2.0 authentication scheme with a custom message consumer plugin, the plugin will not be called in the user disambiguation phase if the Policy Server disambiguates the user with the default search specification defined on the SiteMinder User Directory Properties dialog. For more information, see SAML_IDP_PLUGIN_CLASS.

SAML_IDP_REQUIRE_ENCRYPTED_ASSERTION

Required

No

Default

0

Description

Specifies whether the assertion selected for authentication must be encrypted. If this property is 1 and the authentication scheme is passed an unencrypted assertion, the assertion cannot be authenticated.

Valid values: 0 (false) and 1 (true).

SAML_IDP_REQUIRE_ENCRYPTED_NAMEID

Required

No

Default

0

Description

Specifies whether the Name ID of the principal contained in the assertion must be encrypted. If this property is 1 and the the Name ID is not encrypted, the assertion cannot be authenticated.

Valid values: 0 (false) and 1 (true).

SAML_IDP_SAMLREQ_ATTRIBUTE_SERVICE

Required

No

Default

None

Description

The URL of the Attribute Service on the Attribute Authority.

SAML_IDP_SAMLREQ_ENABLE

Required

Yes

Default

0

Description

Indicates whether the SAML Requester is enabled.

Valid values: 0 (false) and 1 (true).

SAML_IDP_SAMLREQ_GET_ALL_ATTRIBUTES

Required

No

Default

0

Description

Indicates whether the query sent to the Attribute Authority should contain no attributes. This is a short-hand for the Attribute Authority to return all defined attributes.

SAML_IDP_SAMLREQ_NAMEID_ALLOW_NESTED

Required

No

Default

0

Description

Indicates whether nested groups are allowed when selecting a DN attribute for the name identifier.

Valid values: 0 (false) and 1 (true).

SAML_IDP_SAMLREQ_NAMEID_ATTR_NAME

Required

Yes when NameIdTYpe is set to 1 or 2.

Default

None

Description

The attribute name (user or DN) that holds the identifier name when NameIdType is set to 1 or 2.

SAML_IDP_SAMLREQ_NAMEID_DN_SPEC

Required

Yes when NamedIdTYpe is set to 2.

Default

None

Description

The DN specification used when the NameIdType is set to 2.

SAML_IDP_SAMLREQ_NAMEID_FORMAT

Required

No

Default

None

Description

The URI for a SAML 2.0 name identifier.

SAML_IDP_SAMLREQ_NAMEID_STATIC

Required

Yes when NameIdType is set to 0.

Default

None

Description

The static text to be used when NameIdType is set to 0.

SAML_IDP_SAMLREQ_NAMEID_TYPE

Required

No

Default

1 (user attribute)

Description

Represents the type of the name identifier.

Valid values: 0 (static text), 1 (user attribute), and 2 (DN attribute).

SAML_IDP_SAMLREQ_REQUIRE_SIGNED_ASSERTION

Required

No

Default

0

Description

Indicates whether the assertion returned in response to an <AttributeQuery> must be signed.

Valid values: 0 (false) and 1 (true).

SAML_IDP_SAMLREQ_SIGN_ATTRIBUTE_QUERY

Required

No

Default

0

Description

Indicates whether the attribute query must be signed.

Valid values: 0 (false) and 1 (true).

SAML_IDP_SIGN_AUTHNREQUESTS

Required

No

Default

0

Description

Specifies whether authentication requests will be signed.

Valid values: 0 (false) and 1 (true).

SAML_IDP_SPID

Required

Yes

Default

None

Description

The unique provider ID of the Service Provider being protected by this authentication scheme.

SAML_IDP_SPNAME

Required

Yes, if SAML_IDP_BACKCHANNEL_AUTH_TYPE is set to 0 or 1

Default

None

Description

The name of the Service Provider involved in the back-channel authentication. The Service Provider name is used with the back-channel authentication types Basic and Client Cert.

SAML_IDP_SSO_DEFAULT_SERVICE

Required

Yes

Default

None

Description

The URL of the Identity Provider's single sign-on service, for example:

http://mysite.netegrity.com/affwebservices/public/saml2sso

SAML_IDP_SSO_ENFORCE_SINGLE_USE_POLICY

Required

No

Default

1

Description

Specifies whether to enforce a single-use policy for HTTP POST binding.

Setting this property to 1 (the default) ensures that an assertion cannot be ``replayed'' to a Service Provider site to establish a second session, in accordance with SAML POST-specific processing rules.

The single-use policy requirement is enforced even in a clustered Policy Server environment with load-balancing and failover enabled.

Valid values: 0 (false) and 1 (true).

SAML_IDP_SSO_REDIRECT_MODE

Required

No

Default

0

Description

Specifies the method by which response attribute information is passed when the user is redirected to the target resource.

A response passes user attributes, DN attributes, static text, or customized active responses from the Policy Server to a SiteMinder Agent after the Agent isseus a login or authorization request. For more information about response attributes, see CreateAttribute().

Valid values:

- 0. 302 No Data - No response attributes are passed.
- 1. 302 Cookie Data - Response attributes are set as HTTP cookie data. Attribute cookies issued by the authentication scheme are unencrypted.
- 2. Server Redirect - Response attributes are passed as a HashMap object.

Server-side redirects allow passing information to an application within the server application itself. Response attribute data is never sent to the user's browser. This redirection method is part of Java Servlet specification and is supported by all standards-compliant servlet containers.

SAML_IDP_SSO_TARGET

Required

No

Default

None

Description

The URL of the target resource at the Service Provider site. For example, the target might be a web page or an application.

SAML_IDP_WINNT_SEARCH_SPEC

Required

No

Default

None

Description

Search specification for WinNT directories. If user disambiguation is being performed on a user in a WinNT directory, but no search specification is provided, the default search specification defined on the SiteMinder User Directory Properties dialog is used.

When defined for an affiliation, the search specification is shared by all Identity Providers across the affiliation.

If you are extending the functionality of a SAML 2.0 authentication scheme with a custom message consumer plugin, the plugin will not be called in the user disambiguation phase if the Policy Server disambiguates the user with the default search specification defined on the SiteMinder User Directory Properties dialog. For more information, see SAML_IDP_PLUGIN_CLASS.

SAML_IDP_XPATH

Required

No

Default

None

Description

The XPath query that extracts the user's login ID from an assertion. The login ID is then used to disambiguate the user.

By default, if no XPath is provided, an attempt is made to extract the login ID from the Assertion/Subject/NameID element of the SAML 2.0 Response message.

Once successfully extracted, the login ID is inserted into the search string specified for the user directory, and the disambiguation phase begins.

When defined for an affiliation, the XPath is shared by all Identity Providers across the affiliation.

SAML_KEY_AFFILIATION_ID

Required

Yes

Default

None

Description

The URI for the affiliation. The ID is used to verify that a Service Provider and Identity Provider are members of the same affiliation-for example:

- When a Service Provider issues an authentication request to an Identity Provider, the request includes the affiliation ID. The Identity Provider verifies that the Service Provider belongs to the specified affiliation.

- When the Identity Provider generates an assertion and sends it back to the Service Provider, the assertion includes the affiliation ID. The Service Provider verifies that the Identity Provider belongs to the specified affiliation.
- During single logout, the logout requests also contain the affiliation ID. Upon receiving a logout request, the Service Provider and the Identity Provider each verify that the other belongs to the specified affiliation.

The affiliation ID is specified in the SPNameQualifier attribute of the requests and assertions.

SAML_KEY_IDP_SOURCEID

Required

No

Default

A hex-encoded SHA-1 hash of the SAML_KEY_IDPID value

Description

A hex-encoded 20-byte sequence identifier for the artifact issuer. This value uniquely identifies the artifact issuer in the assertion artifact.

The authentication scheme uses the source ID as a key to look up Identity Provider metadata.

The string length must be exactly 40 characters. Only a lower case hex string will be stored.

SAML_KEY_IDPID

Required

Yes

Default

None

Description

The provider ID of the Identity Provider for this authentication scheme. This ID:

- Uniquely identifies the assertion issuer.
- Serves as a key for looking up properties of the Identity Provider.

SAML_KEY_SPID

Required

Yes

Default

None

Description

The unique provider ID of this Service Provider.

SAML_MAJOR_VERSION

Required

No

Default

2

Description

The major version of the SAML protocol that is supported. If a value is supplied, it must be 2.

SAML_MINOR_VERSION

Required

No

Default

0

Description

The minor version of the SAML protocol that is supported. If a value is supplied, it must be 0.

SAML_NAME

Required

Yes

Default

None

Description

The name of the affiliation, authentication scheme, or Service Provider.

The name must be globally unique. With SAML 2.0 affiliations and Service Providers, the name must be lower case.

SAML_OID

Required

No, when the affiliation object is being created (SiteMinder supplies the object identifier during object creation); it is required when custom code references an existing object

Default

None

Description

The unique object identifier for the affiliation object.

The SAML Affiliation Properties dialog box has no corresponding field for this property.

SAML_SKEWTIME

Required

No

Default

30

Description

The difference, in seconds, between the system clock time of the Identity Provider and the system clock time of the Service Provider, as follows:

- With Service Providers, the number of seconds to be subtracted from the current time if its system clock is not synchronized with the Policy Server acting as an Identity Provider.
- With Identity Providers, the number of seconds to be subtracted from the current time if its system clock is not synchronized with the Policy Server acting as a Service Provider.

Skew time is used to calculate the validity duration of assertions and single logout requests. The value provided must be a String representing a positive integer.

SAML_SLO_REDIRECT_BINDING

Required

No

Default

0

Description

Specifies whether HTTP redirect binding is supported for single logout.

Valid values: 0 (false) and 1 (true).

See also SAML_DSIG_VERINFO_ISSUER_DN and SAML_DSIG_VERINFO_SERIAL_NUMBER.

SAML_SLO_SERVICE_CONFIRM_URL

Required

No

Default

None

Description

The URL where a user is redirected after single logout is completed.

SAML_SLO_SERVICE_RESPONSE_URL

Required

No

Default

None

Description

The response location for the single logout service. This property allows SLO response messages to be sent to a different location from where request messages are sent.

SAML_SLO_SERVICE_URL

Required

Yes, if SAML_SLO_REDIRECT_BINDING is 1

Default

None

Description

With HTTP-Redirect bindings, the Identity Provider URL where single logout requsts are sent.

SAML_SLO_SERVICE_VALIDITY_DURATION

Required

No

Default

60 (applies if a value is not provided and SAML_SLO_REDIRECT_BINDING is 1)

Description

The number of seconds for which a single logout request is valid.

The value provided must be a String representing a positive integer.

See also SAML_SKEWTIME.

SAML_SP_ARTIFACT_ENCODING

Required

No

Default

FORM (applies if a value is not provided and SAML_ENABLE_SSO_ARTIFACT_BINDING is 1)

Description

Specifies the encoding to use for the artifact binding. Valid values:

- FORM. The artifact is form-encoded in a hidden control named SAMLart.
- URL. The artifact is URL-encoded in a URL parameter named SAMLart.

FORM and URL encoding is accomplished according to SAML 2.0 specifications.

SAML_SP_ASSERTION_CONSUMER_DEFAULT_URL

Required

Yes

Default

None

Description

The Service Provider URL where generated assertions are sent, for example:

http://mysite.netegrity.com/affwebservices/public/saml2assertionconsumer

SAML_SP_AUTHENTICATION_LEVEL

Required

No

Default

5

Description

This property specifies the minimum protection level required for the authentication scheme that authenticates the principal associated with the current assertion.

SAML_SP_ATTRSVC_AD_SEARCH_SPEC

Required

No

Default

None

Description

Search specification for an AD directory.

SAML_SP_ATTRSVC_CUSTOM_SEARCH_SPEC

Required

No

Default

None

Description

Search specification for a custom directory.

SAML_SP_ATTRSVC_ENABLE

Required

No

Default

0

Description

Indicates whether the Attribute Authority is enabled.

Valid values: 0 (false) and 1 (true).

SAML_SP_ATTRSVC_LDAP_SEARCH_SPEC

Required

No

Default

None

Description

Search specification for an LDAP directory.

SAML_SP_ATTRSVC_ODBC_SEARCH_SPEC

Required

No

Default

None

Description

Search specification for an ODBC directory.

SAML_SP_ATTRSVC_REQUIRE_SIGNED_QUERY

Required

No

Default

None

Description

Specifies whether the attribute query must be signed.

SAML_SP_ATTRSVC_SIGN_ASSERTION

Required

No

Default

0

Description

Indicates whether the SAML assertion should be signed.

Valid values: 0 (false) and 1 (true).

SAML_SP_ATTRSVC_SIGN_RESPONSE

Required

No

Default

0

Description

Indicates whether the SAML response should be signed.

Valid values: 0 (false) and 1 (true).

SAML_SP_ATTRSVC_VALIDITY_DURATION

Required

No

Default

60

Description

The number of seconds for which a generated assertion is valid.

SAML_SP_ATTRSVC_WINNT_SEARCH_SPEC

Required

No

Default

None

Description

Search specification for a WinNT directory.

SAML_SP_AUTHENTICATION_URL

Required

Yes

Default

None

Description

The protected URL for authenticating users of this Service Provider.

SAML_SP_AUTHN_CONTEXT_CLASS_REF

Required

No

Default

urn:oasis:names:tc:SAML:2.0:ac:classes:Password

Description

The class of information that a Service Provider may require to assess its confidence in an assertion. The class is specified in the assertion's AuthnContextClassRef element.

For example, the default authentication context class is Password. This class applies when a principal authenticates through the presentation of a password over an unprotected HTTP session.

Other examples of authentication context class include InternetProtocol (authentication through a provided IP address), X509 (authentication through an X.509 digital signature), and Telephony (authentication through the provision of a fixed-line telephone number transported via a telephony protocol).

The authentication context class is a URI with the following initial stem:

urn:oasis:names:tc:SAML:2.0:ac:classes:

The SAML 2.0 authentication context specification defines the URIs that can be provided as authentication context classes. The class must also be appropriate for the authentication level defined for the Service Provider.

SAML_SP_COMMON_DOMAIN

Required

Yes, if SAML_SP_ENABLE_IPD is 1

Default

None

Description

The common cookie domain for the Identity Provider Discovery profile. The domain must be a subset of the host specified in SAML_SP_IPD_SERVICE_URL.

SAML_SP_CUSTOM_TIME_OUT

Required

No

Default

None

Description

Specifies the value of the SessionNotOnOrAfter parameter set in the assertion. This property is only valid if SAML_SP_SESSION_NOTORAFTER_TYPE is set to Custom.

SAML_SP_DOMAIN

Required

No

Default

None

Description

The unique ID of the affiliate domain where the Service Provider is defined.

The SAML Service Provider Properties dialog box has no corresponding field for this property.

SAML_SP_ENABLE_IPD

Required

No

Default

0

Description

Specifies whether the Identity Provider Discovery profile is enabled.

Valid values: 0 (false) and 1 (true).

SAML_SP_ENCRYPT_ASSERTION

Required

No

Default

0

Description

Specifies whether to encrypt the generated assertion at the Service Provider site. By default, the assertion is not encrypted.

Valid values: 0 (false) and 1 (true).

SAML_SP_ENCRYPT_BLOCK_ALGO

Required

No

Default

tripledes

Description

The type of block encryption algorithm to use. Valid values:

- tripledes. Data Encryption Standard using three separate 56-bit keys.
- aes-128. Advanced Encryption Standard, key length is 128 bits.
- aes-256. Advanced Encryption Standard, key length is 256 bits.

SAML_SP_ENCRYPT_CERT_ISSUER_DN

Required

Yes, in either of the following circumstances:

If either of the following is 1:

- SAML_SP_ENCRYPT_ID
- SAML_SP_ENCRYPT_ASSERTION

If any assertion attribute statements require encryption. These attributes are defined on the Attributes tab of the SAML Service Provider Properties dialog box.

Default

None

Description

The Issuer DN portion of a public key certificate to be used for encryption. This property is used with SAML_SP_ENCRYPT_CERT_SERIAL_NUMBER to locate the Service Provider's certificate in the keystore if it is not provided inline.

SAML_SP_ENCRYPT_CERT_SERIAL_NUMBER

Required

Yes, in either of the following circumstances:

If either of the following is 1:

- SAML_SP_ENCRYPT_ID
- SAML_SP_ENCRYPT_ASSERTION

If any assertion attribute statements require encryption. These attributes are defined on the Attributes tab of the SAML Service Provider Properties dialog box.

Default

None

Description

The serial number portion of a public key certificate to be used for encryption. This property is used with SAML_SP_ENCRYPT_CERT_ISSUER_DN to locate the Service Provider's certificate in the keystore if it is not provided inline.

SAML_SP_ENCRYPT_ID

Required

No

Default

0

Description

Specifies whether the Name ID in the generated assertion should be encrypted at the Service Provider site. By default, the Name ID is not encrypted.

Valid values: 0 (false) and 1 (true).

SAML_SP_ENCRYPT_KEY_ALGO

Required

No

Default

rsa-v15

Description

The type of encryption key algorithm to use. Valid values:

- rsa-v15. RSA encryption, version 1.5.
- rsa-oaep. Optimal Asymmetric Encryption Padding encoding and RSA encryption.

SAML_SP_ENDTIME

Required

No

Default

None

Description

The time by which an assertion must be generated.

Use the Perl time() method to help assign a time to this property. The time value is stored as a string. For example:

```
$SAML_SP_ENDTIME=SAML_SP_ENDTIME;
$time=time() + 20;
$serviceProvider->Property($SAML_SP_ENDTIME,"$time");
```

This property is used with SAML_SP_STARTTIME to define a time restriction for the generation of assertions.

Set SAML_SP_ENDTIME to 0 to end the time restriction immediately.

SAML_SP_IDP_SOURCEID

Required

No

Default

A hex-encoded SHA-1 hash of the SAML_SP_IDPID value

Description

A hex-encoded 20-byte sequence identifier for the artifact issuer. This value uniquely identifies the artifact issuer in the assertion artifact.

The string length must be exactly 40 characters. Only a lower case hex string will be stored.

SAML_SP_IDPID

Required

Yes

Default

None

Description

The provider ID of the Identity Provider that generates the assertions.

SAML_SP_IGNORE_REQ_AUTHNCONTEXT

Required

No

Default

0

Description

Specifies that the Identity Provider ignore "RequestedAuthnContext" in an incoming AuthnRequest message (value of 1), or not (Value of 0).

SAML_SP_IPD_SERVICE_URL

Required

Yes, if SAML_SP_ENABLE_IPD is 1

Default

None

Description

The host URL for the Identity Provider Discovery profile.

SAML_SP_NAMEID_ATTRNAME

Required

Yes, if SAML_SP_NAMEID_TYPE is set to 1 (User Attribute) or 2 (DN Attribute)

Default

None

Description

One of the following values:

- When SAML_SP_NAMEID_TYPE is set to 1, this property specifies the name of the user attribute that contains the name identifier.
- When SAML_SP_NAMEID_TYPE is set to 2, this property specifies the attribute associated with a group or organizational unit DN.

SAML_SP_NAMEID_DNSPEC

Required

Yes, if SAML_SP_NAMEID_TYPE is set to 2 (DN Attribute)

Default

None

Description

A group or organizational unit DN used to obtain the associated Name ID attribute.

SAML_SP_NAMEID_FORMAT

Required

No

Default

Unspecified

Description

The full URI for one of the following nameid-format values:

- Unspecified
- Email Address
- X509 Subject Name
- Windows Domain Qualified Name
- Kerberos Principal Name
- Entity Identifier
- Persistent Identifier
- Transient Identifier

For example, the full URI for the default format Unspecified is:

urn:oasis:names:tc:SAML:2.0:nameid-format:unspecified

For descriptions of these formats, see the following SAML 2.0 specification:

Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0

Note: If a SAML affiliation is specified in SAML_AFFILIATION, this and other SAML_SP_NAMEID... properties are not used. SiteMinder uses the NAMEID information in the specified affiliation.

SAML_SP_NAMEID_STATIC

Required

Yes, if SAML_SP_NAMEID_TYPE is set to 0 (Static)

Default

None

Description

The static text to be used for all name identifiers.

SAML_SP_NAMEID_TYPE

Required

No

Default

1

Description

The type of name identifier. Valid values:

- 0. Static text.
- 1. User attribute.
- 2. DN attribute.

SAML_SP_ONE_TIME_USE

Required

No

Default

False

Description

Specifies whether the Assertion includes an element that indicates that the Assertion should be used only one time.

SAML_SP_PASSWORD

Required

Yes, if SAML_ENABLE_SSO_ARTIFACT_BINDING is 1

Default

None

Description

The password to use for Service Provider access through the back channel.

SAML_SP_PERSISTENT_COOKIE

Required

No

Default

0

Description

Specifies whether an Identity Provider Discovery profile cookie should be persistent.

Applies only if SAML_SP_ENABLE_IPD is 1.

Valid values: 0 (false) and 1 (true).

SAML_SP_PLUGIN_CLASS

Required

No

Default

None

Description

The fully qualified Java class name of the assertion generator plug-in.

An assertion generator plugin allows the content of an assertion to be customized. For more information, see the SiteMinder Java API Documentation.

SAML_SP_PLUGIN_PARAMS

Required

No

Default

None

Description

Any parameters to pass into the assertion generator plug-in specified in SAML_SP_PLUGIN_CLASS.

SAML_SP_REQUIRE_SIGNED_AUTHNREQUESTS

Required

No

Default

0

Description

Specifies whether authentication requests must be signed.

Valid values: 0 (false) and 1 (true).

SAML_SP_REUSE_SESSION_INDEX

Required

No

Default

0

Description

Indicates whether CA SiteMinder sends the same session index in the assertion for the same partner in a single browser session. If a user federates multiple times with the same partner using the same browser window, setting this property tells the IdP to send the same session index in each assertion. The default value (0) for the property instructs SiteMinder to generate a new session index every time single sign-on occurs.

Valid values:

0

Do not reuse the same session index.

1

Reuse the same session index.

SAML_SP_SESSION_NOTORAFTER_TYPE

Required

No

Default

Use Assertion Validity

Description

This property determines the value set for the SessionNotOnOrAfter parameter in the assertion. A third-party SP can use the value of the SessionNotOnOrAfter to set its own session timeout.

If CA SiteMinder is acting as an SP, it ignores the SessionNotOnOrAfter value. Instead, a CA SiteMinder SP sets session timeouts based on the realm timeout that corresponds to the configured SAML authentication scheme that protects the target resource.

Use Assertion Validity

Calculates the SessionNotOnOrAfter value based on the assertion validity duration.

Omit

Instructs the IdP not to include the SessionNotOnOrAfter parameter in the assertion.

IDP Session

Calculates the SessionNotOnOrAfter value based on the IdP session timeout. The timeout is configured in the IdP realm for the authentication URL. Using this option can synchronize the IdP and SP session timeout values.

Custom

Lets you specify a custom value for the SessionNotOnOrAfter parameter. If you select this option, enter a time in the SAML_SP_CUSTOM_TIME_OUT property.

SAML_SP_STARTTIME

Required

No

Default

None

Description

The time when a time restriction for generating an assertion becomes effective.

Use the Perl time() method to help assign a time to this property. The time value is stored as a string. For example:

```
$SAML_SP_STARTTIME=SAML_SP_STARTTIME;
$time=time() + 10;
$serviceProvider->Property($SAML_SP_STARTTIME,"$time");
```

This property is used with SAML_SP_ENDTIME to define a time restriction for the generation of assertions.

Set SAML_SP_STARTTIME to 0 to start the time restriction immediately.

SAML_SP_VALIDITY_DURATION

Required

No

Default

60

Description

The number of seconds for which a generated assertion is valid.

The value provided must be a Strng representing a positive integer.

See also SAML_SKEWTIME.

SAML_SSOECPPROFILE

Required

No

Default

0

Description

Specifies whether the Identity Provider or Service Provider supports SAML 2.0 Enhanced Client and Proxy profile requests.

Valid values: 0 (false) and 1 (true).

SAML2_CUSTOM_ENABLE_INVALID_REQUEST_URL

Required

No

Default

None

Description

Specifies whether the custom error redirect process is enabled for an invalid request.

SAML2_CUSTOM_ENABLE_SERVER_ERROR_URL

Required

No

Default

None

Description

Specifies whether the custom error redirect process is enabled for a server error.

SAML2_CUSTOM_ENABLE_INVALID_REQUEST_URL

Required

No

Default

None

Description

Specifies whether the custom error redirect process is enabled for an invalid request.

SAML2_CUSTOM_INVALID_REQUEST_REDIRECT_MODE

Required

No

Default

None

Description

Specifies the redirect mode for an invalid request. Valid values:

- 0. 302 No Data — HTTP 302 redirection. The URL for the target resource and the reason for the authentication failure are appended to the redirection URL. The SAML 2.0 Response message passed to the authentication scheme is not included.
- 1. Http Post. — HTTP POST redirection. The SAML 2.0 Response message passed to the authentication scheme and the Identity Provider's ID are generated by an HTTP form.

SAML2_CUSTOM_INVALID_REQUEST_REDIRECT_URL

Required

No

Default

None

Description

Specifies the redirect URL for an invalid request.

SAML2_CUSTOM_SERVER_ERROR_REDIRECT_MODE

Required

No

Default

None

Description

Specifies the redirect mode for an internal server error. Valid values:

- 0. 302 No Data — HTTP 302 redirection. The URL for the target resource and the reason for the authentication failure are appended to the redirection URL. The SAML 2.0 Response message passed to the authentication scheme is not included.
- 1. Http Post. — HTTP POST redirection. The SAML 2.0 Response message passed to the authentication scheme and the Identity Provider's ID are generated by an HTTP form.

SAML2_CUSTOM_SERVER_ERROR_REDIRECT_URL

Required

No

Default

None

Description

Specifies the redirect URL for an internal server error .

SAML2_CUSTOM_UNAUTHORIZED_ACCESS_REDIRECT_MODE

Required

No

Default

None

Description

Specifies the redirect mode for forbidden access. Valid values:

- 0. 302 No Data — HTTP 302 redirection. The URL for the target resource and the reason for the authentication failure are appended to the redirection URL. The SAML 2.0 Response message passed to the authentication scheme is not included.
- 1. Http Post. — HTTP POST redirection. The SAML 2.0 Response message passed to the authentication scheme and the Identity Provider's ID are generated by an HTTP form.

SAML2_CUSTOM_UNAUTHORIZED_ACCESS_REDIRECT_URL

Required

No

Default

None

Description

Specifies the redirect URL for a forbidden access error.

WSFED Properties

This section provides the name, type, and description for each WS-Federation meatadata property.

The following properties are for defining a Resource Partner or for defining an Account Partner or for both.

WSFED_AP_ADD_SEARCH_SPEC

Required

No

Type

String

Description

Search specification for an AD directory.

WSFED_AP_CUSTOM_SEARCH_SPEC

Required

No

Type

String

Description

Search specification for a custom directory.

WSFED_AP_FAILURE_REDIRECT_MODE

Required

No

Type

0/1

Description

- 0 - Http 302 redirect without passing federation messages (default).
- 1 - Http Form Post Redirect.

WSFED_AP_FAILURE_REDIRECT_URL

Required

No

Type

String

Description

Contains an optional redirect URL to be used when assertion processing has failed.

WSFED_APID

Required

Yes

Type

String

Description

The ID of the Account Partner.

WSFED_AP_INVALID_REDIRECT_MODE

Required

No

Type

0/1

Description

- 0 - Http 302 redirect without passing federation messages (default).
- 1 - Http Form Post Redirect.

WSFED_AP_INVALID_REDIRECT_URL

Required

No

Type

String

Description

Contains an optional redirect URL to be used when the assertion is invalid.

WSFED_AP_LDAP_SEARCH_SPEC

Required

No

Type

String

Description

Search specification for the LDAP directory.

WSFED_AP_ODBC_SEARCH_SPEC

Required

No

Type

String

Description

Search specification for an ODBC directory.

WSFED_AP_PLUGIN_CLASS

Required

No

Type

String

Description

Name of the Java class that implements customization of assertion consumption.

WSFED_AP_PLUGIN_PARAMS

Required

No

Type

String

Description

Parameters of the Java class that implements customization of assertion consumption. All parameters are concatenated into one line.

WSFED_AP_SIGNOUT_URL

Required

No

Type

String

Description

Signout URL of the Account Partner. This property is required if WSFED_AP_SLO_ENABLED is true.

WSFED_AP_SLO_ENABLED

Required

No

Type

Boolean

Description

Indicates whether Signout is enabled for the Account Partner. If not supplied during Account Partner creation, this defaults to not enabled.

WSFED_AP_SSO_DEFAULT_SERVICE

Required

No

Type

String

Description

The default location of the Single Sign-on service.

WSFED_AP_SSO_REDIRECT_MODE

Required

No

Type

Int

Description

Redirect mode for assertion attributes. Valid values:

- 0. 302 No Data - No response attributes are passed (default).
- 1. 302 Cookie Data - Response attributes are set as HTTP cookie data. Attribute cookies issued by the authentication scheme are unencrypted.
- 2. Server Redirect - Response attributes are passed as a HashMap object.

WSFED_AP_SSO_TARGET

Required

No

Type

String

Description

Target resource at the destination site.

WSFED_AP_USER_NOT_FOUND_REDIRECT_MODE

Required

No

Type

0/1

Description

- 0 - Http 302 redirect without passing federation messages (default).
- 1 - Http Form Post Redirect.

WSFED_AP_USER_NOT_FOUND_REDIRECT_URL

Required

No

Type

String

Description

Contains an optional redirect to be used in either of the following cases:

- When the authentication scheme cannot obtain a Login ID from the federation Message, given the configured query string,
- When the authentication scheme cannot find a user in the specified user directory, given the configured user store search string.

WSFED_AP_WINNT_SEARCH_SPEC

Required

No

Type

String

Description

Search specification for a WinNT directory.

WSFED_AP_XPATH

Required

No

Type

String

Description

XPath query for disambiguating the principal.

WSFED_DESCRIPTION

Required

No

Type

String

Description

A brief description of the provider.

WSFED_DISABLE_SIGNATURE_PROCESSING

Required

No

Type

Boolean

Description

Specifies whether signature processing is disabled. This setting is useful during the initial setup of an Account Partner. When an Account Partner is up and running, this setting must be false to avoid security implications The default value is zero.

WSFED_DSIG_VERINFO_ALIAS

Required

No

Type

String

Description

Locates the certificate of the provider in the key store if it is not provided in-line.

WSFED_ENABLED

Required

No

Type

Bool

Description

Indicates whether the Resource Partner is enabled. If not provided, defaults to true. This property does not get stored physically to the property collections, but is used to enable underlying policy.

WSFED_ENFORCE_SINGLE_USE_POLICY

Required

No

Type

Boolean

Description

If set to a value of 1, the single-use policy for WS-Federation assertions will be enforced. If set to a value of 0, the single-use policy for assertions will not be enforced. The default is 1.

WSFED_KEY_APID

Required

Yes

Type

String

Description

Identifier for the Account Partner. This must be a URI less the 1024 characters long. In addition, this is the key with which properties associated with an Account Partner can be looked up.

WSFED_KEY_RPID

Required

Yes

Type

String

Description

The ID for the for the Resource Partner. This must be a URI less the 1024 characters long. In addition, this is the key with which the properties associated with a Resource Partner can be looked up.

WSFED_MAJOR_VERSION

Required

No

Type

Int

Description

Version of the WS-Federation protocol supported by this provider. The value of this property has to be 1.

WSFED_MINOR_VERSION

Required

No

Type

Int

Description

Version of WS-Federation protocol supported by this provider. The value of this property must be set to 0.

WSFED_NAME

Required

Yes

Type

String

Description

The name of the provider.

WSFED_RPID

Required

Yes

Type

String

Description

Identifier of the Resource Partner.

WSFED_RP_ASSERTION_CONSUMER_DEFAULT_URL

Required

Yes

Type

String

Description

The the URL of the default Assertion Consumer.

WSFED_RP_AUTHENTICATION_LEVEL

Required

No

Type

Int

Description

The principal must have authenticated in a realm by an authentication scheme of at least this level or greater. If not provided when the Resource Partner is created, the default is 5.

WSFED_RP_AUTHENTICATION_METHOD

Required

No

Type

String

Description

The authentication method to use in the assertion. This will typically be one of the authentication method values from the WS-Federation specification.

WSFED_RP_AUTHENTICATION_URL

Required

Yes

Type

String

Description

The protected URL used to authenticate Resource Partner users.

WSFED_RP_DOMAIN

Required

Yes

Type

OID

Description

The Resource Partner domain where this provider is defined.

WSFED_RP_ENDTIME

Required

No

Default

None

Description

The time by which an assertion must be generated.

Use the Perl time() method to help assign a time to this property. The time value is stored as a string. For example:

```
$WSFED_RP_ENDTIME=WSFED_RP_ENDTIME;
$time=time() + 20;
$ResourcePartner->Property($WSFED_RP_ENDTIME,"$time");
```

This property is used with WSFED_RP_STARTTIME to define a time restriction for the generation of assertions.

Set WSFED_RP_ENDTIME to 0 to end the time restriction immediately.

WSFED_RP_NAMEID_ALLOWED_NESTED

Required

No

Type

Boolean

Description

Indicates whether nested groups are allowed when selecting a DN attribute for the name identifer. The default is zero.

WSFED_RP_NAMEID_ATTR_NAME

Required

No

Type

String

Description

The attribute name (user or DN) that holds the name identifier when NameIdType is assigned to 1 or NameIdType is assigned to 2. If NameIdType is set to 1 or 2, then this property must had a value.

WSFED_RP_NAMEID_DN_SPEC

Required

No

Type

String

Description

The DN specification used when the NameIdType is assigned to 2. If NameIdType is assigned to 2, this property must have a value.

WSFED_RP_NAMEID_FORMAT

Required

No

Type

String

Description

The URI for a WS-Federation name identifier.

WSFED_RP_NAMEID_TYPE

Required

No

Type

Int

Description

One of the following types of name identifier:

- Static Text (0)
- User Attribute (1, default)
- DN Attribute (2)

WSFED_RP_NAMEID_STATIC

Required

No

Type

String

Description

The static text to be used as the name identifier when the NameIdType is assigned to 0. An error is returned if there is no value specified for this property and NameIdType is assigned to 0.

WSFED_RP_PLUGIN_CLASS

Required

No

Type

String

Description

The fully-qualified Java class name for the Assertion Generator plug-in.

WSFED_RP_PLUGIN_PARAMS

Required

No

Type

String

Description

The parameters passed to the Assertion Generator plug-in.

WSFED_RP_SIGNOUT_CLEANUP_URL

Required

No

Type

String

Description

Signout cleanup URL of the Resource Partner. This property is required if Signout is enabled.

WSFED_RP_SIGNOUT_CONFIRM_URL

Required

No

Type

String

Description

The URL where the user is redirected when Sign-out is complete and if the request does not have a reply query parameter. Even though this property is part of the Resource Partner object, it is the URL that the user is redirected to when Signout at the Account Partner is complete. If there are multiple Resource Partners available, then the Signout Confirm URL of the last Resource Partner is used. The default is disabled.

WSFED_RP_SLO_ENABLED

Required

No

Type

Boolean

Description

Indicates whether Signout is enabled for the Resource Partner.

WSFED_RP_STARTTIME

Required

No

Default

None

Description

The time when a time restriction for generating an assertion becomes effective.

Use the Perl time() method to help assign a time to this property. The time value is stored as a string. For example:

```
$WSFED_RP__STARTTIME=WSFED_RP_STARTTIME;
$time=time() + 10;
$ResourcePartner->Property($WSFED_RP_STARTTIME,"$time");
```

This property is used with WSFED_RP_ENDTIME to define a time restriction for the generation of assertions.

Set WSFED_RP_STARTTIME to 0 to start the time restriction immediately.

WSFED_RP_VALIDITY_DURATION

Required

No

Type

Integer

Description

The number of seconds for which a generated assertion is valid. If not provided when the Resource Partner is created, the default is 60 seconds.

WSFED_SAML_MAJOR_VERSION

Required

No

Type

Integer

Description

The version of the SAML protocol supported by this provider. The value is 1.

WSFED_SAML_MINOR_VERSION

Required

No

Type

Integer

Description

The version of the SAML protocol supported by this provider. The value is 1.

WSFED_SKEW_TIME

Required

No

Type

String

Description

The skew time between the consumer and the producer side in seconds. This value is used to calculate validity duration of assertions and of Signout requests. The default value is 30.

Index