CA SiteMinder®

Programming Guide for C

12.51

CA Technologies Product References

This document references the following CA Technologies products:

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

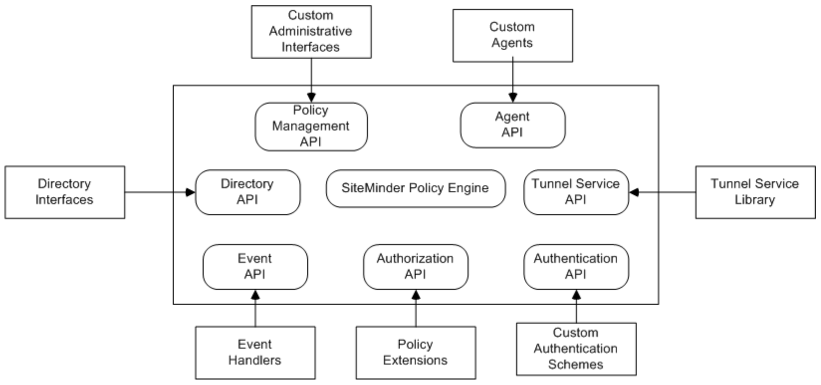
Chapter 1: API Overview

This section contains the following topics:

SiteMinder SDK Overview

The CA SiteMinder Software Development Kit (SDK) includes a set of documented application programming interfaces (APIs) that let you integrate and extend the capabilities of SiteMinder within your specific environment.

The following graphic shows how SiteMinder implements the APIs:



The SiteMinder SDK includes the following APIs:

- Policy Management API—use to develop a custom Administrative UI application, or to customize selected components of policy objects such as rules, policies, and responses within the application. Users of this API will be able to perform most of the data manipulations that users of the native Administrative UI can perform.
- Agent API—use to create custom agent applications for leveraging the authentication and authorization capabilities of the Policy Server. Using this API, you can also construct a secure communication tunnel to transmit application-specific data.
- Authentication API—use to develop custom authentication schemes and integrate the custom schemes. Policy Server plug-in.

- Authorization API—use to develop modules for performing custom authorization functions. Modules that you develop using this API can be configured in the Administrative UI with active rules, active policies, and active responses. Policy Server plug-in.
- Event API—use to build custom event handlers that handle SiteMinder events. Policy Server plug-in.
- Tunnel Service API—use to develop plug-in tunnel services that communicate with agents to transfer data securely. Policy Server plug-in.
- DMS Workflow API—use to create pre- and post-process workflow for DMS events. Policy Server plug-in.
- Directory API—use to mange user directories that use a custom namespace. Policy Server plug-in.

SDK Installation

The SDK is installed to the following location:

- UNIX platforms: <install_path>/sdk
- Windows platforms: <install_path>\sdk

<install_path> refers to the installation path where you installed the SDK software.

Note: For information about the installed directory structure, see the *SDK Overview*. For SDK installation instructions, see the *SDK Release Notes.*

Custom Applications and Policy Server Extensions

Note the following points when you build custom applications and Policy Server extensions:

- No SiteMinder processes need to be running on the machine where you build custom applications and Policy Server extensions.
- In most cases, the SDK does not need to be installed on the same machine as the Policy Server when you build those applications and extensions.
- The Policy Server is required for running the applications and extensions that you build with the SiteMinder SDK.

- Some of the application and extension runtime files need to be local to the Policy Server, while others can be remote.
- An application built with the Policy Management API must run as the same user who installed the Policy Server (for example, smuser on UNIX platforms).
- The C API cannot make JNI calls. There is no synchronization in the Policy Server to support such calls. It is possible, however, to spawn off a separate process that invokes Java and communicates back to the main process by using sockets

Code Samples

The SiteMinder SDK includes tested samples of C code for SiteMinder client applications and extensions. The source files for these samples are located as follows:

- Windows platforms:

  <install_path>\sdk\samples\<api-name>
- UNIX platforms:

  <install_path>/sdk/samples/<api-name>

Support for Custom Code

CA supports the Software Development Kit (SDK) as part of the standard offerings. Code written by customers or partners, however, is not supported. You are responsible for the code you write. If you require assistance designing or implementing SDK-based code, contact your CA customer account team.

Chapter 2: Agent API Guidance For C

This section contains the following topics:

Agent API Overview

The Agent API works with the Policy Server to simplify application development and make applications more scalable. Developers creating applications that are built using the Agent API either directly or indirectly (through another agent) are shielded from the following implementation-specific details:

- User namespaces such as LDAP directories, SQL databases, or NT domains
- Authentication methods as simple as user name and password, or as complex as PKI systems
- Authorizations based on group membership, or individual profile data

Additional benefits provided by the Agent API include full session management support, automatic encryption key rollover, and real-time policy updates.

About the SiteMinder Agent

A SiteMinder Agent is a client of the Agent API. The agent enforces access control policies provided by the Policy Server. The Policy Server is a general-purpose policy engine with no information about resources. Agents establish resource semantics and act as gate keepers to protect resources from unauthorized users.

Different agent types protect different kinds of resources. Some agent types are pre-defined, standard agents that are shipped as part of the SiteMinder product—for example, the Web Agent, which provides HTTP access control for Web Servers. You can also use the Agent API to implement custom agents.

When you create an application using the Agent API, you can create a custom agent to authenticate and authorize users in a variety of ways, depending on the context. For example, you could create an agent for FTP transfers that implements the following functionality:

- Certificate-based authentication instead of basic name and password credentials
- Uploads and downloads based on an individual user's authorization level

When you build a custom agent with the SDK, you must run the custom agent against the version of the SiteMinder Policy Server that corresponds to the version of the SDK.

Agent Initialization

Before an agent can perform work on behalf of its users, it must initialize connections to one or more Policy Servers by calling Sm_AgentApi_Init(). Calling Sm_AgentApi_Init() lets you specify connection parameters such as failover mode and connection pool size. This call creates TCP connections and typically does not need to be done more than once per agent instance.

It is possible to initialize more than one API instance (for example, when working with Policy Servers that use separate policy stores).

Immediately after initialization, the agent should communicate its version information to the Policy Server by calling Sm_AgentApi_DoManagement() with the command SM_AGENTAPI_MANAGEMENT_SET_AGENT_INFO set in the structure Sm_AgentApi_ManagementContext_t. The actual information can be any string containing enough information about the agent, such as the build number and version number. The string is recorded in the Policy Server logs.

Agent Discovery

Agent discovery lets CA SiteMinder® administrators track instances of different types of agents, including agents that have been deployed over a number of years. An agent instance can be any type of agent, for example, Web agent, custom agent, or ERP agent. To come under the purview of agent discovery, the agent must be active and in communication with the Policy Server.

Only 5.x agents and later can be tracked. For agents created before r12.5, the combination of the IP address and trusted host are used to identify the agent. Any change in this combination for the same agent results in multiple entries for the same agent.

A unique GUID identifies each r12.5 agent instance, which is stored in a configuration file. The Sm_AgentApi_SetAgentInstanceInfo() function plays a pivotal role in the process of agent discovery. The Sm_AgentApi_SetAgentInstanceInfo() function determines whether the configuration has a specified GUID for the agent instance. If a GUID is found, the agent has already been discovered and can be tracked. Otherwise, the function creates a GUID for the agent instance and writes it to the configuration file for subsequent invocations. Multiple agent instances cannot share a configuration file.

The Agent API provides the following for registering agent info:

```
int SM_EXTERN Sm_AgentApi_SetAgentInstanceInfo (
    const void* pHandle,
    Sm_AgentApi_AgentDiscovery_t arrParams[], int nCount);
```

- pHandle is a handle generated by Sm_AgentApi_Init, used by all agent api calls.
- arrParams[] is an array of name/value pairs defined by the m_AgentApi_AgentDiscovery_t data type.
- nCount is the number of elements passed in arrParams.

The following macros in SmAgentAPi.h define the supported named elements that can be passed in arrParams. These are all optional, and an agent provides as much, or as little, information as it relevant.

```
#define SM_AGENT_INSTANCE_AGENTPRODUCTTYPE       "AGENTPRODUCTTYPE"
```

Example: WebAgent

```
#define SM_AGENT_INSTANCE_AGENTPRODUCTVERSION    "AGENTPRODUCTVERSION"
```

Example: 1.0

```
#define SM_AGENT_INSTANCE_AGENTPRODUCTSUBTYPE    "AGENTPRODUCTSUBTYPE"
```

Example: IIS

```
#define SM_AGENT_INSTANCE_AGENTPRODUCTOSTYPE     "AGENTPRODUCTOSTYPE"
```

Example: Linux

```
#define SM_AGENT_INSTANCE_AGENTIDFILE            "AGENTIDFILE"
```

Valid path to a text file with write permission, used to store a unique GUID. If a path to an empty file is passed, the agent API will generate a new GUID and write it to this file. In the future, this GUID can be read and used to uniquely identify the agent instance.

```
#define SM_AGENT_INSTANCE_ACONAME                "ACONAME"
```

A value ACO name used by the agent.

```
#define SM_AGENT_INSTANCE_HCONAME                "HCONAME"
```

A valid HCO name used by the agent

```
#define SM_AGENT_INSTANCE_FIPSMODE              "FIPSMODE"
```

The value store in SmHost.conf if host registration is used by the agent.

After Sm_AgentApi_SetAgentInstanceInfo is called successfully, the AgentAPI will make periodic calls to the Policy Server in the back ground like a heartbeat. Calling this API for the first time from an agent results in a new unique record created in the Agent Discovery table used by the policy server. If a GUID is available, this record is indexed by the GUID. It is best to provide an Agent ID file so that a valid GUID is used. Without this value, the agent instance cannot be tracked and maintained as a unique entity in the SiteMinder agent ecosystem.

The correct calling sequence is:

```
Sm_AgentApi_Init
Sm_AgentApi_SetAgentInstanceInfo
..
other calls to PS
..
Sm_AgentApi_Uninit
```

Here is a source example that assumes Sm_AgentApi_Init has been called previously to establish pAgentApiHandle:

```
    Sm_AgentApi_AgentDiscovery_t arrParams[8] = {
                                {SM_AGENT_INSTANCE_AGENTPRODUCTTYPE,
"CustomAgent"},
                                {SM_AGENT_INSTANCE_AGENTPRODUCTVERSION, "1.0"},
                                {SM_AGENT_INSTANCE_AGENTPRODUCTSUBTYPE, "IIS"},
                                {SM_AGENT_INSTANCE_AGENTPRODUCTOSTYPE,
"Windows"},
                                {SM_AGENT_INSTANCE_AGENTIDFILE, "c:\idfile.txt"},
                                {SM_AGENT_INSTANCE_ACONAME, "MyACO"},
                                {SM_AGENT_INSTANCE_HCONAME, "MyHCO"},
                                {SM_AGENT_INSTANCE_FIPSMODE, "COMPAT"}};
    int nCount = 8;

    // Call the Agent API to send the data
    int nResult = Sm_AgentApi_SetAgentInstanceInfo(pAgentApiHandle, arrParams,
nCount);

    switch (nResult)
    {
        case SM_AGENTAPI_NOCONNECTION:
        case SM_AGENTAPI_FAILURE:
        case SM_AGENTAPI_TIMEOUT:
```

```
        {
            // put error handling here
            break;
        }
        case SM_AGENTAPI_NO:
        {
            // The call was refused - probably not properly formatted. This return
indicates
            // that the call ran on the PS and returned an explicit NO response, as
opposed to
            // some other connection or protocol error that might generate the
conditions above.
            break;
        }
        case SM_AGENTAPI_YES:
        {
            // Success
            break;
        }
    } // switch (nResult)
```

(new related group 1)

Sm_AgentApi_AgentDiscovery_t (see page 46)
Sm_AgentApi_SetAgentInstanceInfo() (see page 94)

How to Access a Resource Using the Agent API

After the Agent API has been initialized, the agent can start accepting requests from its users, such as receiving GET requests for URLs.

The following steps describe what is required for an agent to grant access to a resource. The outcome of most steps can be cached to improve agent performance. The agent can choose to cache as little as possible or as much as possible.

To grant user access to a resource

1. Accept a user request to access a resource. This is the application-specific request. For example, the Web Agent can accept a user's GET request for a URL.

2. Determine if the requested resource is protected by calling Sm_AgentApi_IsProtected(). If the resource is protected, the Policy Server returns the required credentials that must be obtained from the user to validate the user's identity. If the resource is not protected, access to the requested resource should be allowed. The outcome of this step can be cached.

   Collect the required credentials from the user and authenticate the user by calling Sm_AgentApi_Login(). After successful authentication, the Policy Server creates a session and returns response attributes including the unique session ID and session specification. These response attributes are policy driven and may include user profile data, static or dynamic privileges, a number of pre-defined authentication state attributes, or any other data that was designated by a policy administrator. The agent can now perform session management by caching user session information and keeping track of session expiration.

3. Validate that the user has access to the requested resource by calling Sm_AgentApi_Authorize(). After successful authorization, the Policy Server returns response attributes, including resource-specific privileges. These response attributes are policy driven and can include user profile data, static or dynamic privileges or any other data that was designated by a policy administrator. The user's authorization information for the requested resource is available and can be cached to speed up future requests.

4. (Optional) Log the transactions for authentication and authorization by calling Sm_AgentApi_Audit() if the agent performs authorizations out of its cache.

5. Allow the user to access the resource when the user's identity is known, authorization has been verified and the required entitlements obtained.

6. (Optional) Issue a management request by calling Sm_AgentApi_DoManagement(). This is an optional step that is used to poll the Policy Server for update commands. In response to a command, agents can update encryption keys or flush caches.

7. Release all API instances by calling Sm_AgentApi_UnInit() for each API instance. This closes TCP connections to all Policy Servers.

The Agent API does not provide a facility for caching in a manner that enforces session validity. By choosing to cache user sessions or resource-specific privileges, the agent becomes obligated to perform its own session management during each user request, because caching on the agent removes the need to contact the Policy Server to perform session validation or resource authorizations or both.

Compile and Link a Custom Agent

To enable your custom agent to interact with SiteMinder, compile your agent application with SmAgentAPI.h and link to the platform-specific library listed following. On UNIX platforms you must add the library to the environment variable that specifies the library search path. The variable takes a colon-separated list of directories and are also listed following.

Platform	Library	Directory	Variable Name
Windows	SmAgentAPI.lib	<install_path>\sdk\lib\win32\	
Solaris	libsmagentapi.so	<install_path>/sdk/bin	LD_LIBRARY_PATH
HP-UX	libsmagentapi.so	<install_path>/sdk/bin	SHLIB_PATH
AIX	libsmagentapi.so	<install_path>/sdk/bin	LIBPATH
Linux	libsmagentapi.so	<install_path>/sdk/bin	LD_LIBRARY_PATH

Note: If you are building the agent API application using Microsoft Visual C++, make sure you link with ws2_32.lib.

Central Host Configuration

SiteMinder agents, including custom agents, connect to a Policy Server through the Agent API. SiteMinder recognizes two different types of agents, based on the way that connection parameters are provided:

- **v4.x agents.** Connection parameters required to connect to a Policy Server are agent-specific. They include the agent’s name, the name or IP address of the host machine where the agent resides, and the shared secret. These parameters are provided when you define an agent object on the Policy Server. Connection parameters defined for a v4.x agent apply to that agent only.

- **v5.x and later agents**. Connection parameters required to connect to the Policy Server can apply to multiple agents on the host machine where the agents reside. With v5.x and later agents, the host machine is called a trusted host.

  The information required to initialize the connection to a Policy Server is stored in an SmHost.conf file on the host machine. Additional information that determines how the Policy Server interacts with its agents is defined on the Policy Server in a host configuration object.

  Host configuration object parameters (such as failover and clustering instructions) apply to all agents associated with the host configuration object. They are centrally managed in the Administrative UI.

Custom v5.x and later agents support central host configuration (which determines the way a Policy Server and its agents interact), but not central agent configuration (which determines the way an agent operates). You cannot define an agent configuration object for a custom agent in the Policy Server. Configuration parameters for the operation of a custom agent are defined in the WebAgent.conf file (or in the case of IIS 6.0 agents, in the LocalConfig.conf file).

Configuration Requirements

To configure a custom agent through a host configuration object on the Policy Server, you must complete the following steps:

1. Register the client machine where the agent resides as a trusted host.

   You register a trusted host with the smreghost tool. This tool is installed in directory <install_path>/sdk/bin.

   Registering a trusted host creates the following items:

   - A host configuration object on the Policy Server. The host configuration object can be modified in the Administrative UI at any time.
   - The file SmHost.conf on the client. This file includes the parameter hostconfigobject, which references the host configuration object on the Policy Server.

     The information in SmHost.conf is used to initialize the connection between the Policy Server and its agents, through the Agent API. Once the connection is initialized, the information in the host configuration object determines how the Policy Server and its agents interact.

2. Define an agent object on the Policy Server.

   An agent object establishes a unique identity for your custom agent by defining a name and other information that is specific to your custom agent.

   The name assigned to the custom agent must match the name that the custom agent passes programmatically to SiteMinder.

Code Requirements

For a custom agent to be configured through a central host configuration object on the Policy Server, the agent must do the following:

- Initialize agent configuration through Sm_AgentApi_GetConfig().

  This function lets you pass to SiteMinder the name and path of the SmHost.conf file that resides on the trusted host. This file references the host configuration object on the Policy Server.

  Alternatively, you can pass the name and path of the WebAgent.conf file if it contains a reference to SmHost.conf.

  After the trusted host connects to the Policy Server, the interaction between the Policy Server and the agents on the trusted host is determined by the host configuration object on the Policy Server.

  Note: When you call Sm_AgentApi_GetConfig(), SiteMinder populates the Sm_AgentApi_Init_t structure. With central host configuration, you do not populate the structure directly.

- Pass the agent's name through Sm_AgentApi_SetDefaultAgentId().

  The name that the custom agent passes to SiteMinder in this function must match the name of the agent object that establishes a unique identity for your custom agent on the Policy Server.

  Call Sm_AgentApi_SetDefaultAgentId() after calling Sm_AgentApi_Init() and before making any other Agent API calls. This allows you to specify your custom agent name to SiteMinder without sending the name with each transaction.

Upgrade an Agent

To upgrade an existing v4.x agent

1. If the host machine where the v4.x agent resides is not currently registered with the Policy Server as a trusted host, run smreghost to register it.

   Registration creates an SmHost.conf file on the trusted host and a host configuration object on the Policy Server.

2. Call Sm_AgentApi_GetConfig() to initialize the structure Sm_AgentApi_Init_t with the information in SmHost.conf.
3. Call Sm_AgentApi_Init() to connect to the Policy Server.
4. Call Sm_AgentApi_SetDefaultAgentId() to set the default name of the custom agent.

Agent Call Sequence

A custom agent usually calls Agent API functions in the in the following order:

```
Sm_AgentApi_GetConfig() // Required with central host configuration
Sm_AgentApi_Init()
Sm_AgentApi_SetAgentInstanceInfo() For agent discovery (12.5 agents and later)
Sm_AgentApi_SetDefaultAgentId() // Central host configuration only
Sm_AgentApi_DoManagement()
Sm_AgentApi_IsProtected()
Sm_AgentApi_Login()
Sm_AgentApi_Authorize()
// . . .

// Call other Agent API functions here, including
// periodic calling of Sm_AgentApi_DoManagement()

Sm_AgentApi_Logout()
Sm_AgentApi_Uninit()
```

Sample Custom Agent

Sample source code for the Agent API is provided in smagentexample.cpp. This file is installed in <install_path>/sdk/samples/smagentapi.

The Agent API sample can connect to the Policy Server as a v5.x or later agent (using central host configuration). When you run the sample, you are prompted to select the Agent Interface (v4.x-type or v5.x-type) before initializing the connection.

Agent API Services

The Agent API provides a rich set of services that let you develop sophisticated, secure, and robust agents. Building an agent involves using these services:

- Session Services
- Authorization Services
- Auditing Services
- Management Services (key encryption, cache updates)
- Tunnel Services

Session Services

A session is created after a successful user login. Once created, a user session persists until it is terminated. To maintain consistent user sessions in a multi-tiered application environment, a user session specification is maintained by the Web Agent (not the Policy Server). The session specification is also called the session ticket. The session specification represents a user session and is the key to SiteMinder Session Management. The environment in which the user session was created is responsible for persistent storage of the session specification. For example, the Web Agent (HTTP environment) stores the session specification in an HTTP cookie.

SiteMinder's universal ID is integrated with the sessioning mechanism. A universal ID identifies the user to an application in a SiteMinder environment using a unique identifier, such as a customer account number. The universal ID facilitates identification of users between old and new applications by delivering the user's identification automatically, regardless of the application. When configured on the Policy Server, a user's universal id is part of the session specification and is made available to agents for the duration of the entire session.

Agents create sessions using Sm_AgentApi_Login(). This function authenticates the user credentials and returns the session specification and unique session id in Sm_AgentApi_Session_t. The session specification is updated on subsequent Agent API calls that also return the updated expiration times. Agents can use this information to perform custom session management and keep track of session timeouts.

If your Web server's user tracking feature is enabled, SiteMinder issues an identity ticket in addition to the session specification. Identity tickets can be used for identity-based personalization when a user is accessing a resource protected by anonymous authentication schemes. Identity tickets never expire.

When an application's logic flow crosses application tiers, sessions can be delegated by passing the session specification between two agents. Each agent can choose to have the session specification validated.

The session specification is validated to make sure that a user session has neither expired nor been terminated or revoked. This can occur at any time during the session's lifetime. Agents call Sm_AgentApi_Login() to validate a session specification.

A session is terminated after a user logs out and the agent discards the session specification, when the session expires, or when the session is revoked. When a session is terminated, the user must log in again to establish a new session.

You should terminate a session if a user is disabled after a session has begun. To find a user's disabled state, call Sm_AgentApi_Login() to validate the session.

To terminate a session, the agent calls Sm_AgetnApi_Logout(). Note that any memory allocated for the session specification (Sm_AgentApi_Session_t) must be deallocated.

Application 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 Agent API provides the following functions for setting, retrieving, and deleting session variables:

- Sm_AgentApi_SetSessionVariables()
- Sm_AgentApi_GetSessionVariables()
- Sm_AgentApi_DelSessionVariables()

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

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 Administrative 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 and the agent discards the session specification, the session ends. In the case of a persistent session, SiteMinder removes all session information, including any session variables, from the session store.

Timeouts

Agents can enforce session timeouts themselves or rely on the Policy Server to validate each request. Typically, caching of user sessions or privileges by the agent requires some form of timeout enforcement on the agent side. In this case, the agent is responsible for keeping track of the last access time and knowing when the session is going to expire.

Agents that do not cache can leverage the Policy Server's enforcement of timeouts. The following Agent API methods return the updated timeout information after every call:

- Sm_AgentApi_Login()
- Sm_AgentApi_Authorize()
- Sm_AgentApi_Audit()

Authorization Services

Agents that perform access control functions use authorization services of the Agent API. These services enable clients to determine what access control is imposed on resources, verify users rights to access resources, and retrieve users privileges for specific resources.

Whether a resource is protected can be determined by calling the Sm_AgentApi_IsProtected() method. This method accepts a resource that is served by the requesting agent, and returns information about the credentials required for authentication.

After the user’s identity has been validated, agents call the Sm_AgentApi_Authorize() method to determine if the requesting user has access to the requested resource. Agents can perform fine-grained access control by testing the values of the response attributes returned by this method.

Transaction Tracking

A facility is provided for agents to keep track of all user activity during a session. Although much of the activity is logged by the Policy Server, there are times when it may be necessary to log authorizations done out of agent cache. Agents can call the Sm_AgentApi_Audit() method to log requests for resources.

By generating a unique transaction id, agents can correlate access control activity with application activity. The transaction id can be given to both the authorization and auditing methods so that the Policy Server would record the transaction-specific id associated with the application activity. This can be used for non-repudiation.

Management Services

A management protocol exists between agents and the Policy Server. This protocol helps agents manage its caches and encryption keys in a manner consistent with policies and administrative changes on the Policy Server.

Cache Commands

Agents issue the Sm_AgentApi_DoManagement() call with the SM_AGENTAPI_MANAGEMENT_GET_AGENT_COMMANDS command to request the latest agent commands. Typically, this is done every *N* seconds by a thread running in the background. The types of agent commands that can be received are cache commands and encryption commands.

Cache commands inform the agent of any changes to its caches that may need to be made as a result of administrative updates to the Policy Server. These agent commands are:

- SM_AGENTAPI_CACHE_FLUSH_ALL
- SM_AGENTAPI_CACHE_FLUSH_ALL_USERS
- SM_AGENTAPI_CACHE_FLUSH_THIS_USER
- SM_AGENTAPI_CACHE_FLUSH_ALL_REALMS
- SM_AGENTAPI_CACHE_FLUSH_THIS_REALM

Encryption commands

Encryption commands inform the agent of new encryption keys that are generated automatically by the Policy Server or administratively. Agents that need to save secure state can leverage this protocol to keep track of the latest encryption keys. These agent commands are:

- SM_AGENTAPI_AGENT_KEY_UPDATE_NEXT
- SM_AGENTAPI_AGENT_KEY_UPDATE_LAST
- SM_AGENTAPI_AGENT_KEY_UPDATE_CURRENT
- SM_AGENTAPI_AGENT_KEY_UPDATE_PERSISTENT

Tunnel Services

Tunnel services enable agents to establish secure communications with a callable service located on the Policy Server. This allows agents to perform custom actions over a secure, VPN-like channel without having to deal with issues such as encryption and key management.

Response Attributes

Response attributes enable the Policy Server to deliver information to agents. There are two types of attributes:

- Well-known attributes
- Policy-based attributes

Well-known attributes are always returned by the Policy Server after certain calls, such as Sm_AgentApi_Login(). These attributes represent static, fixed data such as the user DN and Universal ID.

Policy-based attributes are returned by Sm_AgentApi_Login() and Sm_AgentApi_Authorize(). These attributes are based on policies and are the vehicle for delivering static and dynamic data from the Policy Server to agents, so that the agents can distinguish between authentication and authorization attributes. The actual source of the data is defined on the Policy Server using the responses feature that can be configured to deliver data from a variety of sources. Data may include static information, information from a directory profile or a custom Policy Server plug-in. When the responses are properly configured, agents are capable of performing fine-grained access control as well as profile-driven personalization.

Based on a policy definition, response attributes can time out or be cached for the duration of the user session. The Policy Server delivers an attribute along with the TTL (Time-To-Live) value, calculated in seconds. If the agent is caching user sessions or authorizations or both, it is responsible for keeping the relevant attributes up to date. Agents issue the Sm_AgentApi_UpdateAttributes() call to update stale attributes.

Custom Agents and Single Sign-On

In a single sign-on environment, a user who successfully authenticates through a given agent does not have 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 Agent API contains two functions that allow custom agents to participate in a single sign-on environment with standard SiteMinder Web Agents:

- Sm_AgentApi_DecodeSSOToken(). The custom agent extracts the cookie's contents, called a token, from an existing SMSESSION cookie and passes the token to this function. The function decrypts the token and extracts the specified information. This function can also be used to update the last-access timestamp in the token.
- Sm_AgentApi_CreateSSOToken(). After the user successfully logs in through the custom agent, the custom agent passes information about the user to this function. The function creates an encrypted token from this user information and from session information returned from the login call. The custom agent writes the token to the SMSESSION cookie.

See the sample custom agent code for an example of setting up the parameters for the single sign-on functions and parsing the results. The sample custom agent code is located in the smagentapi directory of <install_path>\sdk\samples.

Standard Agent Support

Custom agents created with SiteMinder SDK v5.5 SPx and later 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 accompanying SDK release notes.

In addition, to enable a SiteMinder agent with the appropriate QMR upgrade to accept SMSESSION cookies created by a custom agent, the standard 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"
```

Set AcceptTPCookie as follows:

- With 4.xQMR4 agents and above, add AcceptTPCookie="yes" directly in the standard agent's Agent Configuration file.
- With 5.xQMR1 agents and later, add the entry to the standard agent's Agent Configuration Object if the AllowLocalConfig parameter for that object is set to no. If AllowLocalConfig is set to yes, you can set AcceptTPCookie in the standard agent's Agent configuration file.

Login Through a Custom Agent

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

1. User logs in through the custom agent.
2. Custom agent calls Sm_AgentApi_Login() to authenticate the user. The user is challenged for credentials.
3. Custom agent calls Sm_AgentApi_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 along with session information returned from the login call. SiteMinder also encrypts the information in the token.
4. Custom agent creates the SMSESSION cookie in the user's browser and writes the token to the cookie.
5. User requests a resource protected by a standard SiteMinder agent.
6. 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.

Login Through a Standard Agent

Here is the typical sequence of events in a single sign-on environment when the initial login is through the standard SiteMinder Web Agent:

1. User logs in through the standard agent.
2. Standard agent authenticates the user by challenging the user for credentials through the login call.
3. SiteMinder creates the SMSESSION cookie in the user's browser and inserts the encrypted token containing session information.
4. User requests a resource protected by a custom agent.
5. The custom agent obtains the SMSESSION cookie from the user's request and extracts the token.
6. The custom agent passes the token to the function Sm_AgentApi_DecodeSSOToken(). The function decodes the token and returns a subset of the token's attributes to the custom agent.
7. The custom agent obtains the session specification from the token and passes the session specification to Sm_AgentApi_Login(). The login call validates the user without challenging the user for credentials.
8. User requests a resource protected by a standard SiteMinder agent.
9. The standard agent performs a login operation, which validates the user based on the contents of the SMSESSION cookie. The user is not challenged for credentials.

Memory Deallocation

You must explicitly deallocate any memory that you allocate for your custom agent. To release the response attributes in the Sm_AgentApi_Attribute_t structure, call Sm_AgentApi_FreeAttributes().

More Information:

Sm_AgentApi_FreeAttributes() (see page 76)

Agent API Data Structures (C)

Before you use a structure, all unused fields should be set to zero. The best way to do this is to set the entire structure to zero.

Sm_AgentApi_AgentDiscovery_t

This structure defines information about agent instance parameters.

Syntax

This structure had the following format:

```
typedef struct Sm_AgentApi_AgentDiscovery_s
{
   char paramname[SM_AGENTAPI_SIZE_NAME];
   char paramvalue[SM_AGENTAPI_SIZE_NAME];
   } Sm_AgentApi_AgentDiscovery_t;
```

Parameters

This structure had the following parameters:

paramname

Use the following names to specify agent instance attributes:

- SM_AGENT_INSTANCE_AGENTPRODUCTTYPE — Type of agent instance
- SM_AGENT_INSTANCE_AGENTPRODUCTVERSION — Agent instance version
- SM_AGENT_INSTANCE_AGENTPRODUCTSUBTYPE— Type of web server
- SM_AGENT_INSTANCE_AGENTPRODUCTOSTYPE — OS type hosting Agent
- SM_AGENT_INSTANCE_AGENTIDFILE — Path of the file containing Agent GUID.
- SM_AGENT_INSTANCE_ACONAM E — Name of the agent configuration object
- SM_AGENT_INSTANCE_HCONAME — Name of host configuration object
- SM_AGENT_INSTANCE_FIPSMODE— FIPS Mode used for Agent-PS connection

paramvalue

The value of the specified attribute.

Remarks

All agent instance attributes are initialized to "unknown" value. The developer using the Agent API specifies the appropriate values by calling Sm_AgentApi_SetAgentInstanceInfo(). When this call is not made, the agent does not come under the purview of agent discovery.

Sm_AgentApi_Attribute_t

This structure defines information about a response attribute.

Syntax

This structure has the following format:

```
typedef struct Sm_AgentApi_Attribute_s
{
   long nAttributeId;
   long nAttributeTTL;
   long nAttributeFlags;
   char lpszAttributeOid[SM_AGENTAPI_SIZE_OID];
   long nAttributeLen;
   char* lpszAttributeValue;
} Sm_AgentApi_Attribute_t;
```

Parameters

This structure has the following parameters:

nAttributeId

ID of the response attribute.

nAttributeTTL

The time-to-live value (in seconds) for the response attribute. The attribute remains in cache for the duration of the TTL value.

nAttributeFlags

Response attribute flag. This flag is used in the following session store functions:

- Sm_AgentApi_DelSessionVariables()
- Sm_AgentApi_GetSessionVariables()
- Sm_AgentApi_SetSessionVariables()

See the ppRespAttributes parameter of these functions for more information.

lpszAttributeOid

The response attribute object identifier.

nAttributeLen

The length of the response attribute.

lpszAttributeValue

The null-terminated attribute value of a response attribute.

Remarks

The following well-known authentication attributes are returned by Sm_AgentApi_Login() and referenced in the nAttributeId field of the Sm_AgentApi_Attribute_t structure:

SM_AGENTAPI_ATTR_AUTH_DIR_OID

The Siteminder object id of the directory where the user was authenticated. This is the internal object id assigned to the SiteMinder user directory.

SM_AGENTAPI_ATTR_AUTH_DIR_NAME

The SiteMinder "name" specification of the directory where the user was authenticated. This is the directory name specified in the SiteMinder User Directory Dialog.

SM_AGENTAPI_ATTR_AUTH_DIR_SERVER

The SiteMinder "server" specification of the directory where the user was authenticated. This is the directory server specified in the SiteMinder User Directory Dialog.

SM_AGENTAPI_ATTR_AUTH_DIR_NAMESPACE

The SiteMinder "namespace" specification of the directory where the user was authenticated. This is the directory namespace (LDAP:, ODBC:, WinNT:, AD:) as specified in the SiteMinder User Directory Dialog.

SM_AGENTAPI_ATTR_USERMSG

The text presented to the user as a result of authentication. Some authentication schemes supply challenge text or a reason why an authentication has failed. A value for this attribute can be provided through the *lpszUserMsg* parameter of SmAuthenticate().

SM_AGENTAPI_ATTR_USERDN

The user’s distinguished name as recognized by SiteMinder.

This attribute is also used in single sign-on operations.

SM_AGENTAPI_ATTR_USERUNIVERSALID

The user’s universal id, as set in the user directory.

SM_AGENTAPI_ATTR_IDENTITYSPEC

The user’s identity ticket. SiteMinder returns this if the user tracking feature has been enabled.

The following well-known attributes are used in single sign-on operations and referenced in the nAttributeId field of the Sm_AgentApi_Attribute_t structure:

SM_AGENTAPI_ATTR_USERDN

The user's distinguished name.

SM_AGENTAPI_ATTR_SESSIONSPEC

The session specification returned from the login call.

SM_AGENTAPI_ATTR_SESSIONID

The session ID returned from the login call.

SM_AGENTAPI_ATTR_USERNAME

The user's name.

SM_AGENTAPI_ATTR_CLIENTIP

The IP address of the machine where the user initiated a request for a protected resource.

SM_AGENTAPI_ATTR_DEVICENAME

The name of the agent that is decoding the token.

SM_AGENTAPI_ATTR_InnnnnDLESESSIONTIMEOUT

Maximum idle time for a session.

SM_AGENTAPI_ATTR_STARTSESSIONTIME

The time the session started after a successful login.

SM_AGENTAPI_ATTR_LASTSESSIONTIME

The time that the Policy Server was last accessed within the session.

SM_AGENTAPI_ATTR_SSOZONE

Specifies the designation of the SSO zone name, which you provide when you call the Sm_AgentApi_CreateSSOToken method. If you do not specify a zone name, the default is "SM." You can read this value in the in the attribute list returned by the Sm_AgentApi_DecodeSSOToken method.

The following well-known management attributes are returned by Sm_AgentApi_DoManagement() and referenced in the nAttributeId field of the Sm_AgentApi_Attribute_t structure:

SM_AGENTAPI_AFFILIATE_KEY_UPDATE

Instructs the agent to update the name of the affiliate agent.

SM_AGENTAPI_AGENT_KEY_UPDATE_NEXT

Instructs the agent to update its "next" Agent key. The value contains 24 bytes of binary data.

SM_AGENTAPI_AGENT_KEY_UPDATE_LAST

Instructs the agent to update its "last" Agent key. The value contains 24 bytes of binary data.

SM_AGENTAPI_AGENT_KEY_UPDATE_CURRENT

Instructs the agent to update its "current" Agent key. The value contains 24 bytes of binary data.

SM_AGENTAPI_AGENT_KEY_UPDATE_ PERSISTENT

Instructs the agent to update its static (persistent) Agent key. The value contains 24 bytes of binary data.

SM_AGENTAPI_CACHE_FLUSH_ALL

Instructs the agent to flush all information in its caches.

SM_AGENTAPI_CACHE_FLUSH_ALL_USERS

Instructs the agent to flush all user information stored in its caches.

SM_AGENTAPI_CACHE_FLUSH_THIS_USER

Instructs the agent to flush all cache information pertaining to a given user. The value contains the following: <user dir oid> / <user dn>.

SM_AGENTAPI_CACHE_FLUSH_ALL_REALMS

Instructs the agent to flush all resource information stored in its caches.

SM_AGENTAPI_CACHE_FLUSH_THIS_REALM

Instructs the agent to flush all resource information pertaining to a given realm. The value is a realm OID.

Sm_AgentApi_Init_t

This structure defines agent initialization, including its server information, and also specifies the failover threshold.

Syntax

This structure had the following format:

```
typedef struct Sm_AgentApi_Init_s
{
   long nVersion;
   char lpszHostName[SM_AGENTAPI_SIZE_NAME];
   char lpszSharedSecret[SM_AGENTAPI_SIZE_NAME];
   long nFailover;
   long nNumServers;
   Sm_AgentApi_Server_t* pServers;
} Sm_AgentApi_Init_t;
```

Parameters

This structure had the following parameters:

nVersion

The version of the Agent API. Set to SM_AGENTAPI_VERSION.

lpszHostName

The agent name. This name must match the agent name provided to the Policy Server. The agent name is not case sensitive.

lpszSharedSecret

Each agent has a shared secret registered with the Policy Server. Enter that secret here. This is a case sensitive field.

nFailover

Setting this to 0 enables the agent to access the specified Policy Servers in a round-robin configuration. Setting this to 1 disables the round-robin configuration, in which case the agent will operate in the failover mode.

nNumServers

The number of Policy Servers defined in the next parameter.

pServers

An array of nNumServers structures of type Sm_AgentApi_Server_t. This structure defines each identically configured Policy Server with which the agent will communicate. The servers must share a common policy store. Depending on the specified Agent API version the array contains either clustered or non-clustered servers.

Remarks

This structure is populated by calling Sm_AgentApi_GetConfig().

More Information:

Sm_AgentApi_GetConfig() (see page 79)
Central Host Configuration (see page 34)

Sm_AgentApi_ManagementContext_t

This structure defines Information about the management command.

Syntax

This structure has the following format:

```
typedef struct Sm_AgentApi_ManagementContext_t
{
   long nCommand;
   char lpszData[SM_AGENTAPI_SIZE_NAME];
} Sm_AgentApi_ManagementContext_t;
```

Parameters

This structure has the following parameters:

nCommand

One of these management commands:

- SM_AGENTAPI_MANAGEMENT_GET_AGENT_COMMANDS
  Requests the latest agent commands. With this command, lpszData should be left blank.
- SM_AGENTAPI_MANAGEMENT_SET_AGENT_INFO
  Provides information about the agent to the Policy Server. The information is contained in lpszData.

lpszData

When SM_AGENTAPI_MANAGEMENT_SET_AGENT_INFO is set, contains the following null-terminated   string of information about the agent:

"Product=XXX,Platform=XXX,Version=XXX,Label=XXX"

Unknown clauses can be omitted.

This information is recorded by the Policy Server for logging and troubleshooting purposes.

Example:

"Product=WebAgent,Platform=NT/ISAPI,
    Version=5.0,Update=SP1,Label=C134"

The Agent API adds the crypto strength and agent time (in GMT) and time zone to this string.

If you want to log information about a custom agent, add the following lines of code to your custom agent definition:

```
ManagementContext.nCommand = SM_AGENTAPI_MANAGEMENT_SET_AGENT_INFO;
strncpy (ManagementContext.lpszData, version info goes here, 255);
```

Sm_AgentApi_Realm_t

This structure defines Information about the realm in which a resource is protected.

Syntax

This structure has the following format:

```
typedef struct Sm_AgentApi_Realm_s
{
   char lpszDomainOid[SM_AGENTAPI_SIZE_OID];
   char lpszRealmOid[SM_AGENTAPI_SIZE_OID];
   char lpszRealmName[SM_AGENTAPI_SIZE_NAME];
   long nRealmCredentials;
   char lpszFormLocation[SM_AGENTAPI_SIZE_URL];
} Sm_AgentApi_Realm_t;
```

Parameters

This structure has the following parameters:

lpszDomainOid

Unique identifier for the domain.

lpszRealmOid

Unique identifier for the realm.

lpszRealmName

Name of the realm in which the resource is protected.

nRealmCredentials

A bit mask of values indicating the required credentials. The values are defined in the Sm_Api_Credentials_t enumerated type (as defined in SmApi.h). The value 0 means that no credentials are required. The types are as follows:

- Sm_AuthApi_Cred_None
  No credential required.
- Sm_AuthApi_Cred_Basic
  Username and password required.
- Sm_AuthApi_Cred_Digest
  Digest required.
- Sm_AuthApi_Cred_X509Cert
  X.509 certificate required.
- Sm_AuthApi_Cred_X509CertUserDN
  X.509 certificate and user DN required.

- Sm_AuthApi_Cred_X509CertIssuerDN
  X.509 certificate issuer DN required.
- Sm_AuthApi_Cred_CertOrBasic
  Either an X.509 certificate or username/password is required.
- Sm_AuthApi_Cred_CertOrForm
  Either an X.509 certificate or a forms-based authentication scheme is required.
- Sm_AuthApi_Cred_NTChalResp
  Use the NT challenge/response protocol.
- Sm_AuthApi_Cred_SSLRequired
  SSL required.
- Sm_AuthApi_Cred_FormRequired
  A redirect to an HTML form is required.
- Sm_AuthApi_Cred_AllowSaveCreds
  Save credentials hint.
- Sm_AuthApi_Cred_PreserveSessionId
  Session id should be preserved if the current session is still valid.
- Sm_AuthApi_Cred_DoNotChallenge
  Do not challenge for credentials.

lpszFormLocation

URL of the form credential provider (http://...).

Sm_AgentApi_ResourceContext_t

This structure defines a resource for protection and authorization.

Syntax

This structure has the following format:

```
typedef struct Sm_AgentApi_ResourceContext_s
{
   char lpszAgent[SM_AGENTAPI_SIZE_NAME];
   char lpszServer[SM_AGENTAPI_SIZE_NAME];
   char lpszAction[SM_AGENTAPI_SIZE_NAME];
   char lpszResource[SM_AGENTAPI_SIZE_URL];
} Sm_AgentApi_ResourceContext_t;
```

Parameters

This structure has the following parameters:

lpszAgent

Name of the Affiliate Agent holding the resource. This flag is reserved for working with Affiliate Agents. Leave this field blank.

lpszServer

Optional field, used to specify the name of the Web server hosting the Web resource—for example, www.myorg.org.

lpszResoure

Name of the Web resource being requested—for example, /inventory/ .

lpszAction

Type of action to be performed on the resource—for example, GET.

Sm_AgentApi_Server_t

This structure defines the connection configuration for each server.

Syntax

This structure has the following format:

```
typedef struct Sm_AgentApi_Server_s
{
   char lpszIpAddr[SM_AGENTAPI_SIZE_NAME];
   long nConnMin;
   long nConnMax;
   long nConnStep;
   long nTimeout;
   long nPort[3];
   void* pHandle[3];
   long nClusterSeq;
} Sm_AgentApi_Server_t;
```

Parameters

This structure has the following parameters:

pszIpAddr

The IP address of the Policy Server. A pipe of TCP connections is formed for each of the three services within the Policy Server (Authorization, Authentication, and Accounting).

nConnMin

Describing the pipe of TCP connections, this is the initial number of connections.

nConnMax

Describing the pipe of TCP connections, this is the maximum number of allowed connections within the pipe.

nConnStep

As necessary, the number of connections in the pipe of TCP connections will be increased by this increment.

nTimeout

The number of seconds until it is determined that the agent can not reach the Policy Server. This parameter is configurable based on the overall throughput and latency conditions of the entire SiteMinder installation.

nPort

When the Policy Server is configured for a single Access Control TCP port, use the constant SM_AGENTAPI_POLICYSERVER to point to the combined port. The constants below are maintained for backward compatibility:

- SM_AGENTAPI_AZ_SERVER
- SM_AGENTAPI_AUTH_SERVER
- SM_AGENTAPI_ACCT_SERVER

pHandle

Reserved; set to null.

nClusterSeq

The cluster sequence number. Sequence numbers begin at 1. Omit this parameter for a non-cluster server.

Sm_AgentApi_Session_t

This structure defines information about the user's session.

Syntax

This structure has the following format:

```
typedef struct Sm_AgentApi_Session_s
{
   long nReason;
   long nIdleTimeout;
   long nMaxTimeout;
   long nCurrentServerTime;
   long nSessionStartTime;
   long nSessionLastTime;
   char lpszSessionId[SM_AGENTAPI_SIZE_OID];
   char lpszSessionSpec[SM_AGENTAPI_SIZE_SESSIONSPEC];
} Sm_AgentApi_Session_t;
```

Parameters

This structure has the following parameters:

nReason

Additional status code: explains the reason for failed authentication or authorization, or is passed in the event that results from a successful invocation of Sm_AgentApi_Logout(). Defined in SmApi.h.

nIdleTimeout

Maximum amount of time a session can be valid without the user accessing a resource before the agent should challenge the user to re-authenticate, defined in seconds.

nMaxTimeout

Maximum amount of time a user session can be active before the agent challenges the user to re-authenticate, defined in seconds.

nCurrentServerTime

Current time (in GMT) set on the Policy Server.

nSessionStartTime

Server time (in GMT) when the session started.

nSessionLastTime

Server time (in GMT) when the session was last seen by the Policy Server.

lpszSessionId

An opaque value returned to identify the session.

lpszSessionSpec

An opaque value returned to identify the session, which represents the session specification.

Sm_AgentApi_TunnelServiceRequest_t

This structure defines information about the remote service library.

Syntax

This structure has the following syntax:

```
typedef struct Sm_AgentApi_TunnelServiceRequest_s
{
   char lpszLibrary[SM_AGENTAPI_SIZE_NAME];
   char lpszFunction[SM_AGENTAPI_SIZE_NAME];
   char lpszParameter[SM_AGENTAPI_SIZE_USERINFO];
   long nLength;
   void* pData;
} Sm_AgentApi_TunnelServiceRequest_t;
```

Parameters

This structure has the following parameters:

lpszLibrary

The name of the service to be invoked by the Policy Server.

lpszFunction

The name of a method to call within the service.

lpszParameter

Arbitrary string parameter to be passed to the method.

nLength

The length of the data passed to the method. The maximum length can be determined by calling Sm_AgentApi_GetMaxTunnelBufSize().

pData

A pointer to the data.

Sm_AgentApi_UserCredentials_t

This structure is used for passing credentials to the server.

Note: The agent supplies only the relevant information as requested by the Policy Server.

Syntax

This structure has the following format:

```
typedef struct Sm_AgentApi_UserCredentials_s
{
   long nChallengeReason;
   char lpszUsername[SM_AGENTAPI_SIZE_USERINFO];
   char lpszPassword[SM_AGENTAPI_SIZE_USERINFO];
   char lszCertUserDN[SM_AGENTAPI_SIZE_USERINFO];
   char lpszCertIssuerDN[SM_AGENTAPI_SIZE_USERINFO];
   long nCertBinaryLen;
   char* lpszCertBinary;
} Sm_AgentApi_UserCredentials_t;
```

Parameters

This structure has the following parameters:

nChallengeReason

The original reason code from a previous authentication that has failed or been challenged.

lpszUsername

Name of the user being authenticated.

lpszPassword

Password of the user being authenticated.

lpszCertUserDN

This field should be set to null. Specify the complete certificate data, including the user DN, in lpszCertBinary.Existing agent applications that specify a user DN in this field are not required to change the value to null. A user DN value is supported for backward compatibility.

lpszCertIssuerDN

This field should be set to null. Specify the complete certificate data, including the issuer DN, in lpszCertBinary.Existnnnnning agent applications that specify an issuer DN in this field are not required to change the value to null. An issuer DN value is supported for backward compatibility. If lpszCertUserDN is null, lpszCertIssuerDN is ignored.

nCertBinaryLen

Number of characters in the Binary Certificate.

lpszCertBinary

Pointer to the certificate data.

Agent API Function Declarations (C)

The following table summarizes the C language functions that are used in the Agent API. The functions appear alphabetically.

Function	**Description**
Sm_AgentApi_Audit() (see page 63)	Audits a transaction.
Sm_AgentApi_Authorize() (see page 64)	Determines if a user has access to a resource.
Sm_AgentApi_ChangePassword() (see page 67)	Changes a user's password.
Sm_AgentApi_CreateSSOToken()	Produces an encrypted single sign-on token that can be shared between standard SiteMinder Web Agents and custom agents.
Sm_AgentApi_DecodeSSOToken()	Decodes a single sign-on token.
Sm_AgentApi_DelSessionVariables() (see page 73)	Deletes the specified session variables from the session store.
Sm_AgentApi_DoManagement() (see page 75)	Requests agent commands.
Sm_AgentApi_FreeAttributes() (see page 76)	Frees the buffer of response attributes.
Sm_AgentApi_FreeServers() (see page 77)	Frees an array of server structures after an Sm_AgentApi_GetConfig() call.
Sm_AgentApi_GetAgentApiUpdateVersion() (see page 77)	Retrieves the current API update version.
Sm_AgentApi_GetAllowedTunnelBufSize() (see page 77)	Retrieves the maximum data buffer size that can be transferred in a call to Sm_AgentApi_Tunnel().
Sm_AgentApi_GetConfig() (see page 79)	Retrieves agent configuration settings as defined either in the Registry (for Microsoft Windows only) or in an agent configuration file.

Function	Description
Sm_AgentApi_GetSessionVariables() (see page 82)	Retrieves the values of existing session variables.
Sm_AgentApi_Init() (see page 85)	Initializes the Agent API to set up connections to the Policy Servers.
Sm_AgentApi_IsProtected() (see page 87)	Determines if a resource is protected.
Sm_AgentApi_Login() (see page 89)	Performs session login and validation.
Sm_AgentApi_Logout() (see page 92)	Logs a user out of a session.
Sm_AgentApi_MakeCertificateHash() (see page 93)	Generates a hash of a certificate.
Sm_AgentApi_SetDefaultAgentId() (see page 95)	Sets the name of a custom agent that is configured through Central Host Configuration.
Sm_AgentApi_SetSessionVariables() (see page 96)	Creates new session variables or updates existing session variables.
Sm_AgentApi_Tunnel() (see page 99)	Transfers data between a remote service on the Policy Server side and your agent.
Sm_AgentApi_UnInit() (see page 101)	Uninitializes the Agent API.
Sm_AgentApi_UpdateAttributes() (see page 102)	Updates response attributes.

More Information:

Agent Call Sequence (see page 37)

Function Return Codes

The Agent API functions can use some or all of the following return codes:

Name	Value
SM_AGENTAPI_NOCONNECTION	-3
SM_AGENTAPI_TIMEOUT	-2
SM_AGENTAPI_FAILURE	-1
SM_AGENTAPI_SUCCESS	0
SM_AGENTAPI_YES	1

Name	Value
SM_AGENTAPI_NO	2
SM_AGENTAPI_CHALLENGE	3

Sm_AgentApi_Audit()

Call this function if the authorizations are to be done out of agent cache.

Syntax

```
int Sm_AgentApi_Audit (
   const void*                          pHandle,
   const char*                          lpszClientIpAddr,
   const char*                          lpszTransactionId,
   const Sm_AgentApi_ResourceContext_t*  pResourceContext,
   const Sm_AgentApi_Realm_t*           pRealm,
   Sm_AgentApi_Session_t*               pSession
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
lpszClientIpAddr	I	The IP address of the client where the user is logging in. This is an optional parameter. If the client IP begins with a star (*), the Policy Server logs the IP address but does not validate it against a session specification.
lpszTransactionId	I	The ID that the agent uses to associate application activity with security activity. The Policy Server logs this ID. This is an optional parameter.
pResourceContext	I	A defined resource definition structure.
pRealm	I	A realm definition structure.
pSession	I	A session definition structure.

ReturnValues

- SM_AGENTAPI_YES. The operation succeeded.
- SM_AGENTAPI_NO. The operation failed.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Sm_AgentApi_Authorize()

Determines if a defined user is authorized by SiteMinder to perform a defined action on a defined resource and returns response attributes about the user with respect to the resource.

Syntax

```
int SM_EXTERN Sm_AgentApi_Authorize (
   const void*                           pHandle,
   const char*                           lpszClientIpAddr,
   const char*                           lpszTransactionId,
   const Sm_AgentApi_ResourceContext_t*  pResourceContext,
   const Sm_AgentApi_Realm_t*            pRealm,
   Sm_AgentApi_Session_t*                pSession,
   long*                                 pNumAttributes,
   Sm_AgentApi_Attribute_t**             ppAttributes
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
lpszClientIpAddr	I	The IP address of the client asking for the resource. This parameter is optional. If the client IP begins with a star (*), the Policy Server logs the IP address but does not validate it against a session specification.
lpszTransactionId	I	The ID that the agent uses to associate application activity with security activity. The Policy Server logs this ID. This is an optional parameter.
pResourceContext	I	A resource definition structure.

Parameter	I/O	Description
pRealm	I	A realm definition structure.
pSession	I	A session definition structure.
pNumAttributes	O	The number of returned attributes.
ppAttributes	O	A pointer to an array of response attribute definition structures.

Return Values

- SM_AGENTAPI_YES. The user is authorized.
- SM_AGENTAPI_NO. The user is not authorized.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Example

See the sample smagentexample.cpp for an example of this function.

Sm_AgentApi_AuthorizeDLP()

Determines if a defined user is authorized by SiteMinder DLP integration to access the resource and returns response attributes about the user with respect to the resource.

Syntax

```
int SM_EXTERN Sm_AgentApi_AuthorizeDLP (
   const void*                         pHandle,
   const char*                         lpszClientIpAddr,
   const char*                         lpszTransactionId,
   const Sm_AgentApi_ResourceContext_t*  pResourceContext,
   const Sm_AgentApi_Realm_t*          pRealm,
   Sm_AgentApi_Session_t*              pSession,
   long*                               pNumAttributes,
   Sm_AgentApi_Attribute_t**           ppAttributes
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().

Parameter	I/O	Description
lpszClientIpAddr	I	The IP address of the client asking for the resource. This parameter is optional. If the client IP begins with a star (*), the Policy Server logs the IP address but does not validate it against a session specification.
lpszTransactionId	I	The ID that the agent uses to associate application activity with security activity. The Policy Server logs this ID. This is an optional parameter.
pResourceContext	I	A resource definition structure.
pRealm	I	A realm definition structure.
pSession	I	A session definition structure.
pNumAttributes	O	The number of returned attributes.
ppAttributes	O	A pointer to an array of response attribute definition structures.

Return Values

- SM_AGENTAPI_YES. The user is authorized.
- SM_AGENTAPI_NO. The user is not authorized.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Sm_AgentApi_ChangePassword()

Changes a user's password. The resulting attributes plus the reason code from the session object are used to construct the correct password services redirect.

Syntax

```
int SM_EXTERN Sm_AgentApi_ChangePassword (
   const void* pHandle,
   const char* lpszClientIpAddr,
   const char* lpszNewPassword,
   const char* pszTokenValue,
   const Sm_AgentApiResourceContext_t* pResourceContext,
   const Sm_AgentApi_Realm_t* pRealm,
   const Sm_AgentApi_UserCredentials_t* pUserCredentials,
   Sm_AgentApi_Session_t* pSession,
   long * numAttributes,
   Sm_AgentApi_Attribute_t** ppAttributes
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
lpszClientIpAddr	I	The IP address of the client asking for the resource. This parameter is optional. If the client IP begins with a star (*), the Policy Server logs the IP address but does not validate it against a session specification.
lpszNewPassword	I	The new password (string) to which the user wants to change.
pszTokenValue	I	The token that is exchanged between the Policy Server and the Web Agent in the case of a Password Services redirect. Use this parameter to send to the Policy Server an extracted SMTOKEN from the ppAttributes (returned by Sm_AgentApi_ChangePassword()). This value can be NULL if the nChallengeReason value of the pUserCredentials parameter is set to Sm_Api_Reason_PWSelfChange (indicating a user-initiated password change).
pResourceContext	I	A pointer to a resource definition structure.
pRealm	I	A realm definition structure.

Parameter	I/O	Description
pUserCredentials	I	A user credentials definition structure.
pSession	O	A user session definition structure.
pNumAttributes	O	The number of attributes in *ppAttributes*.
ppAttributes	O	A pointer to an array of response attribute definition structures.

Return Values

- SM_AGENTAPI_YES. The user's password was changed.
- SM_AGENTAPI_NO. The user's password was not changed.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_TIMEOUT. The function timed out.
- SM_AGENTAPI_FAILURE. The password was not changed

Sm_AgentApi_CreateSSOToken()

Produces an encrypted token of session and other information that can be shared between standard SiteMinder Web Agents and custom agents. The mutual access to this information allows a custom agent to participate in a single sign-on environment with a standard SiteMinder Web Agent.

Syntax

```
int SM_EXTERN Sm_AgentApi_CreateSSOToken (
   const void*                      pHandle,
   Sm_AgentApi_Session_t*           pSession,
   long                             nNumAttributes,
   Sm_AgentApi_Attribute_t*         pTokenAttributes,
   long*                            pNumSSOTokenLength,
   char*                            lpszSSOToken
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
pSession	I	Session information returned by the Sm_AgentApi_Login() call.

Parameter	I/O	Description
nNum Attributes	I	The number of attributes to include in the token. The attributes are specified in the parameter *pTokenAttributes*.
pToken Attributes	I	The user attributes to include in the token. Valid values: ■ SM_AGENTAPI_ATTR_USERDN. The user's distinguished name. ■ SM_AGENTAPI_ATTR_USERNAME. The user's name. ■ SM_AGENTAPI_ATTR_CLIENTIP. The IP address of the machine where the user initiated a request for a protected resource. Any other attribute is ignored. The fields in the Sm_AgentApi_Attribute_t structure that apply to this function are: ■ *nAttributeId* (one of the above values) ■ *nAttributeLen* ■ *lpszAttributeValue*
pNumSSOToken Length	I, O	The length of the *lpszSSOToken* buffer passed in to receive the token. The maximum size is specified by SSO_TOKEN_MAX_SIZE, defined in SmAgentAPI.h. Allow space for the null-terminator character. On output, this parameter is set to the actual length of the returned token, including the null-terminator character.
lpszSSOToken	O	The token returned from this function. Write this token to the SMSESSION cookie.

Return Values

- SM_AGENTAPI_SUCCESS. The operation successful.
- SM_AGENTAPI_FAILURE. The token was not created.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.

Remarks

This function associates the user attribute information specified in the *pTokenAttributes* parameter with session and other attribute information returned from the call to Sm_AgentApi_Login(). The information in the resulting token can be shared between standard SiteMinder Web Agents and custom agents, allowing single sign-on operations between the standard and custom agents.

This call does not allocate any memory.

To decode token information, call Sm_AgentApi_DecodeSSOToken().

Sm_AgentApi_DecodeSSOToken()

Decodes a single sign-on token and returns a subset of its attributes. Optionally, you can update the token's last-access timestamp, and then update the SMSESSION cookie with the new token.

Syntax

```
int SM_EXTERN Sm_AgentApi_DecodeSSOToken (
   const void*                  pHandle,
   const char*                  lpszSSOToken,
   long*                        nTokenVersion,
   long*                        pThirdPartyToken,
   long*                        pNumAttributes,
   Sm_AgentApi_Attribute_t**    ppTokenAttributes,
   long                         nUpdateToken,
   long*                        pNumUpdatedSSOTokenLength,
   char*                        lpszUpdatedSSOToken
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().

Parameter	I/O	Description
lpszSSOToken	I	Null-terminated character array that contains the token to be decoded. The custom agent finds the token in either of these locations: ■ If the token was created by a custom agent, the token is returned in the output parameter *lpszSSOToken* from the call to Sm_AgentApi_CreateSSOToken(). ■ If the token was created by a standard SiteMinder Web Agent, the token is contained in the SMSESSION cookie. The custom agent is responsible for extracting the contents of the cookie and assigning it to this parameter.
nTokenVersion	O	The SiteMinder version of the token.
pThirdParty Token	O	A non-zero value indicates that the token was originally produced by a custom (third-party) agent and has not yet been updated by a standard SiteMinder agent.
pNumAttributes	O	The number of attributes retrieved from the token. The attributes are specified in the parameter *ppTokenAttributes*.
ppToken Attributes	O	The attributes extracted from the token. Valid values: ■ SM_AGENTAPI_ATTR_USERDN ■ SM_AGENTAPI_ATTR_SESSIONSPEC ■ SM_AGENTAPI_ATTR_SESSIONID ■ SM_AGENTAPI_ATTR_USERNAME ■ SM_AGENTAPI_ATTR_CLIENTIP ■ SM_AGENTAPI_ATTR_DEVICENAME ■ SM_AGENTAPI_ATTR_IDLESESSIONTIMEOUT ■ SM_AGENTAPI_ATTR_MAXSESSIONTIMEOUT ■ SM_AGENTAPI_ATTR_STARTSESSIONTIME ■ SM_AGENTAPI_ATTR_LASTSESSIONTIME

Parameter	I/O	Description
nUpdateToken	I	A non-zero value indicates that an updated token is requested. The updated token is written to *lpszUpdatedSSOToken*. Set the *nUpdateToken* flag to a non-zero value if you want to update the attribute SM_AGENTAPI_ATTR_LASTSESSIONTIME.
pNumUpdatedSSO TokenLength	I, O	The length of the *lpszUpdatedSSOToken* buffer to receive the token. The maximum size is specified by SSO_TOKEN_MAX_SIZE, defined in SmAgentAPI.h. Allow space for the null-terminator character. On output, this parameter is set to the actual length of the returned token, including the null-terminator character.
lpszUpdated SSOToken	O	The updated token returned from this function. Write this token to the SMSESSION cookie. A token is returned only if *nUpdateToken* is set to a non-zero value.

Returns

- SM_AGENTAPI_SUCCESS. The operation succeeded.
- SM_AGENTAPI_FAILURE. The token was not decoded.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.

Remarks

This function accepts a single sign-on token as input and returns a subset of the token's attributes.

You can update the token's last-access timestamp. To do so, assign a non-zero value to the parameter *nUpdateToken*. The token that includes the updated timestamp is returned in *lpszUpdatedSSOToken*. Write the updated token to the SMSESSION cookie.

This function allocates memory for the attribute list. To deallocate this memory, call Sm_AgentApi_FreeAttributes().

To create a single sign-on token, call Sm_AgentApi_CreateSSOToken().

Sm_AgentApi_DelSessionVariables()

Deletes the specified session variables from the session store.

Syntax

```
int Sm_AgentApi_DelSessionVariables (
   const void*                          pHandle,
   const Sm_AgentApi_ResourceContext_t* pResourceContext,
   const char*                          lpszSessionId,
   long                                 nNumReqAttributes,
   Sm_AgentApi_Attribute_t*             pReqAttributes,
   long*                                pRespNumAttributes,
   Sm_AgentApi_Attribute_t**            ppRespAttributes
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
pResourceContext	I	Reserved for future use. Set all fields to 0.
lpszSessionId	I	A unique identifier of the session for which the variable is to be deleted. Variables can only be deleted for an active session. After a successful login, the session ID is returned in the *lpszSessionId* field of the structure Sm_AgentApi_Session_t.
nNumReqAttributes	I	Size of the array of session variables in *pReqAttributes*.
pReqAttributes	I	An array of attributes representing the names of session variables to be deleted. Set the variable name in the field *lpszAttributeOid* of structure Sm_AgentApi_Attribute_t. Set the *nAttributeFlags* field of the Sm_AgentApi_Attribute_t structure to SM_AGENTAPI_REQATTR_FLAGS_NONE. The structure’s *nAttributeId* and *nAttributeTTL* fields are ignored.
pRespNumAttributes	O	Size of the array of responses, if any, in *ppRespAttributes*.

Parameter	I/O	Description
ppRespAttributes	O	If all the specified variables are deleted, no response attributes are returned in this parameter. If this function returns SM_AGENTAPI_UNRESOLVED, the response attribute result set will contain variables that could not be deleted. Also, each response attribute will have the field *nAttributeFlags* of the structure Sm_AgentApi_Attribute_t set to SM_AGENTAPI_RESPATTR_FLAGS_UNRESOLVED. The structure's *nAttributeId* and *nAttributeTTL* fields are ignored.

Return Values

- SM_AGENTAPI_YES. The operation succeeded.
- SM_AGENTAPI_NO. The call was refused.
- SM_AGENTAPI_UNRESOLVED. Some variables could not be deleted. See the description of the *ppRespAttributes* parameter.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Remarks

This function fails if the session referenced by *lpszSessionId* is not active, or the connection to the session store is lost.

To release the memory allocated for any variables returned in the *ppRespAttributes* parameter, call Sm_AgentApi_FreeAttributes().

More Information:

Sm_AgentApi_FreeAttributes() (see page 76)

Sm_AgentApi_DoManagement()

Requests agent commands from the Policy Server. Agent commands indicate work to be performed by agents.

Syntax

```
int SM_EXTERN Sm_AgentApi_DoManagement (
   const void*                        pHandle,
   Sm_AgentApi_ManagementContext_t*   pManagementContext,
   long*                              pNumAttributes,
   Sm_AgentApi_Attribute_t**          ppAttributes
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
pMananagement Context	I	A management definition structure. The agent uses this structure to define a management command.
pNumAttributes	I	The number of attributes in *ppAttributes*.
ppAttributes	O	A pointer to an array of requested attribute definition structures. One or more of the following attributes may be returned: ■ SM_AGENTAPI_AGENT_KEY_UPDATE_NEXT ■ SM_AGENTAPI_AGENT_KEY_UPDATE_LAST ■ SM_AGENTAPI_AGENT_KEY_UPDATE_CURRENT ■ SM_AGENTAPI_AGENT_KEY_UPDATE_PERSISTENT ■ SM_AGENTAPI_CACHE_FLUSH_ALL ■ SM_AGENTAPI_CACHE_FLUSH_ALL_USERS ■ SM_AGENTAPI_CACHE_FLUSH_THIS_USER ■ SM_AGENTAPI_CACHE_FLUSH_ALL_REALMS ■ SM_AGENTAPI_CACHE_FLUSH_THIS_REALM

Return Values

- SM_AGENTAPI_YES. The operation succeeded.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.

- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Example

See the function SmAgentExample::Sm_DoManagement() in the example application smagentexample.cpp.

Sm_AgentApi_FreeAttributes()

Frees the buffer of response attributes.

Syntax

```
void Sm_AgentApi_FreeAttributes (
   const long                        nNumAttributes,
   const Sm_AgentApi_Attribute_t*    pAttributes
);
```

Parameter	I/O	Description
nNumAttributes	I	The number of attributes in *pAttributes*.
pAttributes	I	A pointer to an array of response attribute definition structures.

Example

See the function SmAgentExample::Sm_Login() in the example application smagentexample.cpp.

Sm_AgentApi_FreeServers()

Frees an array of server structures after a call to Sm_AgentApi_GetConfig().

Syntax

```
void SM_EXTERN Sm_AgentApi_FreeServers (
        Sm_AgentApi_Server_t* pServers
);
```

Parameter	I/O	Description
pServers	I	The server structures to free.

Sm_AgentApi_GetAgentApiUpdateVersion()

Retrieves the current API update version in SM_AGENTAPI_UPDATE_VERSION.

For example, if the current version of the Agent API is v6.0 SP3, you can call Sm_AgentApi_GetAgentApiUpdateVersion() to verify that the custom program was compiled against Agent API v6.0 SP3.

Syntax

```
int SM_EXTERN Sm_AgentApi_GetAgentApiUpdateVersion();
```

Sm_AgentApi_GetAllowedTunnelBufSize()

Retrieves the maximum data buffer size that can be transferred in the Sm_AgentApi_Tunnel() function call.

Syntax

```
long SM_EXTERN Sm_AgentApi_GetAllowedTunnelBufSize (
   void* pHandle,
   int nServer
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().

Parameter	I/O	Description
nServer	I	The server that will process the request at the time of the tunnel call. One of these values, defined in smAgentAPI.h: /* server ports */ /* authorization server */ #define SM_AGENTAPI_AZ_SERVER 0 /* authentication server */ #define SM_AGENTAPI_AUTH_SERVER 1 /* accounting server */ #define SM_AGENTAPI_ACCT_SERVER 2

Returns

The maximum size of the buffer.

Remarks

This function was introduced in SDK v5.5. Beginning with that release, maximum allowable buffer sizes are larger than in previous releases.

Sm_AgentApi_GetConfig()

Retrieves configuration information for an agent.

This function requires an Agent API version of v5.0 or later.

This function can read configuration information either from a configuration file or, on a Microsoft Windows platform, from the Windows Registry.

Syntax

```
int SM_EXTERN Sm_AgentApi_GetConfig (
   Sm_AgentApi_Init_t*   pInit,
   const char*           lpszAgentName,
   const char*           lpszPath
);
```

Parameter	I/O	Description
pInit	O	A pointer to an Sm_AgentApi_Init_t structure. Sm_AgentApi_GetConfig() retrieves agent configuration information and copies each setting to a field of *pInit*. To free the array of server structures that may be placed in Sm_AgentApi_Init_t, call Sm_AgentApi_FreeServers().
lpszAgentName	I	Name of the target agent. The function Sm_AgentApi_GetConfig() searches the agentname list for the agent name specified in *lpszAgentName*. If *lpszAgentName* is empty, Sm_AgentApi_GetConfig() searches for the default agent name. In the case of a v5.x or v6.x custom agent that is configured through central host configuration, this parameter is ignored.

Parameter	I/O	Description
lpszPath	I	Full path and name of the configuration file that contains the agent configuration information. Set this parameter as an empty string ("") only for the 4QMRx web agent on IIS. If *lpszPath* is an empty string, Sm_AgentApi_GetConfig() checks whether the agent is written for a WIN32 platform. If it is, the function searches the Windows Registry for configuration information. If it is not, the function returns SM_AGENTAPI_FAILURE.

Return Values

- SM_AGENTAPI_SUCCESS. Configuration retrieval succeeded.
- SM_AGENTAPI_FAILURE. Configuration retrieval failed.
- SM_AGENTAPI_FAILURE is returned under any of these conditions:
  - No agent name is specified in *lpszAgentName*, and Sm_AgentApi_GetConfig() cannot get the default agent name
  - An agent name is specified in *lpszAgentName*, but Sm_AgentApi_GetConfig() cannot find that agent name
  - Sm_AgentApi_GetConfig() cannot find at least one SiteMinder Policy Server configuration
  - There is no path specified in *lpszPath*, and one of the following occurs:
    - The platform is not WIN32, so there are no Registry settings to retrieve.
    - Sm_AgentApi_GetConfig() cannot retrieve one of the required settings from the Registry. For example, it cannot open the Registry key that holds the shared secret.
    - There is a path specified in *lpszPath*, but Sm_AgentApi_GetConfig() cannot open and read the configuration file named in *lpszPath*.

Remarks

This function *must* be called by custom agents that are configured through central host configuration.

With v5.x or later agent connectivity:

- The function's *lpszPath* parameter references the SmHost.conf file created when the trusted host was registered with the Policy Server, or a WebAgent.conf file that references SmHost.conf.
- The lpszAgentName parameter is ignored.

With 4.x agent connectivity:

- The agent name is specified in the *lpszAgentName* parameter.
- A v4.x WebAgent.conf file is referenced in *lpszPath*.

Sm_AgentApi_GetConfig() checks the value of *lpszPath* to find the path and name of a configuration file:

- If *lpszPath* is empty and the agent is written for a Microsoft Windows platform, Sm_AgentApi_GetConfig() searches the Windows Registry for the configuration information.
- If *lpszPath* is empty and the agent is running on a UNIX system, Sm_AgentApi_GetConfig() returns SM_AGENTAPI_FAILURE.

If Sm_AgentApi_GetConfig() cannot find the agent name, it uses the default agent name. If the default agent name cannot be retrieved, the function returns SM_AGENTAPI_FAILURE.

When Sm_AgentApi_GetConfig() locates the configuration information for the correct agent, it copies the information into the fields of an initialization structure (Sm_AgentApi_Init_t). The parameter *pInit* points to this initialization structure. For example, suppose the agent name parameter contains the string Agent1 and the agentname list of the configuration file is set as follows:

```
agentname="Agent1,123.112.12.12"
```

In this circumstance, Sm_AgentApi_GetConfig() sets the *lpszHostName* field of initialization structure *pInit* to Agent1. The IP Address is ignored. Sm_AgentApi_GetConfig() then retrieves the information for the other fields of the initialization structure.

Sm_AgentApi_GetMaxTunnelBufSize()

Deprecated in SDK v5.5. Replaced by Sm_AgentApi_GetAllowedTunnelBufSize() (see page 77).

Sm_AgentApi_GetSessionVariables()

Retrieves the values of existing session variables.

Syntax

```
int Sm_AgentApi_GetSessionVariables (
   const void*                          pHandle,
   const Sm_AgentApi_ResourceContext_t* pResourceContext,
   const char*                          lpszSessionId,
   long                                 nNumReqAttributes,
   Sm_AgentApi_Attribute_t*             pReqAttributes,
   long*                                pRespNumAttributes,
   Sm_AgentApi_Attribute_t**            ppRespAttributes
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
pResourceContext	I	Reserved for future use. Set all fields to 0.
lpszSessionId	I	A unique identifier of the session for which the variable is to be retrieved. Variables can only be retrieved for an active session. After a successful login, the session ID is returned in the *lpszSessionId* field of the structure Sm_AgentApi_Session_t.
nNumReqAttributes	I	Size of the array of attributes in *pReqAttributes*.

Parameter	I/O	Description
pReqAttributes		An array of attributes representing the names of session variables to be retrieved. Set the variable name in the field *lpszAttributeOid* of structure Sm_AgentApi_Attribute_t. Set the *nAttributeFlags* field of the Sm_AgentApi_Attribute_t structure to one of these values: ■ SM_AGENTAPI_REQATTR_FLAGS_NONE Retrieve the named variable, but don't delete it. ■ SM_AGENTAPI_REQATTR_FLAGS_DELETE Delete the variable from the session store after retrieving it. The structure's *nAttributeId* and *nAttributeTTL* fields are ignored. Set the *nAttributeFlags* field of the Sm_AgentApi_Attribute_t structure to one of these values: ■ SM_AGENTAPI_REQATTR_FLAGS_NONE Retrieve the named variable, but don't delete it. ■ SM_AGENTAPI_REQATTR_FLAGS_DELETE Delete the variable from the session store after retrieving it. The structure's *nAttributeId* and *nAttributeTTL* fields are ignored.
pRespNumAttributes	O	Size of the array of responses in *ppRespAttributes*.

Parameter	I/O	Description
ppRespAttributes	O	An array of response attributes representing session variables and their values. The value returned from this function indicates the contents of the response attribute result set, as follows: ■ SM_AGENTAPI_UNRESOLVED Some variables could not be fetched. If a given variable can't be fetched, the associated *nAttributeFlags* field of the Sm_AgentApi_Attribute_t structure is set to SM_AGENTAPI_ RESPATTR_FLAGS_UNRESOLVED. For the variables that are fetched, the *nAttributeFlags* field is set to SM_AGENTAPI_RESPATTR_FLAGS_NONE. ■ SM_AGENTAPI_YES All requested variables and their values were fetched from the session store and returned in *ppRespAttributes*. The structure's *nAttributeId* and *nAttributeTTL* fields are ignored.

Return Values

- SM_AGENTAPI_YES. The operation succeeded.
- SM_AGENTAPI_NO. The call was refused.
- SM_AGENTAPI_UNRESOLVED. Some variables could not be fetched. See the description of the *ppRespAttributes* parameter.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.
- SM_AGENTAPI_SUCCESS. The operation succeeded.

Remarks

This function fails if the session referenced by *lpszSessionId* is not active, if no variables are found, or if the connection to the session store is lost.

To delete a variable from the session store after fetching it, set the SM_AGENTAPI_REQATTR_FLAGS_DELETE flag in the *pReqAttributes* parameter. To delete a variable without fetching it, call the function Sm_AgentApi_DelSessionVariables().

To release the memory allocated for any variables returned in the *ppRespAttributes* parameter, call Sm_AgentApi_FreeAttributes() (see page 76).

Sm_AgentApi_Init()

Initializes the Agent API and sets up connections to the Policy Server. This function is called once per agent.

Note: This call succeeds even if a connection to the Policy Server cannot be established immediately. The Agent API will keep trying to reconnect. See the Remarks for more information.

Syntax

```
int SM_EXTERN Sm_AgentApi_Init (
   const Sm_AgentApi_Init_t*  pInitStruct,
   void**                     ppHandle
);
```

Parameter	I/O	Description
pInitStruct	I	A pointer to information about the server.
ppHandle	O	The address of a pointer to hold the returned handle for this API session. This is an opaque type.

Returns

- SM_AGENTAPI_SUCCESS. Initialization succeeded.
- SM_AGENTAPI_FAILURE. Initialization failed.

Remarks

All agents should issue the DoManagement() call and specify the SM_AGENTAPI_MANAGEMENT_SET_AGENT_INFO command once at startup.

This function is designed to fail only when a connection to the Policy Server is established, but the shared secret and/or agent name are incorrect. In all other circumstances, this function returns SM_AGENTAPI_SUCCESS, such as in the following circumstances:

- An incorrect Policy Server IP address and/or port number are provided during the initialization operation
- A correct Policy Server IP address and port number are provided, but the Policy Server is down

In these cases, the Agent API returns a status of success and continues to try to establish the connection to the Policy Server (the connection layer does not know if the information provided is correct or incorrect). You should not assume that a connection to the Policy Server is established if the Sm_AgentApi_Init() function succeeds.

You are responsible for deallocating memory for your custom agent. When you initialize the Agent API with Sm_AgentAPI_Init(), all information in the Sm_AgentApi_Init_t structure is copied, allowing you to deallocate the structure's memory after initialization.

Example

See the function SmAgentExample::Sm_Init() in the example application smagentexample.cpp.

Sm_AgentApi_IsProtected()

Checks if the defined resource is protected by SiteMinder.

Syntax

```
int SM_EXTERN Sm_AgentApi_IsProtected (
   const void                          pHandle,
   const char*                         lpszClientIpAddr,
   const Sm_AgentApi_ResourceContext_t*   pResourceContext,
   Sm_AgentApi_Realm_t*                pRealm
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
lpszClientIpAddr	I	The IP address of the client asking for the resource. This is an optional parameter.
pResourceContext	I	A resource definition structure.
pRealm	O	A realm definition structure. The resource is protected by the returned realm.

Return Values

- SM_AGENTAPI_YES. The resource is protected.
- SM_AGENTAPI_NO. The resource is not protected.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Example

See the function SmAgentExample::Sm_IsProtected() in the example application smagentexample.cpp.

Sm_AgentApi_IsProtectedDLP()

Checks if the defined resource is protected by SiteMinder DLP integration.

Syntax

```
int SM_EXTERN Sm_AgentApi_IsProtectedDLP (
   const void                           pHandle,
   const char*                          lpszClientIpAddr,
   const Sm_AgentApi_ResourceContext_t*   pResourceContext,
   Sm_AgentApi_Realm_t*                 pRealm
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
lpszClientIpAddr	I	The IP address of the client asking for the resource. This is an optional parameter.
pResourceContext	I	A resource definition structure.
pRealm	O	A realm definition structure. The resource is protected by the returned realm.

Return Values

- SM_AGENTAPI_YES. The resource is protected by DLP integration.
- SM_AGENTAPI_NO. The resource is not protected by DLP integration.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Sm_AgentApi_Login()

This function performs session login and session validation.

The Policy Server authenticates user credentials during session login and validates the session specification during session validation. Whether the Policy Server performs session login or session validation depends on whether a session specification is defined in the field *lpszSessionSpec* of the structure Sm_AgentApi_Session_t, as follows:

- If the session specification exists, the Policy Server performs session validation. During session validation, if the *lpszSessionSpec* field has a length, the Policy Server takes the following actions:
    - Verifies that the session has not expired based on the *nMaxTimeout* field of Sm_AgentApi_Session_t.
    - Checks the IP address.
    - Verifies that the user is in the user directory and has not been disabled.
- If the session specification does not exist, the Policy Server performs session login. During session login, if the session ID is specified, it will be used as the session ID upon successful authentication.

Syntax

```
int SM_EXTERN Sm_AgentApi_Login (
   const void*                           pHandle,
   const char*                           lpszClientIpAddr,
   const Sm_AgentApi_ResourceContext_t*  pResourceContext,
   const Sm_AgentApi_Realm_t*            pRealm,
   const Sm_AgentApi_UserCredentials_t*  pUserCredentials,
   Sm_AgentApi_Session_t*                pSession,
   long*                                 pNumAttributes,
   Sm_AgentApi_Attribute_t**             ppAttributes
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
lpszClientIpAddr	I	The IP address of the client that the user is logging from. This is an optional parameter. If the client IP begins with a star (*), the Policy Server logs the IP address but does not validate it against a session specification.
pResourceContext	I	A pointer to a resource definition structure.

Parameter	**I/O**	**Description**
pRealm	I	A realm definition structure.
pUserCredentials	I	A user credentials definition structure.
pSession	O	A User Session definition structure.
pNumAttributes	O	The number of attributes in *ppAttributes*.
ppAttributes	O	A pointer to an array of response attribute definition structures. This function returns the following attributes, when available: ■ SM_AGENTAPI_ATTR_AUTH_DIR_OID ■ SM_AGENTAPI_ATTR_AUTH_DIR_NAME ■ SM_AGENTAPI_ATTR_AUTH_DIR_SERVER ■ SM_AGENTAPI_ATTR_AUTH_DIR_NAMESPACE ■ SM_AGENTAPI_ATTR_USERMSG ■ SM_AGENTAPI_ATTR_USERDN ■ SM_AGENTAPI_ATTR_USERUNIVERSALID ■ SM_AGENTAPI_ATTR_IDENTITYSPEC See Remarks for information about the attributes that are set when a resource is protected by an anonymous authentication scheme.

Return Values

- SM_AGENTAPI_YES. The user was authenticated.
- SM_AGENTAPI_NO. The user was not authenticated.
- SM_AGENTAPI_CHALLENGE. A challenge is required for authentication.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Remarks

Response attributes can be returned when authentication events occur. Both well-known and policy-based attributes can be returned, as described in Response Attributes. For example, upon successful authentication, a response could return the user's DN.

When a resource is protected by an anonymous authentication scheme, only the following attributes are set:

- SM_AGENTAPI_ATTR_USERDN. Set with the SessionID for the anonymous session.
- SM_AGENTAPI_ATTR_IDENTITYSPEC. Set with the globally unique identity ticket for the anonymous session.

Supply only the required credentials (as determined by a call to Sm_AgentApi_IsProtected(), which should be called before Sm_AgentApi_Login()). Unused fields in the user credentials structure must be zero-initialized.

Sm_AgentApi_Login() returns attributes in the Sm_AgentApi_Attribute_t structure. Call Sm_AgentApi_FreeAttributes() to release the attributes.

On successful login, the Sm_AgentApi_Session_t structure is populated with the session specification. If you allocated memory for this structure, it is your responsibility to deallocate it.

Example

See the example application smagentexample.cpp for an example of this function.

Sm_AgentApi_Logout()

Logs a user out of a user session and issues an event. No database is updated.

When a user logs out, you must explicitly terminate the session by discarding the session specification.

This function does not deallocate memory. It is your responsibility to deallocate any memory you allocated for your custom agent.

Syntax

```
int SM_EXTERN Sm_AgentApi_Logout (
   const void*                  pHandle,
   const char*                  lpszClientIpAddr,
   const Sm_AgentApi_Session_t*  pSession
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *pHandle* of Sm_AgentApi_Init().
lpszClientIpAddr	I	The IP address of the client that the user is logging out from. This is an optional parameter. If the client IP begins with a star (*), the Policy Server logs the IP address but does not validate it against a session specification.
pSession	I	A session definition structure (Sm_AgentApi_Session_t) for the user's session. *The nreason* field will be passed in the event issued by the Policy Server. See Sm_Api_Reason_t.

Return Values

- SM_AGENTAPI_YES. The user logged out successfully.
- SM_AGENTAPI_NO. The user was not logged out.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.

- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Remarks

To terminate a user's session when a user logs out, you must discard the session specification. You can do so by re-initializing the session specification in the Sm_AgentApi_Session_t structure, as illustrated in the following code:

```
iResult = Sm_AgentApi_Logout (
                pHandle,
                SMAPI_SAMPLE_AGENTIP,
                &pSession);

if (SM_AGENTAPI_YES==iResult)
      memset(&pSession,0,sizeof(Sm_AgentApi_Session_t));
```

Sm_AgentApi_MakeCertificateHash()

Use this function to generate an SHA1 hash of a binary certificate. The hash should be placed in the binary certificate in the user credentials structure. For this function to work properly, the *iCertHashLen* parameter must be greater than or equal to 20 bytes. This function can be used to generate SHA1 hashes of an arbitrary buffer.

Syntax

```
int Sm_AgentApi_MakeCertificateHash (
   const unsigned char*   pCertificateData,
   const int              nCertLen,
   unsigned char*         pCertHash,
   const int              nCertHashLen
);
```

Parameter	I/O	Description
pCertificateData	I	A pointer to a buffer.
nCertLen	I	The length in bytes of the buffer.
pCertHash	O	The buffer into which the hash is placed.
nCertHashLen	I	The size of the hash buffer in bytes. The size must be at least 20.

Return Values

- SM_AGENTAPI_SUCCESS. The operation succeeded.
- SM_AGENTAPI_FAILURE. The size is too small.

Sm_AgentApi_SetAgentInstanceInfo()

Sets the agent instance information specified in the agent discovery data structure. All values in the Sm_AgentApi_AgentDiscovery_t are initialized to "unknown". Call this function after Sm_AgentApi_Init() to set the values of these attributes when you want the agent to come under the purview of agent discovery.

Syntax

```
int SM_EXTERN Sm_AgentApi_SetAgentInstnaceInfo (
   const void*                                          pHandle,
   Sm_AgentApi_AgentDiscovery_t*          arrParams[]
   int                            nCount
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
arrParams[]	I	Array containing name/value pairs of attributes for the agent instance.
nCount	I	Number of elements in the array of attributes.

Return Values

- SM_AGENTAPI_YES. The resource is protected.
- SM_AGENTAPI_NO. The resource is not protected.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Sm_AgentApi_SetDefaultAgentId()

Sets the name of a v5.x or later agent that is configured through Central Host Configuration.

Syntax

```
int SM_EXTERN Sm_AgentApi_SetDefaultAgentId(
   const char *pszAgentIdentity,
   void* pHandle
);
```

Parameter	I/O	Description
pszAgentIdentity	I	Specifies the name to set as the default agent name. This name must match the name of the corresponding agent object on the Policy Server.
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().

Return Values

- SM_AGENTAPI_SUCCESS. The default agent name was set successfully.
- SM_AGENTAPI_FAILURE. Parameter *pszAgentIdentity* was NULL or exceeded SM_AGENTAPI_SIZE_NAME.
- SM_AGENTAPI_NOCONNECTION. Parameter *pHandle* was NULL.

Remarks

Call this function after calling Sm_AgentApi_Init() and before calling any other function in the Agent API. Doing so avoids having to pass the name of the agent with each Agent API request.

This function is only used with v5 or later custom agents that are configured through central host configuration.

Sm_AgentApi_SetSessionVariables()

Creates new session variables or updates existing session variables.

Syntax

```
int Sm_AgentApi_SetSessionVariables (
   const void*                          pHandle,
   const Sm_AgentApi_ResourceContext_t* pResourceContext,
   const char*                          lpszSessionId,
   long                                 nNumReqAttributes,
   Sm_AgentApi_Attribute_t*             pReqAttributes,
   long*                                pRespNumAttributes,
   Sm_AgentApi_Attribute_t**            ppRespAttributes
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().
pResourceContext	I	Reserved for future use. Set all fields to 0.
lpszSessionId	I	A unique identifier of the session for which the variable is to be set. Variables can only be set for an active session. After a successful login, the session ID is returned in the *lpszSessionId* field of the structure Sm_AgentApi_Session_t.
nNumReqAttributes	I	Size of the array of session variables in *pReqAttributes*.

Parameter	**I/O**	**Description**
pReqAttributes	I	An array of attributes representing session variable values. Set the variable name in the field *lpszAttributeOid* of structure Sm_AgentApi_Attribute_t: ■ If a variable name already exists, the associated variable value is overwritten by the value of *pReqAttributes*. ■ If a variable name doesn't exist, a new variable is created. Set the *nAttributeFlags* field of the Sm_AgentApi_Attribute_t structure to SM_AGENTAPI_REQATTR_FLAGS_NONE. The structure's *nAttributeId* and *nAttributeTTL* fields are ignored.
pRespNumAttributes	O	Size of the array of responses in *ppRespAttributes*.
ppRespAttributes	O	An array of response attributes representing variables that could not be set. If this function returns SM_AGENTAPI_UNRESOLVED, the response attribute result set will contain unresolved variables. Also, for each unresolved variable returned, the field *nAttributeFlags* of the structure Sm_AgentApi_Attribute_t will be set to SM_AGENTAPI_RESPATTR_UNRESOLVED. The structure's *nAttributeId* and *nAttributeTTL* fields are ignored.

Returns

- SM_AGENTAPI_YES. The operation succeeded.
- SM_AGENTAPI_NO. The operation failed.
- SM_AGENTAPI_UNRESOLVED. Some variables could not be set. The list of unresolved variables is returned in the *ppRespAttributes* parameter.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.
- SM_AGENTAPI_SUCCESS. The operation succeeded.

Remarks

This function fails if the session referenced by *lpszSessionId* is not active or the connection to the session store is lost.

To release the memory allocated for any unresolved variable values returned in the *ppRespAttributes* parameter, call Sm_AgentApi_FreeAttributes() (see page 76).

Sm_AgentApi_Tunnel()

Call this function to transfer data between a remote service on the Policy Server side and your agent. The Sm_AgentApi_GetMaxTunnelBufSize() function call gives you the maximum data size that can be transferred. At this time this function supports only one buffer for each call. *pServiceRequest* holds the information about the remote service that will be invoked by the Policy Server.

Note: SMTUNNEL is a predefined tunnel agent name whose shared secret is also SMTUNNEL. You can initialize a tunnel agent using these names without specifically creating the agent ahead of time. The predefined SMTUNNEL agent can only call Sm_AgentApi_Tunnel().

If you explicitly create a tunnel agent that has the name and shared secret SMTUNNEL, it is also limited to calling Sm_AgentApi_Tunnel().

If an agent named SMTUNNEL makes a call to a function other than Sm_AgentApi_Tunnel(), the Policy Server returns an error.

Syntax

```
int SM_EXTERN Sm_AgentApi_Tunnel (
  const void*                          pHandle,
  const int                            nServer,
  const char*                          lpszClientIpAddr,
  const char*                          lpszTransactionId,
  const Sm_AgentApi_ResourceContext_t*   pResourceContext,
  const Sm_AgentApi_TunnelServiceRequest_t* pServiceRequest,
  long*                                pRespNumAttributes,
  Sm_AgentApi_Attribute_t**            ppRespAttributes
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().

Parameter	**I/O**	**Description**
nServer	I	The server that will process the request at the time of the tunnel call. One of these values, defined in smAgentAPI.h: /* server ports */ /* authorization server */ #define SM_AGENTAPI_AZ_SERVER 0 /* authentication server */ #define SM_AGENTAPI_AUTH_SERVER 1 /* accounting server */ #define SM_AGENTAPI_ACCT_SERVER 2
lpszClientIpAddr	I	(Optional) The IP address of the client from which the user is logging.
lpszTransactionId	I	(Optional) The ID that the agent uses to associate application activity with security activity. The Policy Server logs this ID.
pResourceContext	I	A resource definition structure.
pServiceRequest	I	A service request definition structure.
pRespNumAttributes	O	The number of attributes in *ppRespAttributes*.
ppRespAttributes	O	A pointer to an array of response attribute definition structures. The attribute identifier SM_AGENTAPI_ATTR_SERVICE_DATA indicates that the response structure contains the data returned by the remote service. The attribute identifier SM_AGENTAPI_ATTR_STATUS_MESSAGE has the status message from the remote service.

Returns

- SM_AGENTAPI_YES. The operation succeeded.
- SM_AGENTAPI_NO. The operation failed.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Example

For an example of Sm_AgentApi_Tunnel(), see:

<install_path>\sdk\samples\smtunnelagent\smtunnelexample.cpp

Sm_AgentApi_UnInit()

Once the agent is no longer needed, uninitialize all API instances by issuing the Sm_AgentApi_UnInit() call for each API instance. This closes TCP connections to all Policy Servers.

This function does not deallocate memory. It is your responsibility to deallocate any memory you allocated for your custom agent.

Syntax

```
int SM_EXTERN Sm_AgentApi_UnInit (
   void** ppHandle
);
```

Parameter	**I/O**	**Description**
ppHandle	I	The address of a pointer to the handle for this API session. This is an opaque type.

Returns

- SM_AGENTAPI_SUCCESS. Uninitialization succeeded.
- SM_AGENTAPI_NOCONNECTION. Uninitialization was not done.

Example

See the function SmAgentExample::Sm_UnInit() in the example application smagentexample.cpp.

Sm_AgentApi_UpdateAttributes()

Call this function to update response attributes when the time-to-live (TTL) value has expired. When specifying request attributes, each attribute structure should be zero-initialized, then set the *lpszAttributeOid* fields in the Sm_AgentApi_Attribute_t structure. Use the same field to map the response attributes to requested attributes. Not all attributes may be updated.

This function will recalculate an active response that originally held no data. An active response attribute is considered valid if its return value has a non-zero length or its TTL value is set.

Syntax

```
int SM_EXTERN Sm_AgentApi_UpdateAttributes (
   const void*                        pHandle,
   const char*                        lpszClientIpAddr,
   const char*                        lpszTransactionId,
   const Sm_AgentApi_ResourceContext_t* pResourceContext,
   const Sm_AgentApi_Realm_t*         pRealm,
   const Sm_AgentApi_Session_t*       pSession,
   long                               nNumReqAttributes,
   Sm_AgentApi_Attribute_t*           pReqAttributes,
   long*                              pRespNumAttributes,
   Sm_AgentApi_Attribute_t**          ppRespAttributes
);
```

Parameter	I/O	Description
pHandle	I	Agent API session handle returned in parameter *ppHandle* of Sm_AgentApi_Init().

Parameter	I/O	Description
lpszClientIpAddr	I	The IP address of the client that the user is logging from. This is an optional parameter. If the client IP begins with a star (*), the Policy Server logs the IP address but does not validate it against a session specification.
lpszTransactionId	I	The ID that the agent uses to associate application activity with security activity. The Policy Server logs this ID. This is an optional parameter.
pResourceContext	I	A resource definition structure.
pRealm	O	A realm definition structure.
pSession	I	A session definition structure.
nNumReqAttributes	I	The number of attributes in *pReqAttributes*.
pReqAttributes	I	An array of requested attribute definition structures.
pRespNumAttributes	O	The number of attributes in *ppRespAttributes*.
ppRespAttributes	O	An array of response attribute definition structures.

Returns

- SM_AGENTAPI_YES. The operation succeeded.
- SM_AGENTAPI_NO. The operation failed.
- SM_AGENTAPI_NOCONNECTION. The initialization was not done.
- SM_AGENTAPI_FAILURE. The server could not be reached.
- SM_AGENTAPI_TIMEOUT. The function timed out.

Chapter 3: Configuring Custom Agent Types

This section contains the following topics:

Custom Agent Type Overview

Using the Agent API, you can develop custom agents tailored to meet the specific needs of your environment. After you have developed the agent, you must configure a new Agent Type for the agent in the Administrative UI.

The Agent Type defines the behavior of an agent. For example, if you developed a custom FTP Agent, you would then need to define an Agent Type for the FTP Agent in the Administrative UI.

Agent Type Worksheet

Before you begin the process of defining a custom Agent Type, you should have a clear understanding of the environment your agent will support. Answer the following questions before following the steps for configuring and agent type.

- Which protocol is going to be protected by this agent—for example, HTTP?
- What are the actions or commands that you want to protect on these resources? Based on the protocol, list the actions you will allow. For example, for an FTP site, you may want to protect GET and PUT.
- What responses do you want the agent to receive? What data types will you want to pass in your responses: string, number, IP address?
  - A response can be a name-value pair. For example: time-restriction=yes
  - It can also be a string (personalized greeting) that you want to display to your user—for example, Hello John Smith, where name=John Smith.

After you have answered these questions, you can begin configuring your custom Agent Type.

Configure an Agent Type

To configure an Agent Type

1. Log into the Administrative UI.
2. Click Infrastructure, Agent Type, Create Agent Type.

   The SiteMinder Create Agent Type dialog opens.
3. Enter the name of the new Agent Type in the Name field.
4. Enter a brief description in the Description field.
5. If the new Agent Type is a RADIUS Agent, select the RADIUS Device check box.

   If you select the RADIUS Device check box, the fields in the Agent Type Definition Tab change to the following:

   - Vendor Specific Offset—Enter the Offset value provided by the RADIUS vendor. This is not a required field.
   - IETF Vendor ID—Enter an integer value assigned to a Vendor by the Internet Assigned Numbers Authority (IANA). This is not a required field. Do not use a value of 2552, because this is reserved for the SiteMinder Agents.
6. If the new Agent Type is not a RADIUS Agent, click Create in the Actions box to specify the actions you want to allow on the resources protected by this agent.
7. Enter a text string that describes an action the new Agent Type will recognize, such as authenticate, then click OK.

   The Agent Action dialog closes, and the new action is added to the list of actions in the Agent Type dialog.
8. To configure attributes for this Agent type, click Create Agent Type Attribute.
9. Click Submit.

Note: The Resource Type is URL. This type of resource is processed in a string-like manner: resource matching can be done based on wild cards and regular expressions.

Agent Type Attributes

The following Agent Type Attributes can be entered when configuring an Agent Type:

- Properties
- Values

Properties

The fields in the Properties group box enable you to define the behavior of a response attribute that can be used when constructing SiteMinder responses.

- Name—Name of the Agent Type Property.
- Description—Brief description of the Agent Type Property
- Data Type—One of the following data types for the value that the Policy Server returns to the agent:
  - String—A string value.
  - Number—A four-byte numeric value in a network byte order.
  - IP Address—A four-byte numeric value in a network byte order that can be interpreted as an IP address.
  - Encrypted String—A string encrypted using RADIUS-like encryption. An encrypted string is used for RADIUS agents.
- Identifier—This is a numeric ID assigned to the attribute, ranging from 1 to 255. For custom Agents, available values are 224-255 and 1-150.

When assigning Response Behavior, consider the following:

- What is the purpose of this response type: authentication or authorization?
- What types of response behavior will this response require: success or failure?
- How many instances of this type of an attribute will you allow in one response: *Zero or One* or *Zero or Many*?
- If you are setting up an authorization response type, you can use one of the following two attribute selections: Access Accept or Access Reject.
- If you are setting up an authentication Response Type, you can use any one of the three displayed attribute selections: Access Accept, Access Reject, or Access Challenge.

Values

This group box only appears when you select an attribute Data Type to be a Number in the Agent Type Attributes. You can assign a Symbolic Name and a Numeric Value to a response of such data type. This makes it easier to remember what each numeric value stands for when you create responses using this response type.

Configure the Agent Type Attributes Properties and Values

To configure Agent Type Attributes

1. Click Create Agent Type Attributes in the Agent Type Attributes group box.
2. Enter a name and description for the attribute In the General group box.
3. Enter the data type and identifier in the Properties group box.
4. If the data type is a number, the Values group box appears. Enter the symbolic name and numerical value.
5. Enter the appropriate response behaviors.
6. Click Submit to save your selections.

Modify an Agent Type

To edit Agent Type Properties

1. Log into the Administrative UI.
2. Click Infrastructure, Agent Types, Modify Agent Types.
3. Enter the name of the Agent Type you want to modify and click Search, or just click Search to view a complete list of Agent Types.
4. Select the Agent Type.

   The Modify Agent Type dialog appears.
5. Click the Edit button to the left of the the Action or Attribute you want to modify. You can also delete an action or attribute (Delete button on the right of the listing).
6. Make the changes you require.
7. Click Submit to save the changes.
   The Modify Agent Type dialog appears.
8. Click Submit again to save the changes for the Agent Type.

Chapter 4: Policy Management API Guidance

This section contains the following topics:

Policy Management API Overview

The Policy Management API lets you manipulate policy objects within a SiteMinder installation. Using the Policy Management API, you can perform most of the data manipulations that are provided by the Administrative UI. You can also develop your own custom interface to SiteMinder.

Note: Before you work with the Policy Management API, be sure that you are familiar with SiteMinder concepts.

Policy Management Setup

To use the Policy Management API

1. Install the Policy Server and the SiteMinder Software Development Kit on the same machine. In the Windows environment, the Policy Server is required for running Policy Management applications. In the UNIX environment, the Policy Server is required for both building and running Policy Management applications.

   Note: You can build your Policy Management application without running the Policy Server services.

2. Use the Policy Server Management Console to configure the Policy Server so that it points to the policy store you want to access.

3. Run your Policy Management application on the machine where the Policy Server is installed and that has been configured to point to the policy store. The policy store can be on a different machine than the Policy Server.

To run your Policy Management application**,** you need the following files:

Windows platforms: SmPolicyApi45.dll

UNIX platforms: libsmpolicyapi45.so and libsmutilities.so, in the following location:

<siteminder_install_location>\Netegrity\SiteMinder\lib

Refer to the sample makefile before executing a UNIX build.

To build your policy application, include SmPolicyAPI45.h and link to the required shared libraries.

- **Windows platforms:** Link to SmPolicyAPI45.lib, located in <install_path>\sdk\lib\win32\
- **UNIX platforms:** Link to the libraries libsmpolicyapi45.so and libsmutilities.so, located as described in the previous section.

Note: Before you build policy management applications for UNIX, you must install the SiteMinder SDK on the same machine as the Policy Server.

Object Retrieval Functions

These functions retrieve information about an object from the SiteMinder policy store.

If the return code indicates success, a linked list of objects that match the request is returned. In most cases, the API returns a single item that matches the unique object identifier. If a matching object is not found, the return code indicates failure and the returned linked list pointer points to NULL.

Object retrieval functions are prefixed with Sm_PolicyApi_Get. To find the function that retrieves information for a particular object, look in the table of functions for that object.

More Information:

Functions by Category in the Policy Management API (see page 243)

Object Creation Functions

To create a SiteMinder object, you must fill in the appropriate data structure and call the appropriate function with a properly initialized handle. If the call is successful:

- The function returns Sm_PolicyApi_Success.
- The object is added to the SiteMinder policy store.
- The *pszOid* field in the corresponding object structure is set to the object identifier of the object.

Object creation functions are prefixed with either Sm_PolicyApi_Add or Sm_PolicyApi_Create.

More Information:

Functions by Category in the Policy Management API (see page 243)

Object Deletion Functions

These functions delete objects from the SiteMinder policy store. Only one object at a time can be deleted.

Object deletion functions are prefixed with Sm_PolicyApi_Delete or Sm_PolicyApi_Remove.

More Information:

Functions by Category in the Policy Management API (see page 243)

Object Associations

Some objects can be associated with or disassociated from one another-for example, Sm_PolicyApi_AddAdminToDomain() adds an administrator object to a domain, and Sm_PolicyApi_RemoveAdminFromDomain() removes an administrator object from a domain.

An "add-to" operation requires that both objects exist prior to the call and have an established association. After a "remove-from" operation, both objects still exist, but they are no longer associated with one other.

When you're looking for a function that associates or disassociates two objects, look in the category of the method that you are adding or removing. For example, the functions Sm_PolicyApi_AddAdminToDomain() and Sm_PolicyApi_RemoveAdminFromDomain() are both found in Administrator Functions.

Object Identifiers

With the introduction of nested realms, the unique identification of an object can no longer rely on a realm name. When a SiteMinder object is created, a unique *object identifier* (OID) is written in the *pszOid* field of the object's defining structure.

These functions do not return SiteMinder objects. Instead, they return an array of string pointers that contain the OIDs of SiteMinder objects. You pass in OIDs to SiteMinder Object Retrieval Functions (Sm_PolicyApi_Get...) to specify objects to retrieve.

The functions that return arrays of OIDs are:

- Sm_PolicyApi_GetDomainObjects()
- Sm_PolicyApi_GetGlobalObjects()
- Sm_PolicyApi_GetUserDirSearchOrder()

Free the memory allocated by this group of functions by calling Sm_PolicyApi_FreeMemoryEx().

Directory Search Order Functions

The following functions help you retrieve and set the search order of user directories:

- Sm_PolicyApi_GetUserDirSearchOrder() retrieves the *object identifiers* (OIDs) of user directory objects associated with the specified domain.
- Sm_PolicyApi_SetUserDirSearchOrder() sets the search order of the user directories in a domain. The ordered list of OIDs is specified in the *pszArray* string array. The user directories in this array must match in OID and number (but not order) the list of user directory OIDs that were retrieved by a call to Sm_PolicyApi_GetUserDirSearchOrder().

Performance Enhancement

By performing either of the following actions, a custom Policy Management application can reduce the time it takes to update policy store objects:

- Omit the Sm_PolicyApi_InitFlags_PreLoadCache flag in the call to Sm_PolicyApi_Init().
- Introduce a very small time delay in the custom application to ensure adequate time for cache processing before exiting.

Memory, Cache, and Agent Key Management

The following functions free memory allocated by the Policy Management API:

- Sm_PolicyApi_FreeMemory() and Sm_PolicyApi_FreeMemoryEx() free any of the linked list pointers that are returned by the API.
- Sm_AgentApi_FreeServers() frees an array of server structures allocated by Sm_AgentApi_GetConfig().
- Sm_PolicyApi_FreeString() frees memory allocated for a single string (for example, pszUseMsg and pszErrMsg strings).
- Sm_PolicyApi_FreeStringArray() frees arrays of returned strings.

Another management command, Sm_PolicyApi_ManagementCommand(), performs cache and agent encryption key management, such as:

- Flushing all caches
- Flushing users cache
- Flushing realms from the resource cache
- Changing dynamic keys
- Changing persistent keys

The type of management operation you want to perform is determined by the management command you pass to Sm_PolicyApi_ManagementCommand().

Object Scope

SiteMinder objects can be classified according to scope:

- *Domain objects are* visible only within the domain. They cannot be shared between domains.
- *Global object*s are visible across all domains. Global objects are sometimes called system objects.

The scope of SiteMinder objects is as follows:

- Global objects include:
  - Administrators
  - Agent types
  - Agents and agent groups
  - Authentication schemes
  - Authentication/authorization maps
  - Certificate maps

  - Domains
  - ODBC query schemes
  - Password policies
  - Policies
  - Registration schemes
  - Responses
  - Rules
  - User directories
- Domain objects include:
  - Policies
  - Realms
  - Responses and response groups
  - Response attributes
  - Rules and rule groups
  - User policies

Federation API

The federation APIs support the manipulation of policy store data related to Affiliate Domain objects, which can include Affiliates, Service Providers, and Resource Partners.

The CA SiteMinder® federation products support SAML 1.x, SAML 2.0, and WS-Federation profiles. Partners have the ability to exchange user profile information in a secure manner.

More Information:

Federation Functions (see page 249)
SAML 2.0 Configuration Functions (see page 257)

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 Java APIs in the CA SiteMinder® 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 Affiliations

A SAML 2.0 affiliation consists of Service Providers and Identity Providers that have a shared Name ID namespace. Identity Providers also share the user disambiguation properties across the affiliation.

A SAML 2.0 affiliation can have multiple Service Providers and Identity Providers. However, a Service Provider or Identity Provider can belong to no more than one SAML 2.0 affiliation.

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:

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 occurs transparently 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 Attribute Authority

SiteMinder supports authorization that uses the values of predetermined user attributes from a remote site 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 preferred 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 a customized rate sheet.

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.

In the Policy Management API, the Sm_PolicyApi_SAMLRequesterAttr_t structure defines an attribute that can be requested by the SAML Requester. It specifies the actual name of the attribute known by the Attribute Authority, as well as a local name used in Federation attribute variables. The local name maps to a variable defined in the SAML 2.0 authentication scheme.

More Information:

Sm_PolicyApi_SAMLRequesterAttr_t (see page 181)
SAML 2.0 Attribute Authority Functions (see page 257)

SAML 2.0 Indexed Endpoints

When configuring single sign-on at the Identity Provider, you can configure more than one endpoint for the Assertion Consumer Service, the service that enables a Service Provider to consume a SAML assertion. Each endpoint you configure 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 a part of a Service Provider 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 (Artifact or POST).

In the Policy Management API, you can, for example, add a new Assertion Consumer Service to the Service Provider programmatically by calling the Sm_PolicyApi_AddAssertionConsumerServiceToSAMLSP() function as follows:

```
iSmApiRetCode = Sm_PolicyApi_AddAssertionConsumerServiceToSAMLSP (
    pSmApiSessionHandle,
    &structSAMLSPACS2,
    pszOid);
```

Parameters:

- pSmAPiSessionHandle is a pointer to a structure that holds information about the administrative session and the client session.
- pstructSAMLSPACS2 is a pointer to Sm_PolicyApi_SAMLSPAssertionConsumerService_t. This structure specifes the index, binding, and Assertion Consumer Service URL.
- pszOid is a pointer to a string containng the OID of the Service Provider.

In addition, the API includes a function to remove an Assertion Consumer Service and a function to retrieve all the Assertion Consumer Services defined in the Service Provider object. The C Policy Management API sample program, smpolicyapiexample.cpp, shows how these functions are implemented.

More Information:

SAML 2.0 Indexed Endpoint Functions (see page 259)

Sample Application for Affiliates

The C sample program smpolicyapi has been augmented for the affiliate functionality.

To run the affiliate portions

1. Install the SiteMinder Option Pack on the Policy Server
2. Define the Affiliate Policy Store objects in smpolicy.smdif and import the objects to the Policy Store.
3. Install the sample. This creates a sample user directory, which is used by the affiliate sample.

Three options have been added to the smpolicyapi program.

- Do you wish to install Affiliate Sample Api? (Y/N):

  Answering **Y** causes smpolicyapi to:

  - Add an affiliate domain
  - Add the sample user directory to the affiliate domain
  - Add an affiliate to the affiliate domain
  - Add users from the sample user directory to the affiliate
  - Add an affiliate attribute to the affiliate
- Do you wish to traverse affiliate domain? (Y/N):

  Answering Y causes smpolicyapi to:

  - Get the affiliate domain and print it.
  - Get all user dir OIDs in the sample affiliate domain and print them
  - Get all affiliate OIDs in the sample affiliate domain and print them
  - Get all affiliates in the sample affiliate domain and print their properties
  - Get all users in Affiliate and print them
  - Get all attributes in Affiliate and print them
- Do you wish to uninstall Affiliate Sample Api? (Y/N):

  Answering **Y** causes smpolicyapi to delete the affiliate domain.

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 Signout in the WS-Federation environment are implmented 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.

The C Policy Management API sample program, smpolicyapiexample.cpp, includes examples of how to define, list, and delete a Resource Partner, as well as define, list, and delete a WS-Federation authentication scheme.

More Information:

Sm_PolicyApi_WSFEDProviderProp_t (see page 204)
Sm_PolicyApi_WSFEDResourcePartner_t (see page 213)
WS-Federation Functions (see page 264)

Policy Management API Data Structures

Each data structure represents an entity in the SiteMinder policy store. The structures have a common format:

- The first field in each structure denotes the type of the structure, so that it can be freed by the correct Policy Management API memory function. When the structure is used as an input parameter, the first field is ignored by the Policy Management API. When the structure is used for a return, the first field is set properly by the Policy Management API.
- The last field of each structure is a pointer to the next item in a returned linked list. On input, this field is ignored, since the Policy Management API does not accept linked lists as input parameters. (If you supply a linked list as an input parameter, the function examines only the first item in the list.)

All the character arrays within the following structures are sized according to BFSIZE, which is set to 1024 characters. A 24-character TIMESIZE buffer is used for policy time restrictions.

Sm_PolicyApi_Admin_t

Defines a SiteMinder Administrator object.

Syntax

```
typedef struct Sm_PolicyApi_Admin_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                    /* Required */
   char pszDesc[BFSIZE];
   char pszPassword[BFSIZE];
   char pszUserDirOid[BFSIZE];
   char pszSchemeOid[BFSIZE];
   Sm_PolicyApi_AdminRights_t nRights;      /* Required */
   struct Sm_PolicyApi_Admin_s* next;
} Sm_PolicyApi_Admin_t;
```

Field	Description
iStructId	Administrator data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the administrator object.
pszName	Name of the administrator.

Field	Description
pszDesc	Brief description of the administrator.
pszPassword	This is required if the *pszUserDirOid* is not specified. The SiteMinder stores the password in its directory.
pszUserDirOid	Object identifier of the user directory if the administrator is stored in an external directory.
pszSchemeOid	Object identifier of the authentication scheme to use to authenticate the administrator. This is required if the *pszUserDirOid* is specified.
nRights	Rights of the administrator, as specified in Sm_PolicyApi_AdminRights_t.
next	Pointer to the next administrator structure.

Sm_PolicyApi_Affiliate_t

Represents an affiliate object.

Syntax

```
typedef struct Sm_PolicyApi_Affiliate_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];
   char pszDesc[BFSIZE];
   char pszAffiliateDomainOid[BFSIZE];
   char pszPassword[BFSIZE];
   bool bIsEnabled;
   bool bAllowNotification;
   char pszAuthURL[BFSIZE];
   char pszAudience[BFSIZE];
   long nValidityDuration;
   bool bSharedSession;
   long nSyncInterval;
   long nSkewTime;
   long nStartTime;
   long nEndTime;
   unsigned char pszTimeGrid[TIMESIZE];
   Sm_PolicyApi_IPAddress_t *pIPAddress;
   struct Sm_PolicyApi_Affiliate_s* next;
   long nSAMLVersion;
   char pszAssertionPluginClass[BFSIZE];
   char pszAssertionPluginParameters[BFSIZE];
   Sm_PolicyApi_SAML_Profile_t SAMLProfile;
   char pszConsumerURL[BFSIZE];
} Sm_PolicyApi_Affiliate_t;
```

Field	Description
iStructId	Domain data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the affiliate object.
pszName	Name of the affiliate.
pszDesc	Brief description of the affiliate.
pszAffiliateDomain Oid	The object identifier of the affiliate domain object.

Field	Description
pszPassword	The password for the affiliate as a null terminated string.
bIsEnabled	Boolean indicating if the affiliate is enabled.
bAllowNotification	Boolean indicating if notifications are allowed for the affiliate.
pszAuthURL	The authentication URL for the affiliate.
pszAudience	A URI of the document that describes the terms and conditions of the agreement between the portal and the affiliate.
nValidityDuration	The amount of time, in seconds, that the assertion is valid.
bSharedSession	Boolean that enables the sharing of session information between the portal and the affiliate.
nSyncInterval	The frequency, in seconds, at which the affiliate contacts the portal to validate session status.
nSkewTime	The difference, in seconds, between the system clock time of the portal and the system clock time of the affiliate.
nStartTime	The time when the time restriction becomes effective. This value is stored in standard time_t format. Set *nStartTime* to 0 to start the time restriction immediately.
nEndTime	The time when the time restriction expires. This value is stored in standard time_t format. Set *nEndTime* to 0 to end the time restriction immediately.
pszTimeGrid	An array containing time restrictions for an entire week.
pIPAddress	IP address that user must use in order to gain access to the resources governed by the Policy.
next	Pointer to the next affiliate structure.
nSAMLVersion	One of the following SAML version constants: ■ SM_POLICY_API_SAML_1_0 ■ SM_POLICY_API_SAML_1_1 The SAML version has effect only if the Policy Management API's session version is at least SM_POLICY_API_VERSION_6_0_1.

Field	Description
pszAssertion PluginClass	The fully qualified class name of a custom assertion generator plug-in. The plug-in lets you customize the default assertion that SiteMinder generates for an affiliate. A plug-in class and parameter string are supported only if the Policy Management API's session version is at least SM_POLICY_API_VERSION_6_0_2. Custom assertion generator plug-ins are implemented with the Java SDK.
pszAssertion PluginParameters	The parameter string to pass to a custom assertion generator plug-in.
SAMLProfile	The type of profile used to send and receive SAML assertions. Defined in Sm_PolicyApi_SAML_Profile_t. Valid profiles: ■ Sm_PolicyApi_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. ■ Sm_PolicyApi_SAML_Profile_POST. The generated SAML assertion is POSTed to the URL specified in *pszConsumerURL*. This profile is supported only if the Policy Management API's session version is at least SM_POLICY_API_VERSION_6_0_2.
pszConsumerURL	With a SAML POST profile, this field 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.

Sm_PolicyApi_AffiliateAttr_t

Represents affiliate attributes. Used with affiliate methods to manipulate affiliate attributes.

Syntax

```
typedef struct Sm_PolicyApi_AffiliateAttr_s
{
   int iStructId;
   Sm_PolicyApi_AffiliateAttrType_t nAttrType;
   char pszValue[BFSIZE];
   struct Sm_PolicyApi_AffiliateAttr_s* next;
} Sm_PolicyApi_AffiliateAttr_t;
```

Field	Description
iStructId	Domain data structure ID, defined in Sm_PolicyApi_Structs_t.
nAttrType	An affiliate attribute type from Sm_PolicyApi_AffiliateAttrType_t.
pszValue	An affiliate attribute specification. The affiliate attribute's name and value, in one of these formats: ■ 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"
next	Pointer to the next Affiliate Attribute structure.

Sm_PolicyApi_AffiliateDomain_t

Represents an affiliate domain.

Syntax

```
typedef struct Sm_PolicyApi_Domain_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];
   char pszDesc[BFSIZE];
   struct Sm_PolicyApi_Domain_s*   next;
} Sm_PolicyApi_Domain_t, Sm_PolicyApi_AffiliateDomain_t;
```

Field	Description
iStructId	Affiliate Domain data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the Affiliate Domain.
pszName	Name of the Affiliate Domain.
pszDesc	Brief description of the Affiliate Domain.
next	Pointer to the next Affiliate Domain structure.

Sm_PolicyApi_Agent_t

Defines a SiteMinder Agent object.

Syntax

```
typedef struct Sm_PolicyApi_Agent_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                    /* Required */
   char pszDesc[BFSIZE];
   char pszIpAddr[BFSIZE];
   char pszSecret[BFSIZE];
   char pszAgentTypeOid[BFSIZE];            /* Required */
   int nRealmHintAttrId;                    /* Required */
   struct Sm_PolicyApi_Agent_s* next;
} Sm_PolicyApi_Agent_t;
```

Field	Description
iStructId	Agent data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the agent object.
pszName	Name of the agent. Agent names are converted to lower case when written to the policy store.
pszDesc	Brief description of the agent.
pszIpAddr	IP address of the server on which the agent resides.
pszSecret	A shared secret known to two parties for the purpose of establishing secure data exchange. If a shared secret is provided, the agent is considered to be a SiteMinder version 4.x agent. If a shared secret is not provided, the agent is considered to be a SiteMinder version 5.x or 6.0 agent.
pszAgentTypeOid	Type of agent.
nRealmHintAttrId	The hint attribute is a RADIUS attribute that is sent by the RADIUS client device.
next	Pointer to the next agent structure.

Sm_PolicyApi_AgentConfig_t

Defines an agent configuration object.

Syntax

```
typedef struct Sm_PolicyApi_AgentConfig_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                    /* Required */
   char pszDesc[BFSIZE];
   struct Sm_PolicyApi_AgentConfig_s* next;
} Sm_PolicyApi_AgentConfig_t;
```

Field	**Description**
iStructId	Data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the agent configuration object.
pszName	Name of the agent configuration.
pszDesc	Brief description of the agent configuration.
next	Pointer to the next agent configuration object structure.

Sm_PolicyApi_AgentType_t

Defines a SiteMinder Agent Type object.

Syntax

```
typedef struct Sm_PolicyApi_AgentType_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];
   char pszDesc[BFSIZE];
   int nRfcId;
   int nAgentType;
   int nResourceType;
   int nAgentTypeSpecific;
   struct Sm_PolicyApi_AgentType_s* next;
} Sm_PolicyApi_AgentType_t;
```

Field	Description
iStructId	Agent Type data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the Agent Type object.
pszName	Name of the Agent Type object.
pszDesc	Brief description of the Agent Type object.
nRfcId	The IETF Vendor ID, assigned by the Internet Assigned Numbers Authority (IANA). (Applies to RADIUS device only.)
nAgentType	Agent type: RADIUS (value=0) or Web Agent (value=1).
nResourceType	The resource type that the agent will protect. One of the following values: ■ 0 = None ■ 1 = URL ■ 2 = IpAddr ■ 3 = IpAddrRange ■ 4 = AgentAuth ■ 5 = Radius Authentication

Field	Description
nAgentTypeSpecific	The vendor-specific offset provided by the RADIUS vendor. (Applies to RADIUS device only.) Not a required field.
next	Pointer to the next Agent Type structure.

Sm_PolicyApi_AgentTypeAttr_t

Defines a SiteMinder Agent Type Attribute object.

Syntax

```
typedef struct Sm_PolicyApi_AgentTypeAttr_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];
   char pszDesc[BFSIZE];
   char pszAgentTypeOid[BFSIZE];
   struct Sm_PolicyApi_AgentTypeAttr_s* next;
} Sm_PolicyApi_AgentTypeAttr_t;
```

Field	Description
iStructId	Agent Type Attribute data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the Agent Type Attribute object.
pszName	Name of the Agent Type Attribute object.
pszDesc	Brief description of the Agent Type Attribute object.
pszAgentTypeOid	The object identifier of the Agent Type.
next	Pointer to the next Agent Type Attribute structure.

Sm_PolicyApi_Association_t

Defines a configuration parameter name and its associated value for an agent configuration object.

Syntax

```
typedef struct Sm_PolicyApi_Association_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                   /* Required */
   char* pszValue;
   int iFlags;
   struct Sm_PolicyApi_Association_s* next;
} Sm_PolicyApi_Association_t;
```

Field	Description
iStructId	Data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the configuration parameter association object.
pszName	The name of the configuration parameter.
pszValue	The value of the configuration parameter.
iFlags	If 1, the name/value pair is stored in encrypted format. If 0, storage is in plain text.
next	Pointer to the next configuration parameter association structure.

Sm_PolicyApi_AuthAzMap_t

Defines a SiteMinder authentication and authorization mapping object.

Syntax

```
typedef struct Sm_PolicyApi_AuthAzMap_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszAuthDirOid[BFSIZE];              /* Required */
   char pszAuthDirName[BFSIZE];
   char pszAzDirOid[BFSIZE];                /* Required */
   char pszAzDirName[BFSIZE];
   Sm_PolicyApi_AuthAzMapType_t nMapType;   /* Required */
   struct Sm_PolicyApi_AuthAzMap_s *next;
} Sm_PolicyApi_AuthAzMap_t;
```

Field	Description
iStructId	Authentication and authorization mapping object data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the authentication and authorization mapping object.
pszAuthDirOid	The object identifier for the Authentication Directory.
pszAuthDirName	The name of the Authentication Directory.
pszAzDirOid	The object identifier for the Authorization Directory.
pszAzDirName	The name of the Authorization Directory.
nMapType	The type of mapping between an authentication directory and an authorization directory. The mapping can be based on a DN, a universal identifier, or on an attribute in the directory.
next	Pointer to the next authentication and authorization mapping structure.

Sm_PolicyApi_CertMap_t

Defines a SiteMinder certification mapping object.

Syntax

```
typedef struct Sm_PolicyApi_CertMap_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszDesc[BFSIZE];
   char pszUserDirOid[BFSIZE];              /* Required */
   char pszIssuerDN[BFSIZE];                /* Required */
   char pszCaDN[BFSIZE];
   char pszMapAttr[BFSIZE];
   Sm_PolicyApi_CertMapFlags_t nFlags;      /* Required */
   Sm_PolicyApi_CertMapAttrType_t nAttrType;
   Sm_PolicyApi_DirType_t nDirType;         /* Required */
   struct Sm_PolicyApi_CertMap_s *next;
} Sm_PolicyApi_CertMap_t;
```

Field	Description
iStructId	Certificate mapping data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the certificate mapping object.
pszDesc	Brief description of the certificate mapping object.
pszUserDirOid	Object identifier of the user directory.
pszIssuerDN	DN of the server certificate.
pszCaDN	DN of the issuing Certificate Authority.

Field	Description
pszMapAttr	You can perform single attribute mapping or custom mapping: ■ Single Attribute Mapping The format of the attribute is: %{<attribute name>} <*attribute name*> is the name of the attribute, matching a single attribute from the subject DN of a user's certificate to a single attribute stored in the user directory to verify the user's identity. ■ Custom Mapping Using custom mapping expressions for complex multiple attribute mapping, you can specify multiple user attributes that should be extracted from a user DN to establish a certificate mapping. The syntax for a custom mapping expression is a parsing specification designed to enable full mapping flexibility. It indicates which information to take from the certificate and where it should be applied to in the user directory. The basic syntax is: UserAttribute=%{CertificateAttribute}, UserAttribute2=%{CertificateAttribute}
nFlags	Set one or more of these flags: ■ Sm_PolicyApi_CertMapFlags_CertRequired ■ Sm_PolicyApi_CertMapFlags_UseDistributionPoints ■ Sm_PolicyApi_CertMapFlags_VerifySignature ■ Sm_PolicyApi_CertMapFlags_CRLCheck ■ Sm_PolicyApi_CertMapFlags_Cache

Field	Description
nAttrType	This enumeration specifies how the X.509 client certificate maps to the user information in the authentication directory. ■ Sm_PolicyApi_CertMapAttrType_Single = 1 Specify single attribute to make the Policy Server match a single attribute from the subject DN of a user's certificate to a single attribute stored in the user directory to verify the user's identity. ■ Sm_PolicyApi_CertMapAttrType_Custom = 2 You can specify a custom mapping expression to verify the user's identity. Specify this attribute type if the mapping is based on a custom expression. ■ Sm_PolicyApi_CertMapAttrType_Exact = 3 Specify exact attribute type to make the Policy Server match the user's entire DN from the certificate to the entire DN in the authentication directory.
nDirType	The type of directory used to authenticate users. One of these values: ■ Sm_PolicyApi_DirType_LDAP = 1 ■ Sm_PolicyApi_DirType_WinNT = 2 ■ Sm_PolicyApi_DirType_ODBC = 3
next	Pointer to the next registration scheme structure.

Sm_PolicyApi_Domain_t

Defines a SiteMinder Domain object.

Syntax

```
typedef struct Sm_PolicyApi_Domain_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                    /* Required */
   char pszDesc[BFSIZE];
   struct Sm_PolicyApi_Domain_s* next;
   Sm_PolicyApi_DomainFlags_t iFlags;
} Sm_PolicyApi_Domain_t;
```

Field	Description
iStructId	Domain data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the Domain object.
pszName	Name of the Domain.
pszDesc	Brief description of the Domain.
next	Pointer to the next domain structure.
iFlags	Flag to enable or disable global policies processing for the domain.

Sm_PolicyApi_Group_t

Defines a SiteMinder Group object. SiteMinder Groups are defined in the Sm_PolicyApi_Groups_t enumeration.

Syntax

```
typedef struct Sm_PolicyApi_Group_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                        /* Required */
   char pszDesc[BFSIZE];
   char pszAgentTypeOid[BFSIZE];                /* Required */
   struct Sm_PolicyApi_Group_s* next;
} Sm_PolicyApi_Group_t;
```

Field	Description
iStructId	Group data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the Group object.
pszName	Name of the Group.
pszDesc	Brief description of the Group.
pszpszAgentTypeOid	Object identifier of the agent.
next	Pointer to the next Group structure.

Sm_PolicyApi_HostConfig_t

Defines a host configuration object.

Syntax

```
typedef struct Sm_PolicyApi_HostConfig_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                    /* Required */
   char pszDesc[BFSIZE];
   char pszIPAdress[BFSIZE];
   bool bEnableFailOver;
   int iMaxSocketsPerPort;
   int iMinSocketsPerPort;
   int iNewSocketStep;
   int iRequestTimeout;
   struct Sm_PolicyApi_HostConfig_s* next;
   Sm_PolicyApi_Server_t* pServer;
   Sm_PolicyApi_Server_t* pCluster;
   long nFailoverThreshold;
} Sm_PolicyApi_HostConfig_t;
```

Field	Description
iStructId	Data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the host configuration object.
pszName	The name of the host configuration.
pszDesc	Brief description of the host configuration.
pszIPAddr	The IP address of the host configuration object.
bEnableFailOver	Specifies whether an agent and the Policy Server should communicate through failover or round-robin. The parameter is applicable to non-cluster servers specified in the *pServer* parameter.
iMaxSocketsPerPort	The maximum number of TCP/IP sockets that can be opened between an agent and a particular Policy Server process.
iMinSocketsPerPort	The minimum number of TCP/IP sockets that should be opened between an agent and a particular Policy Server process.

Field	Description
iNewSocketStep	The incremental number of TCP/IP sockets that should be opened between an agent and a particular Policy Server process when demand increases.
iRequestTimeout	The length of time in seconds that an agent will wait for a response from the Policy Server.
next	Pointer to the next host configuration structure.
pServer	A linked list of Sm_PolicyApi_Server_t structures that describe TCP/IP connectivity information for a particular Policy Server installation.
pCluster	Pointer to an array of Sm_PolicyApi_Server_t structures. The array specifies cluster servers.
nFailoverThreshold	Specifies the failover threshold percent. The parameter is applicable to the cluster servers specified in *pCluster*.

Sm_PolicyApi_IPAddress_t

Defines an IP address restriction for an object-for example, you can define IP address restrictions that must be met for a policy to fire.

You can specify a single host IP address, a range of IP addresses, a host name, or a subnet mask.

Syntax

```
typedef struct Sm_PolicyApi_IPAddress_s
{
   int iStructId;
   Sm_PolicyApi_IPAddressType_t iIPAddressType;
   unsigned long nIPAddress;
   unsigned long nEndIPAddress;
   unsigned long nSubnetMask;
   char pszHostName[BFSIZE];
   struct Sm_PolicyApi_IPAddress_s *next;
} Sm_PolicyApi_IPAddress_t;
```

Field	Description
iStructId	IP address structure ID, defined in Sm_PolicyApi_Structs_t.

Field	Description
iIPAddressType	Type of IP address, as enumerated in Sm_PolicyApi_IPAddressType_t.
nIPAddress	Starting IP address.
nEndIPAddress	Ending IP address.
nSubnetMask	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 *nSubnetMask* would be 25.
pszHostName	Host name of the machine that a user must be using in order for a policy to fire.
next	Pointer to next IP Address structure.

Sm_PolicyApi_ManagementCommand_t

Defines a management command. Management commands enable an agent to retrieve information from the Policy Server.

Syntax

```
typedef struct Sm_PolicyApi_ManagementCommand_s
{
   Sm_PolicyApi_ManagementCommands_t iCommand;
   char pszData[BFSIZE];
} Sm_PolicyApi_ManagementCommand_t;
```

Field	Description
iCommand	Management command, as specified in Management Commands.
pszData	Reserved.

Sm_PolicyApi_ODBCQueryScheme_t

Defines a SiteMinder ODBC Query Scheme object.

Syntax

```
typedef struct Sm_PolicyApi_ODBCQueryScheme_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                   /* Required */
   char pszDesc[BFSIZE];
   char pszQueryEnumerate[BFSIZE];         /* Required */
   char pszQueryGetObjInfo[BFSIZE];        /* Required */
   char pszQueryLookup[BFSIZE];            /* Required */
   char pszQueryInitUser[BFSIZE];          /* Required */
   char pszQueryAuthenticateUser[BFSIZE];  /* Required */
   char pszQueryGetUserProp[BFSIZE];       /* Required */
   char pszQuerySetUserProp[BFSIZE];       /* Required */
   char pszQueryGetUserProps[BFSIZE];      /* Required */
   char pszQueryLookupUser[BFSIZE];        /* Required */
   char pszQueryGetGroups[BFSIZE];         /* Required */
   char pszQueryIsGroupMember[BFSIZE];     /* Required */
   char pszQueryGetGroupProp[BFSIZE];      /* Required */
   char pszQuerySetGroupProp[BFSIZE];      /* Required */
   char pszQueryGetGroupProps[BFSIZE];     /* Required */
   char pszQueryLookupGroup [BFSIZE];      /* Required */
   char pszQuerySetPassword [BFSIZE];      /* Required */
   struct Sm_PolicyApi_ODBCQueryScheme_s* next;
} Sm_PolicyApi_ODBCQueryScheme_t;
```

Field	Description
iStructId	ODBC Query Scheme data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the ODBC Query Scheme object.
pszName	Name of the ODBC Query Scheme.
pszDesc	Brief description of the ODBC Query Scheme.
pszQueryEnumerate	Query that lists the names of user objects in the directory.
pszQueryGetObjInfo	Query that fetches the class of the object.

Field	Description
pszQueryLookup	Query that returns objects based on an attribute specified in a group table.
pszQueryInitUser	Query that determines if a user with a given name exists in the database.
pszQueryAuthenticateUser	Query that retrieves a password from a user.
pszQueryGetUserProp	Query that retrieves the value of a user property. The property must be one of the properties listed with *pszQueryGetUserProps*.
pszQuerySetUserProp	Query that sets the value of a user property. The property must be one of the properties listed in *pszQueryGetUserProps*.
pszQueryGetUserProps	Query that returns a comma-separated list of user attributes that reside in the same table as the user name.
pszQueryLookupUser	Query that retrieves a user name using an attribute of the user table.
pszQueryGetGroups	Query that retrieves the names of the groups of which the user is a member.
pszQueryIsGroupMember	Query that identifies the group membership of a particular user.
pszQueryGetGroupProp	Query that returns the value of a property defined in *pszQueryGetGroupProps*
pszQuerySetGroupProp	Query that sets the value of a group property. The property must be one of the properties listed in *pszQueryGetGroupProps*.
pszQueryGetGroupProps	Query that returns a comma-separated list of group attributes.
pszQueryLookupGroup	Query that retrieves a group name using an attribute of the group table.
pszQuerySetPassword	Query that changes a user password.
next	Pointer to the next ODBC query scheme structure.

Sm_PolicyApi_Oid_t

Used by the following functions to retrieve a SiteMinder object collection:

- Sm_PolicyApi_GetDomainObjects()
- Sm_PolicyApi_GetChildren()
- Sm_PolicyApi_GetGlobalObjects()
- Sm_PolicyApi_GetGroupOids().

Syntax

```
typedef struct Sm_PolicyApi_Oid_s
{
   int iStructId;
   int iObjectId;
   char pszOid[BFSIZE];
   struct Sm_PolicyApi_Oid_s *next;
} Sm_PolicyApi_Oid_t;
```

Field	Description
iStructId	OID data structure ID, defined in Sm_PolicyApi_Structs_t.
iObjectId	The object type identifier (enumerated in Sm_PolicyApi_Objects_t).
pszOid	The unique object identifier.
next	Pointer to the next OID structure.

Sm_PolicyApi_PasswordMsgField_t

Describes a password policy message field. A password policy message field contains information about an error that occurred during a validation attempt for a new password.

A password policy message field is associated with a password policy message identifier. This identifier is returned in the *nMsgId* parameter of Sm_PolicyApi_GetPasswordMsg().

This structure is returned in the *ppStructMsgField* parameter of Sm_PolicyApi_GetPasswordMsg().

Syntax

```
typedef struct Sm_PolicyApi_PasswordMsgField_s
{
   int iStructId;
   Sm_PolicyApi_PasswordMsgFieldId_t nId;
   Sm_PolicyApi_FieldType_t nType;
   char pszMsg[BFSIZE];
   int nValue;
   struct Sm_PolicyApi_PasswordMsgField_s* next;
} Sm_PolicyApi_PasswordMsgField_t;
```

Field	Description
iStructId	Password policy data structure ID, defined in Sm_PolicyApi_Structs_t.
nId	The password message field identifier. Message field identifiers are enumerated in Sm_PolicyApi_PasswordMsgFieldId_t.
nType	The data type of the message field (integer, string, or none) as enumerated in Sm_PolicyApi_FieldType_t. The value of *nType* determines whether *pszMsg* or *nValue* is populated.
pszMsg	Text that provides information about the error.
nValue	Integer that provides information about the error.
next	Pointer to the next password message field structure.

Sm_PolicyApi_PasswordPolicy_t

Defines a SiteMinder password policy object.

Syntax

```
typedef struct Sm_PolicyApi_PasswordPolicy_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                    /* Required */
   char pszDesc[BFSIZE];
   bool bEntireDir;                         /* Required */
   bool bIsEnabled;                         /* Required */
   char pszUserDirectoryOid[BFSIZE];        /* Required */
   char pszPath[BFSIZE];                    /* Required */
   char pszClass[BFSIZE];                   /* Required */
   int nResolution;                         /* Required */
   int nLoginMaxFailures;                   /* Required */
   int nLoginMaxInactivity;                 /* Required */
   int nLoginInactivityWarn;
   int nLoginDaysGrace;                     /* Required */
   char pszDictionaryName[BFSIZE];
   int nDictionaryPartial;                  /* Required */
   int nExpirationDelay;                    /* Required */
   int nReenablement;                       /* Required */
   int nPasswordBehavior;                   /* Required */
   char pszPasswordServicesRedirect[BFSIZE];
   int nPWMaxLength;                        /* Required */
   int nPWMinLength;                        /* Required */
   int nPWMaxRepeatingChar;                 /* Required */
   int nPWMinAlphaNum;                      /* Required */
   int nPWMinAlpha;                         /* Required */
   int nPWMinNonAlpha;                      /* Required */
   int nPWMinNonPrintable;                  /* Required */
   int nPWMinNumbers;                       /* Required */
   int nPWMinPunctuation;                   /* Required */
   int nPWReuseCount;                       /* Required */
   int nPWReuseDelay;                       /* Required */
   int nPWPercentDifferent;                 /* Required */
   int nPWPercentSequence;                  /* Required */
   int nPWSpecialsLength;                   /* Required */
   struct Sm_PolicyApi_PasswordPolicy_s* next;
   int nPriority;                           /* Required */
   int nPWMinLowerAlpha;                    /* Required */
   int nPWMinUpperAlpha;                    /* Required */
   int nReserved1;
   int nReserved2;
   int nReserved3;
   int nReserved4[BFSIZE];
} Sm_PolicyApi_PasswordPolicy_t;
```

Field	Description
iStructId	Password policy data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the password policy object.
pszName	Name of the password policy.
pszDesc	Brief description of the password policy object.
bEntireDir	Flag: Set to true to apply the password policy to the entire LDAP directory.
bIsEnabled	Flag: Set to true to enable the password policy.
pszUserDirectoryOid	Object identifier of the user directory to which the password policy will apply.
pszPath	If *bEntireDir* is set to false, use this field to specify the users to whom the password policy applies. For example, values you can specify include: ■ A complete user DN ■ A complete DN of an organization or group containing the users to whom the password policy applies ■ A search expression that represents the users to whom the password policy applies See *nResolution* for more information.
pszClass	If *bEntireDir* is set to false, use this field to specify the object class where the password policy applies. Specify the object class as understood by a particular user directory, such as organizationalUnit in LDAP or User in NT.

Field	Description
nResolution	A constant that shows how the password policy applies to the users specified in *pszPath*. For example: ■ If *pszPath* is the path to a group, the following constant indicates that the password policy applies to the members of the group: Sm_PolicyResolution_UserGroup ■ If *pszPath* is a search expression that searches for all groups containing a particular attribute, the following constant indicates that the password policy applies to the members of the matching groups: Sm_PolicyResolution_GroupProp Policy resolution constants are defined in Sm_PolicyResolution_t.
nLoginMaxFailures	Maximum number of failed login attempts a user can make before the user account is disabled.
nLoginMaxInactivity	Number of days of inactivity allowed before a user's password expires. The account is not disabled until the user tries to log in after the expiration.
nLoginInactivityWarn	Reserved.
nLoginDaysGrace	Number of days in advance to notify user that the password will expire.
pszDictionaryName	The location of a dictionary file that lists words that cannot be used in a password.
nDictionaryPartial	The minimum number of letters to qualify for dictionary checking. For example, set this field to 5 to avoid checking words of 4 or fewer characters. Set this field to 0 to reject only passwords that match a word in the dictionary exactly.
nExpirationDelay	Number of days of inactivity allowed before user account is disabled.

Field	Description
nReenablement	Specifies the number of minutes a user must wait before attempting to log in again or before the user's account is re-enabled. The condition that applies after the *nReenablement* time period is determined by the following flag (which is set through *nPasswordBehavior*): Sm_PasswordPolicyBehavior_ FullReenable
nPasswordBehavior	Bit mask flags expressing the behavior of the password policy, as defined in Sm_PasswordPolicyBehavior_t. You can also set recursive behavior for the password policy through the additional password behavior flag below. Use this flag to indicate that the password policy applies to the group specified in *pszPath* and to any groups nested within it: Sm_PolicyBehavior_Recursive_Yes All password behavior flags are defined in SmApi.h.
pszPasswordServices Redirect	The URL to which the user should be redirected when an invalid password is entered. This must be the URL of the Password Services CGI.
nPWMaxLength	The maximum length for user passwords.
nPWMinLength	Minimum length for user passwords.
nPWMaxRepeatingChar	Maximum number of identical characters that can appear consecutively in a password.
nPWMinAlphaNum	Minimum number of alphabetic or numeric characters (A-Z, a-z, or 0-9) that a password must contain. May be set in conjunction with *PWMinAlpha* or *PWMinNumbers*. For example, if *PWMinAlphaNum* and *PWMinNumbers* are set to 4, the password 1234 is valid.
nPWMinAlpha	Minimum number of alphabetic characters (A-Z, a-z) a password must contain.

Field	Description
nPWMinNonAlpha	Minimum number of non-alphanumeric characters a password must contain. These characters include punctuation marks and other symbols located on the keyboard, such as "@", "$", and "*."
nPWMinNonPrintable	Minimum number of non-printable characters that must be in a password. These characters cannot be displayed on a computer screen.
nPWMinNumbers	Minimum number of numeric characters (0-9) a password must contain.
nPWMinPunctuation	Minimum number of punctuation marks a password must contain. These characters include periods, commas, exclamation marks, slashes, hyphens, dashes, and other marks used for punctuation.
nPWReuseCount	Number of new passwords that must be used before an old one can be reused.
nPWReuseDelay	Number of days a user must wait before reusing a password.
nPWPercentDifferent	The percentage of characters a new password must contain that differ from characters in the previous password. If the value is set to 100, the new password may contain no characters that were in the previous password, unless *nPWPercentSequence* is set to 0.
nPWPercentSequence	Flag that indicates whether to ignore sequence (character position) when the different-from-previous-characters percentage is calculated. To ignore character position, set *nPWPercentSequence* to 1. This flag works in conjunction with *nPWPercentDifferent*. For examples of how this parameter works with *nPWPercentDifferent*, see Figure 31 on page 150.

Field	Description
nPWSpecialsLength	Specifies the minimum character sequence to check against the user's personal information. 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.
next	Pointer to the next registration scheme structure.
nPriority	Priority of password policy, when multiple password policies apply. The value can be any integer, including a negative one. The higher number has priority over the lower number.
nPWMinLowerAlpha	Minimum number of lowercase alphabetic characters.
nPWMinUpperAlpha	Minimum number of uppercase alphabetic characters.

Sm_PolicyApi_Policy_t

Defines a SiteMinder Policy object.

Syntax

```
typedef struct Sm_PolicyApi_Policy_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                    /* Required */
   char pszDesc[BFSIZE];
   char pszActiveExpr[BFSIZE];
   bool bIsEnabled;                         /* Required */
   char pszDomainOid[BFSIZE];               /* Required */
   long nStartTime;
   long nEndTime;
   unsigned char   pszTimeGrid[TIMESIZE];
   Sm_PolicyApi_IPAddress_t *pIPAddress;
   struct Sm_PolicyApi_Policy_s* next;
   char pszVariableExpr[BFSIZE]
   Sm_PolicyApi_Oid_t* pVariableList;
} Sm_PolicyApi_Policy_t;
```

Field	Description
iStructId	Policy data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the Policy object.
pszName	Name of the Policy.
pszDesc	Brief description of the Policy.

Field	Description
pszActiveExpr	Active expression associated with the policy. The expression is a string of variable definitions in the following format. The non-alphanumeric characters are required characters. For example: <@lib="LibName" func="FuncName" param="FuncParam"@> In the format example: ■ LibName is the name of the shared library that supports the Active Policy. ■ FuncName is the name of the actual function in the shared library that implements the Active Policy. ■ FuncParam is an optional list of parameters to be passed to the function in the shared library.
bIsEnabled	Flag to enable or disable the policy.
pszDomainOid	The object identifier of the domain. Required for domain-specific policy; ignored for global policy.
nStartTime	The time when the time restriction becomes effective. This value is stored in standard time_t format. Set *nStartTime* to 0 to start the time restriction immediately.
nEndTime	The time when the time restriction expires. This value is stored in standard time_t format. Set *nEndTime* to 0 to end the time restriction immediately.
pszTimeGrid	An array containing time restrictions for an entire week.
pIPAddress	IP address that user must use in order to gain access to the resources governed by the Policy.
next	Pointer to the next Policy structure.
pszVariableExpr	Unique object identifier that corresponds to a variable type.
pVariableList	Linked list of variable OIDs used by this expression.

Time Grid Array

The time grid array (used with the field *pszTimeGrid*) holds time restrictions for an entire week. The array contains a one-byte element for every hour of the day, starting with 12 AM. In every byte, the seven days of the week are represented, beginning with Sunday as the lowest order bit. Bits that are set enable the policy to fire. A zero bit prevents the policy from firing on that day during the associated hour.

Examples:

- To disable policy firing for the hour 12-1 AM on Saturdays and Sundays, the hexadecimal value for the entire grid is:
  `3E7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F`
- To fire the policy at all times, leave all bits set:
  `7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F7F`
- To restrict the policy from being fired all day Thursday, turn off the Thursday bit in all the hour elements:
  `6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F`
- To restrict the policy from being fired from 8 AM to 10AM on Tuesday, turn off the Tuesday bit in hours 8 AM and 9 AM:
  `7F7F7F7F7F7F7F7F7C7C7F7F7F7F7F7F7F7F7F7F7F7F7F7F`
- To fire between 8 AM and 8 PM on all days:
  `00000000000000007F7F7F7F7F7F7F7F7F7F7F7F00000000`

Sm_PolicyApi_PolicyLink_t

Defines a SiteMinder Policy Link object.

Syntax

```
typedef struct Sm_PolicyApi_PolicyLink_s

{
   int iStructId;
   char pszOid[BFSIZE];
   char pszRuleOid[BFSIZE];                  /* Required */
   char pszResponseOid[BFSIZE];
   struct Sm_PolicyApi_PolicyLink_s* next;
} Sm_PolicyApi_PolicyLink_t;
```

Field	**Description**
iStructId	Policy Link data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the Policy Link object.
pszRuleOid	Object identifier of the rule.
pszResponseOid	Object identifier of the response.
next	Pointer to the next Policy Link structure.

Sm_PolicyApi_Realm_t

Sm_PolicyApi_Realm_t type defines a SiteMinder Realm object.

Syntax

```
typedef struct Sm_PolicyApi_Realm_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                /* Required */
   char pszDesc[BFSIZE];
   char pszResourceFilter[BFSIZE];
   char pszAgentOid[BFSIZE];            /* Required */
   char pszSchemeOid[BFSIZE];           /* Required */
   char pszParentRealmOid[BFSIZE];      /* Required */
   char pszDomainOid[BFSIZE];           /* Required */
   char pszAzUserDirOid[BFSIZE];        /* Required */
   char pszRegSchemeOid[BFSIZE];        /* Required */
   bool bProcessAuthEvents;             /* Required */
   bool bProcessAzEvents;               /* Required */
   bool bProtectAll;                    /* Required */
   int nMaxTimeout;                     /* Required */
   int nIdleTimeout;                    /* Required */
   bool bSyncAudit;                     /* Required */
   struct Sm_PolicyApi_Realm_s* next;
} Sm_PolicyApi_Realm_t;
```

Field	Description
iStructId	Realm data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the Realm object.
pszName	Name of the realm.
pszDesc	Brief description of the realm.
pszResourceFilter	Path for the resource filter.
pszAgentOid	Object identifier of the agent or agent group associated with the realm.
pszSchemeOid	Object identifier of the authentication scheme associated with the realm.

Field	Description
pszParentRealmOid	Object identifier of the parent realm under which this realm will be added. If the realm being added is the top realm in the realm hierarchy, set this field to the domain OID. Otherwise, set it to the parent realm OID.
pszDomainOid	Object identifier of the domain.
bProcessAuthEvents	Boolean to trigger this rule in an event of authentication attempts. Authentication event processing affects performance. If no rules in the realm are triggered by authentication events, set this field to false.
bProcessAzEvents	Boolean to trigger this rule in an event of authorization attempts. Authorization event processing affects performance. If no rules in the realm are triggered by authorization events, set this field to false.
bProtectAll	Boolean to protect all the resources contained in the new realm.
nMaxTimeout	Maximum amount of time a user can access the protected resources in the realm before they must re-authenticate.
nIdleTimeout	Amount of time a user can remain authenticated for the protected resources in the realm without interacting with the resources before they must re-authenticate.
bSyncAudit	Boolean to enable synchronous auditing. When enabled, users cannot access resources within a realm until their activity has been successfully recorded in the audit logs of both the Policy Server and the Web Agent.
pszAzUserDirOid	OID of the directory against which users accessing resources in this realm will be authorized.
pszRegSchemeOid	OID of the registration scheme that will be used to register new users accessing resources in this realm.
next	Pointer to the next realm structure.

Sm_PolicyApi_RegistrationScheme_t

Defines a SiteMinder registration scheme object.

Syntax

```
typedef struct Sm_PolicyApi_RegistrationScheme_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                     /* Required */
   char pszDesc[BFSIZE];
   char pszUserDirOid[BFSIZE];               /* Required */
   char pszWelcomePageURL[BFSIZE];
   char pszTemplatePath[BFSIZE];
   bool bEnableLogging;                      /* Required */
   struct Sm_PolicyApi_RegistrationScheme_s* next;
} Sm_PolicyApi_RegistrationScheme_t;
```

Field	**Description**
iStructId	Registration scheme data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the registration scheme object.
pszName	Name of the registration scheme.
pszDesc	Brief description of the registration scheme.
pszUserDirOid	Object identifier of the LDAP user directory in which user information is stored.
pszWelcomePageURL	Location of the form used to welcome users who registered and who have been successfully authenticated.
pszTemplatePath	Location of a registration template.
bEnableLogging	Flag to indicate whether to log registration information. Set this flag to true to enable logging.
next	Pointer to the next registration scheme structure.

Sm_PolicyApi_RegularExpression_t

Defines a regular expression.

Syntax

```
typedef struct Sm_PolicyApi_RegularExpression_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];
   char ExpressionDef[BFSIZE];
   bool matchExpression;
   struct Sm_PolicyApi_RegularExpression_s *next;
} Sm_PolicyApi_RegularExpression_t;
```

Field	Description
iStructId	Regular expression structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the regular expression object.
pszName	Name of the regular expression.
ExpressionDef	Definition of the regular expression.
matchExpression	1 if password must match this expression. 0 if password must not match this expression.
next	Pointer to the next regular expression structure.

Sm_PolicyApi_Response_t

Defines a SiteMinder Response object.

Syntax

```
typedef struct Sm_PolicyApi_Response_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                      /* Required */
   char pszDesc[BFSIZE];
   char pszAgentTypeOid[BFSIZE];              /* Required */
   char pszDomainOid[BFSIZE];                 /* Required */
   struct Sm_PolicyApi_Response_s* next;
} Sm_PolicyApi_Response_t;
```

Field	Description
iStructId	Response data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the Response object.
pszName	Name of the Response.
pszDesc	Brief description of the Response.
pszAgentTypeOid	The object identifier of the agent type.
pszDomainOid	The object identifier of the domain. Required for a domain-specific response; ignored for a global response.
next	Pointer to the next Response structure.

Sm_PolicyApi_ResponseAttr_t

Defines a SiteMinder Response attribute object.

Syntax

```
typedef struct Sm_PolicyApi_ResponseAttr_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszAgentTypeAttrOid [BFSIZE];        /* Required */
   char pszValue[BFSIZE];
   int iTTL;                                 /* Required */
   int iFlags;
   struct Sm_PolicyApi_ResponseAttr_s* next;
   char pszActiveExpr[BFSIZE];
   Sm_PolicyApi_Oid_t* pVariableList;
} Sm_PolicyApi_ResponseAttr_t;
```

Field	Description
iStructId	Response Attribute data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the Response Attribute object.
pszAgentTypeAttrOid	The object identifier of the agent type attribute.
pszValue	A response attribute type, as described in Response Attribute Types.
iTTL	Amount of time (in seconds) that can elapse before the value of the response attribute is recalculated.
iFlags	Reserved.
next	Pointer to the next Response Attribute structure.
pszActiveExpr	Active expression associated with the response attribute. For information, see the bullet "Active Response" in the section Response Attribute Types.
pVariableList	Linked list of variable OIDs used by the active expression.

Response Attribute Types

The field *pszValue* contains one of the following response attribute types:

Note: The non-alphanumeric characters in the formats below are required characters.

- **Static**. A string that is part of a SiteMinder response. The string has the following format:

  ```
  variable-name=variable-value
  ```

  In the format example:

  - *variable-name* is the name for the name/value pair that this response attribute will return to the Web Agent.
  - *variable-value* is the static text that will be returned as the second half of the name/value pair.

- **User Attribute**. A string containing profile information from a user's entry in a user directory. The string has the following format:

  ```
  User-Attr-variable-name=<%userattr="user-attr-name"%>
  ```

  In the format example:

  - *User-Attr-variable-name* is the name for the name/value pair that this response attribute will return to the Web Agent.
  - *user-attr-name* is a user attribute that can be retrieved from an LDAP, WinNT, or ODBC user directory.

- **DN Attribute**. A string containing profile information from a directory object in an LDAP or ODBC user directory. The string has the following format:

  ```
  DN-Variable-Name=<#dn="DN-Spec" attr="DN-Attribute-Name"#>
  ```

  In the format example:

  - *DN-Variable-Name* is the name for the name/value pair that this response attribute will return to the Web Agent.
  - *DN-Spec* is the distinguished name of the user group from which you want to retrieve the user attribute.
  - *DN-Attribute-Name* is an attribute associated with an LDAP or ODBC directory object to which the user is related, such as a group or an organizational unit (OU).

- **Active Response**. An active expression associated with the Response Attribute. The expression is a string of variable definitions in the following format:

  ```
  Name=<@lib="LibName" func="FuncName" param="Param"@>
  ```

  In the format example:

  - *Name* is the name of the variable (with WebAgent-HTTP-Header-Variable response attributes) or cookie (with WebAgent-HTTP-Cookie-Variable response attributes) associated with the name/value pairs in the active expression.

- *LibName* is the name of the shared library that supports the Active Response.
- *FuncName* is the name of the actual function in the shared library that implements the Active Response.
- *Param* is an optional list of parameters to be passed to the function in the shared library.

Note: For information about configuring active expressions in responses, rules, or policies, see *Policy Design*.

Sm_PolicyApi_Rule_t

Defines a SiteMinder Rule object.

Syntax

```
typedef struct Sm_PolicyApi_Rule_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                    /* Required */
   char pszDesc[BFSIZE];
   char pszRealmOid[BFSIZE];                /* Required */
   char pszAction[BFSIZE];                  /* Required */
   char pszResource[2*BFSIZE];
   bool bAllowAccess;                       /* Required */
   bool bRegularExpression;                 /* Required */
   char pszActiveExpr[BFSIZE];
   bool bIsEnabled;                         /* Required */
   long nStartTime;
   long nEndTime;
   unsigned char   pszTimeGrid[TIMESIZE];
   struct Sm_PolicyApi_Rule_s* next;
   char pszAgentOid[BFSIZE];                /* Required */
} Sm_PolicyApi_Rule_t;
```

Field	Description
iStructId	Rule data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the rule object.
pszName	Name of the Rule.
pszDesc	Brief description of the rule.

Field	Description
pszRealmOid	Object identifier of the Realm associated with the rule. Required for a domain-specific rule; ignored for a global rule.
pszAction	The type of action the rule is executing. The supported Web Agent Actions consist of the following HTTP operations: Get, Put, and Post.
pszResource	Resource protected by the rule.
bAllowAccess	Flag to allow or deny access to the resource protected by the rule.
bRegularExpression	Flag to perform regular expression pattern matching. Regular expressions are text patterns used for string matching.

pszActiveExpr	Active expression associated with the rule. The expression is a string of variable definitions in the following format:
	The non-alphanumeric characters are required characters .For example:
	`<@lib="LibName" func="FuncName"`
	`        param="FuncParam"@>`
	In the format example:
	■ LibName is the name of the shared library that supports the Active Rule.
	■ FuncName is the name of the actual function in the shared library that implements the Active Rule.
	■ FuncParam is an optional list of parameters to be passed to the function in the shared library.

dbIsEnabled	Flag to enable or disable the rule.
nStartTime	The time when the time restriction becomes effective. This value is stored in standard time_t format. Set *nStartTime* to 0 to start the time restriction immediately.
nEndTime	The time when the time restriction expires. This value is stored in standard time_t format. Set *nEndTime* to 0 to end the time restriction immediately.
pszTimeGrid	An array containing time restrictions for an entire week.

Field	Description
next	Pointer to the next rule structure.
pszAgentOid	Object identifier of agent or agent group associated with the global rule.

Sm_PolicyApi_SAMLAffiliation_t

Defines 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.

Syntax

```
typedef struct Sm_PolicyApi_SAMLAffiliation_s
{
   int iStructId;
   Sm_PolicyApi_SAMLProviderProp_t *pProps;
   struct Sm_PolicyApi_SAMLAffiliation_s *next;
} Sm_PolicyApi_SAMLAffiliation_t;
```

Field	Description
iStructId	SAML affiliation structure ID, defined in Sm_PolicyApi_Structs_t.
pProps	SAML 2.0 metadata properties associated with the affiliation. If you do not assign a value to a property associated with a default value, the default will be assigned.
next	Pointer to the next SAML affiliation structure.

Sm_PolicyApi_SAMLProviderProp_t

Defines a SAML 2.0 metadata property as a name/value pair.

An Sm_PolicyApi_SAMLProviderProp_t structure consists of a single name/value pair. You define a set of properties for a given SAML 2.0 object through a linked list of Sm_PolicyApi_SAMLProviderProp_t structures.

Use the following structures and function to define a set of properties for a SAML 2.0 Service Provider, affiliation, or authentication scheme and associated Identity Provider:

- Sm_PolicyApi_SAMLSP_t
- Sm_PolicyApi_SAMLAffiliation_t
- Sm_PolicyApi_AddSAMLScheme()

Syntax

```
typedef struct Sm_PolicyApi_SAMLProviderProp_s
{
   int iStructId;
   char pszName[BFSIZE];
   char pszValue[BFSIZE];
   struct Sm_PolicyApi_SAMLProviderProp_s *next;
} Sm_PolicyApi_SAMLProviderProp_t;
```

Field	Description
iStructId	SAML 2.0 properties structure ID, defined in Sm_PolicyApi_Structs_t.
pszName	The property name. See Property Lists for a list of valid property names.
pszValue	The value to assign to *pszName*.
next	Pointer to the next SAML 2.0 properties structure.

Example

The following is an example of a helper method that creates an instance of Sm_PolicyApi_SAMLProviderProp_t and assigns it the name/value pair that is passed into it:

```
Sm_PolicyApi_SAMLProviderProp_t * CreateSAMLProp(
                                         const char *pszName,
                                         const char *pszValue)
{
   Sm_PolicyApi_SAMLProviderProp_t * pProp = new
                              Sm_PolicyApi_SAMLProviderProp_t;
   memset(pProp, 0, sizeof(Sm_PolicyApi_SAMLProviderProp_t));
   pProp->iStructId = Sm_PolicyApi_SAMLProviderProp_ID;
   strncpy(pProp->pszName, pszName, BFSIZE);
   strncpy(pProp->pszValue, pszValue, BFSIZE);
   return pProp;
}
```

The following is an example of a helper method that retrieves the value of the property whose name is passed into it:

```
char * FindSAMLParam(const Sm_PolicyApi_SAMLSP_t *pStructSP,
                     const char *pszName)
{
   for(Sm_PolicyApi_SAMLProviderProp_t *pCurProp=pStructSP->pProps;
       pCurProp != NULL;
       pCurProp = pCurProp->next)
   {
       if (0 == strcmp(pCurProp->pszName, pszName))
      {
          return pCurProp->pszValue;
      }
   }

   return NULL;
}
```

Property Lists

Each Sm_PolicyApi_SAMLProviderProp_t structure contains a SAML 2.0 metadata property defined as a name/value pair. A complete set of properties for a particular SAML 2.0 object is defined as a linked list of Sm_PolicyApi_SAMLProviderProp_t structures.

The following metadata properties apply to a given type of SAML 2.0 object:

- Service Provider Properties
- SAML Affiliation Properties

■ SAML 2.0 Authentication Scheme Properties

Note: Some properties are used with multiple object types.

Service Provider Properties

A Service Provider object contains information that an Identity Provider needs to produce assertions for the Service Provider. Service Provider properties are stored within an Sm_PolicyApi_SAMLSP_t structure.

The properties of a SAML 2.0 Service Provider object are listed following.

Property Name	**Comments**
General Properties	
SAML_NAME	String, required.
SAML_DESCRIPTION	String.
SAML_SP_AUTHENTICATION_URL	String, required.
SAML_ENABLED	Boolean. Default: SAML_TRUE.
SAML_SP_DOMAIN	String, required.
Name IDs Tab	
SAML_SP_NAMEID_FORMAT	String. Default: Unspecified.
SAML_SP_NAMEID_TYPE	Integer. Default: 1.
SAML_SP_NAMEID_STATIC	String. Required conditionally.
SAML_SP_NAMEID_ATTRNAME	String. Required conditionally.
SAML_SP_NAMEID_DNSPEC	String. Required conditionally.
SAML_AFFILIATION	String.
SAML_KEY_SPID	String, required.
	String, required.

Property Name	Comments
SAML_MAJOR_VERSION	Integer. Default: 2.
SAML_MINOR_VERSION	Integer. Default: 0.
SAML_SKEWTIME	Integer. Default: 30.
SAML_DISABLE_SIGNATURE_ PROCESSING	Boolean. Default: SAML_FALSE.
SAML_DSIG_VERINFO_ISSUER_DN	String. Required conditionally.
SAML_DSIG_VERINFO_SERIAL_NUMBER	String. Required conditionally.
SSO Properties	
SAML_AUDIENCE	String, required.
SAML_SP_ASSERTION_CONSUMER_ DEFAULT_URL	String, required.
SAML_ENABLE_SSO_ARTIFACT_ BINDING	Boolean. Default: SAML_FALSE.
SAML_SP_ARTIFACT_ENCODING	String. Default: FORM. Applies if no value is provided and SAML_ENABLE_SSO_ ARTIFACT_BINDING is SAML_TRUE.
SAML_SP_IDP_SOURCEID	String. Default: A hex-encoded SHA-1 hash of the SAML_SP_IDPID value.
SAML_SP_PASSWORD	String. Required conditionally (see page 34).
SAML_ENABLE_SSO_POST_BINDING	Boolean. Default: SAML_FALSE.
SAML_SSOECPPROFILE	Boolean. Default: SAML_FALSE.

Property Name	**Comments**
SAML_SP_REQUIRE_SIGNED_ AUTHNREQUESTS	Boolean. Default: SAML_FALSE.
SAML_SP_AUTHENTICATION_LEVEL	Integer. Default: 5.
SAML_SP_AUTHN_CONTEXT_CLASS_REF	String. Default: urn:oasis:names:tc:SAML: 2.0:ac:classes:Password
SAML_SP_VALIDITY_DURATION	Integer. Default: 60.
SAML_SP_STARTTIME	Long.
SAML_SP_ENDTIME	Long.
SLO Properties	
SAML_SLO_REDIRECT_BINDING	Boolean. Default: SAML_FALSE.
SAML_SLO_SERVICE_VALIDITY_ DURATION	Integer. Default: 60. Applies if no value is provided and SAML_SLO_REDIRECT_BINDING is SAML_TRUE.
SAML_SLO_SERVICE_URL	String. Required conditionally.
SAML_SLO_SERVICE_RESPONSE_URL	String.
SAML_SLO_SERVICE_CONFIRM_URL	String.
IPD Properties	
SAML_SP_ENABLE_IPD	Boolean. Default: SAML_FALSE.
SAML_SP_IPD_SERVICE_URL	String. Required conditionally.
SAML_SP_COMMON_DOMAIN	String. Required conditionally.
SAML_SP_PERSISTENT_COOKIE	Boolean. Default: SAML_FALSE.

Property Name	Comments
Encryption Properties	
SAML_SP_ENCRYPT_ID	Boolean. Default: SAML_FALSE.
SAML_SP_ENCRYPT_ASSERTION	Boolean. Default: SAML_FALSE.
SAML_SP_ENCRYPT_BLOCK_ALGO	String. Default: tripledes.
SAML_SP_ENCRYPT_KEY_ALGO	String. Default: rsa-v15.
SAML_SP_ENCRYPT_CERT_ISSUER_DN	String. Required conditionally.
SAML_SP_ENCRYPT_CERT_SERIAL_ NUMBER	String. Required conditionally.
Attribute Service Properties	
SAML_SP_ATTRSVS_ENABLE	Boolean. Default: False (0).
SAML_SP_ATTRSVS_VALIDITY_DURATION	Integer. Default: 60 (seconds)
SAML_SP_ATTRSVS_SIGN_ASSERTION	Boolean. Default: False (0).
SAML_SP_ATTRSVS_LDAP_SEARCH_SPEC	String.
SAML_SP_ATTRSVS_ODBC_SEARCH_SPEC	String.
SAML_SP_ATTRSVS_WINNT_SEARCH_SPEC	String.
SAML_SP_ATTRSVS_CUSTOM_SEARCH_SPEC	String.
SAML_SP_ATTRSVS_AD_SEARCH_SPEC	String.
Advanced Properties	
SAML_SP_PLUGIN_CLASS	String.
SAML_SP_PLUGIN_PARAMS	String.

SAML Affiliation Properties

The properties of a SAML 2.0 affiliation object are listed below. Properties are grouped according to the way they are presented on the SAML Affiliation Properties dialog box.

SAML affiliation properties are stored within an Sm_PolicyApi_SAMLAffiliation_t structure.

Property Name	**Comments**
General Properties	
SAML_NAME	String, required.
SAML_DESCRIPTION	String
SAML_KEY_AFFILIATION_ID	String, required.
SAML_MAJOR_VERSION	Integer. Default: 2.
SAML_MINOR_VERSION	Integer. Default: 0.
SAML_OID	String. SiteMinder supplies the object identifier when an affiliation object is created.
Name IDs Tab	
SAML_SP_NAMEID_FORMAT	String. Default: Unspecified.
SAML_SP_NAMEID_TYPE	Integer. Default: 1.
SAML_SP_NAMEID_STATIC	String. Required conditionally.
SAML_SP_NAMEID_ATTRNAME	String. Required conditionally.
SAML_SP_NAMEID_DNSPEC	String. Required conditionally.
Users Tab	
SAML_IDP_XPATH	String.

Property Name	Comments
SAML_IDP_LDAP_SEARCH_SPEC	String.
SAML_IDP_ODBC_SEARCH_SPEC	String.
SAML_IDP_WINNT_SEARCH_SPEC	String.
SAML_IDP_CUSTOM_SEARCH_SPEC	String.
SAML_IDP_AD_SEARCH_SPEC	String.

SAML 2.0 Authentication Scheme Properties

The properties listed in this section define:

- Authentication schemes based on the SAML 2.0 Template.
- Metadata properties of the associated Identity Provider. The properties are stored with the authentication scheme.

You define a SAML 2.0 authentication scheme to represent an Identity Provider for a particular Service Provider.

The properties of a SAML 2.0 authentication scheme and its associated Identity Provider are listed below. Properties are grouped according to the way they are presented on the SAML Authentication Scheme Properties dialog box for the SAML 2.0 Template.

You define properties for a SAML 2.0 authentication scheme and its associated Identity Provider by calling Sm_PolicyApi_AddSAMLScheme().

Property Name	Comments
General Properties	
SAML_NAME	String, required.
SAML_DESCRIPTION	String
Scheme Setup Tab	
SAML_IDP_SPID	String, required.
SAML_KEY_IDPID	String, required.
SAML_MAJOR_VERSION	Integer. Default: 2.
SAML_MINOR_VERSION	Integer. Default: 0.

Property Name	Comments
SAML_SKEWTIME	Integer. Default: 30.
SAML_DISABLE_SIGNATURE_ PROCESSING	Boolean. Default: SAML_FALSE.
SAML_DSIG_VERINFO_ISSUER_DN	String. Required conditionally (see page 9).
SAML_DSIG_VERINFO_SERIAL_NUMBER	String. Required conditionally (see page 10).
Additional Configuration - Users Tab	
SAML_IDP_XPATH	String.
SAML_IDP_LDAP_SEARCH_SPEC	String.
SAML_IDP_ODBC_SEARCH_SPEC	String.
SAML_IDP_WINNT_SEARCH_SPEC	String.
SAML_IDP_CUSTOM_SEARCH_SPEC	String.
SAML_IDP_AD_SEARCH_SPEC	String.
SAML_AFFILIATION	String.
Additional Configuration - SSO Tab	
SAML_IDP_SSO_REDIRECT_MODE	Integer. Default: 0.
SAML_IDP_SSO_DEFAULT_SERVICE	String, required.
SAML_AUDIENCE	String, required.
SAML_IDP_SSO_TARGET	String.
SAML_ENABLE_SSO_ARTIFACT_ BINDING	Boolean. Default: SAML_FALSE.
SAML_KEY_IDP_SOURCEID	String. Default: A hex-encoded SHA-1 hash of the SAML_KEY_IDPID value.
SAML_IDP_ARTIFACT_RESOLUTION_ DEFAULT_SERVICE	String. Required conditionally (see page 12).

Property Name	Comments
SAML_IDP_BACKCHANNEL_AUTH_TYPE	Integer. Default: 0.
SAML_IDP_SPNAME	String. Required conditionally (see page 19).
SAML_IDP_PASSWORD	String. Required conditionally (see page 14).
SAML_ENABLE_SSO_POST_BINDING	Boolean. Default: SAML_FALSE.
SAML_IDP_SSO_ENFORCE_SINGLE_ USE_POLICY	Boolean. Default: SAML_TRUE.
SAML_SSOECPPROFILE	Boolean. Default: SAML_FALSE.
SAML_IDP_SIGN_AUTHNREQUESTS	Boolean. Default: SAML_FALSE.
Additional Configuration - SLO Tab	
SAML_SLO_REDIRECT_BINDING	Boolean. Default: SAML_FALSE.
SAML_SLO_SERVICE_VALIDITY_ DURATION	Integer. Default: 60. Applies if no value is provided and SAML_SLO_REDIRECT_BINDING is SAML_TRUE.
SAML_SLO_SERVICE_URL	String. Required conditionally.
SAML_SLO_SERVICE_RESPONSE_URL	String.
SAML_SLO_SERVICE_CONFIRM_URL	String.
Additional Configuration - Encryption Tab	
SAML_IDP_REQUIRE_ENCRYPTED_ ASSERTION	Boolean. Default: SAML_FALSE.
SAML_IDP_REQUIRE_ENCRYPTED_ NAMEID	Boolean. Default: SAML_FALSE.

Property Name	Comments
Additional Configuration - Attributes Tab	
SAML_IDP_SAMLREQ_ENABLE	Boolean. Default: False (0).
SAML_IDP_SAMLREQ_REQUIRE_SIGNED_ ASSERTION	Boolean. Default: False (0).
SAML_IDP_SAMLREQ_ATTRIBUTE_SERVICE	String.
Additional Configuration - NameID tab	
SAML_IDP_SAMLREQ_NAMEID_FORMAT	String.
SAML_IDP_SAMLREQ_NAMEID_TYPE	Integer. Default: 1 (User Attribute)
SAML_IDP_SAMLREQ_NAMEID_STATIC	String.
SAML_IDP_SAMLREQ_NAMEID_ATTR_NAM E	String.
SAML_IDP_SAMLREQ_NAMEID_DN_SPEC	String.
SAML_IDP_SAMLREQ_NAMEID_ALLOW_ NESTED	Boolean. Deafult: False (0).
Additional Configuration - Advanced Tab	
SAML_SP_PLUGIN_CLASS	String.
SAML_SP_PLUGIN_PARAMS	String.
SAML_IDP_REDIRECT_URL_USER_ NOT_FOUND	String.
SAML_IDP_REDIRECT_MODE_USER_ NOT_FOUND	Integer. Default: 0.
SAML_IDP_REDIRECT_URL_FAILURE	String.
SAML_IDP_REDIRECT_MODE_FAILURE	Integer. Default: 0.
SAML_IDP_REDIRECT_URL_INVALID	String.
SAML_IDP_REDIRECT_MODE_INVALID	Integer. Default: 0.

More Information:

SAML 2.0 Property Reference (see page 769)

Sm_PolicyApi_SAMLRequesterAttr_t

Defines an attribute that can be requested by a SAML Requester in an AttributeQuery message.

Syntax

```
typedef struct Sm_PolicyApi_SAMLRequesterAttr_s
{
   int iStructId;
   Sm_PolicyApi_SAMLSPAttrNameFormat_t nAttrNameFormat;
   char pszLocalName[BFSIZE];
   char pszName[BUFSIZE];
   struct Sm_PolicyApi_SAMLRequesterAttr_s* next;
} Sm_PolicyApi_SAMLRequesterAttr_t;
```

Field	Description
iStructId	SAML attribute structure ID, defined in Sm_PolicyApi_Structs_t.
nAttrNameFormat	The format of the attribute name, as defined by the SAML 2.0 standard.
pszLocalName	Name of the attribute as defined in the SAML 2.0 authentication scheme.
pszName	Actual name of the attribute requested from the Attribute Authority.
next	Pointer to the next requester attribute structure.

Sm_PolicyApi_SAMLSP_t

Defines a SAML 2.0 Service Provider object for an Identity Provider.

A Service Provider offers services (such as access to applications and other resources) to principals within a federation.

Syntax

```
typedef struct Sm_PolicyApi_SAMLSP_s
{
   int iStructId;
   Sm_PolicyApi_SAMLProviderProp_t *pProps;
   Sm_PolicyApi_IPAddress_t *pIPAddress;
   unsigned char   pszTimeGrid[TIMESIZE];
   struct Sm_PolicyApi_SAMLSP_s *next;
} Sm_PolicyApi_SAMLSP_t;
```

Field	Description
iStructId	SAML Service Provider structure ID, defined in Sm_PolicyApi_Structs_t.
pProps	SAML 2.0 metadata properties associated with the Service Provider. If you do not assign a value to a property associated with a default value, the default will be assigned.
pIPAddress	The Service Provider's IP address.
pszTimeGrid	An array containing time restrictions for an entire week.
next	Pointer to the next Service Provider structure.

Sm_PolicyApi_SAMLSPAssertionConsumerService_t

Defines a structure that supports the Assertion Consumer Service.

Syntax

```
typedef struct Sm_PolicyApi_SAMLSPAssertionConsumerService_s
{
   int iStructId;
   int iIndex;
   Sm_PolicyApi_SAMLSPACSBinding_t nACSBinding;
   char pszAssertionConsumerURL[BFSIZE};
   bool bIsDefault;
   struct Sm_PolicyApi_SAMLSPAssertionConsumerService_s* next;
} Sm_PolicyApi_SAMLSPAsserttionConsumer_t;
```

Field	**Description**
iStructId	SAML Assertion Consumer Service ID, defined in Sm_PolicyApi_Structs_t.
iIndex	Index value assigned to this Assertion Consumer Service. The value must be 0 or a positive integer.
nACSBinding	One of the following bindings associated with the Assertion Consumer Service: ■ Sm_PolicyApi_SAMLSP_HTTP_Post ■ Sm_PolicyApi_SAMLSP_HTTP_Artifact ■ Sm_PolicyApi_SAMLSP_PAOS
pszAssertionConsumer-URL	Location of the Assertion Consumer Service. In the case of the HTTP-Artifact binding, this is the URL that contains a SAML artifact and target as query parameters, which are used by the credential collector to obtain the SAML assertion and redirect the user to the target. In the case of the HTTP-Post binding, it is the destination site URL to which the user's browser must POST a generated assertion.
bIsDefault	Specifies whether this Assertion Consumer Service is the default.
next	Pointer to the defintion of the next Assertion Consumer Service.

Sm_PolicyApi_SAMLSPAttr_t

Defines an attribute of a principal for a particular SAML 2.0 Service Provider.

Syntax

```
typedef struct Sm_PolicyApi_SAMLSPAttr_s
{
   int iStructId;
   Sm_PolicyApi_SAMLSPAttrNameFormat_t nAttrNameFormat;
   char pszValue[BFSIZE];
   bool bEncrypted;
   struct Sm_PolicyApi_SAMLSPAttr_s* next;
   Sm_PolicyApi_SAMLSPAttrMode_t  nMode;
} Sm_PolicyApi_SAMLSPAttr_t;
```

Field	Description
iStructId	SAML attribute structure ID, defined in Sm_PolicyApi_Structs_t.
nAttrNameFormat	The format of the attribute name, as defined by the SAML 2.0 standard.

pszValue	The attribute's name and value, in one of these formats:
	■ Static attributes:
	variableName=value
	■ User attributes:
	variableName=<%userattr="AttrName"%>
	■ DN attributes:
	variablName=<#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"

bEncrypted	Specifies whether the attribute is encrypted.
next	Pointer to the next attribute structure.
nMode	One of the three following uses of an attribute: ■ Sm_PolicyApi_SAMLSP_SSO_Only = 0 ■ Sm_PolicyApi_SAMLSP_Atribute_Only = 1 ■ Sm_PolicyApi_SAMLSP_Both = 2

Sm_PolicyApi_Scheme_t

Defines a SiteMinder authentication scheme object.

Syntax

```
typedef struct Sm_PolicyApi_Scheme_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                   /* Required */
   char pszDesc[BFSIZE];
   int nLevel;                             /* Required */
   char pszLib[BFSIZE];
   char pszParam[BFSIZE];
   char pszSecret[BFSIZE];
   bool bIsTemplate;
   bool bIsUsedbyAdmin;
   Sm_Api_SchemeType_t nType;              /* Required */
   bool bAllowSaveCreds;                   /* Required */
   bool bIsRadius;                         /* Required */
   bool bIgnorePwCheck;                    /* Required */
   struct Sm_PolicyApi_Scheme_s* next;
} Sm_PolicyApi_Scheme_t;
```

Field	Description
iStructId	Authentication scheme data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the authentication scheme object.
pszName	Name of the authentication scheme.
pszDesc	Brief description of the authentication scheme.
nLevel	The protection level of the authentication scheme. The level may vary from 1 through 1000. The higher the number, the more secure is the scheme. With Anonymous authentication schemes, set this value to 0.
pszLib	The name of the shared library that implements the custom authentication scheme.
pszParam	Information that is passed on to the custom authentication scheme.

Field	Description
pszSecret	A shared secret known to two parties for the purpose of establishing secure data exchange. This information is passed on to the custom authentication scheme.
bIsTemplate	Flag to indicate whether the authentication scheme is a template. **Note:** Setting an authentication scheme as a template with the C Policy Management API is deprecated in SDK v6.0 SP3.
bIsUsedbyAdmin	Flag to indicate if the custom authentication scheme can be used to authenticate administrators.
nType	The type of the authentication scheme, defined in Sm_Api_SchemeType_t.
bAllowSaveCreds	Flag to allow user credentials to be saved.
bIsRadius	Flag to indicate if the scheme is of type Radius.
bIgnorePwCheck	If this flag is set to true, password policies for the authentication scheme will be disabled.
next	Pointer to the next Scheme structure.

Sm_PolicyApi_SharedSecretPolicy_t

Defines a shared secret policy.

In the Administrative UI, a shared secret policy is defined in the Shared Secret Rollover tab of the Key Management dialog box.

Syntax

```
typedef struct Sm_PolicyApi_SharedSecretPolicy_s
{
   int iStructID;
   bool bIsEnabled;
   int iRolloverPeriod;
   int iRolloverFrequency;
} Sm_PolicyApi_SharedSecretPolicy_t;
```

Field	Description
iStructID	Authentication scheme data structure ID, defined in Sm_PolicyApi_Structs_t.
bIsEnabled	Is shared secret rollover enabled?
iRolloverPeriod	The unit of time (hourly, daily, weekly, monthly) that is used with *iRolloverFrequency* to determine how often the shared secret is automatically changed (for example, every 3 days, every 2 months, etc.). Valid values are specified in Sm_PolicyApi_SecretRolloverPeriod_t.
iRolloverFrequency	Specifies the number of *iRolloverPeriod* units between rollovers.

Sm_PolicyApi_Server_t

Defines TCP/IP connectivity information for a Policy Server.

Syntax

```
typedef struct Sm_PolicyApi_Server_s
{
   int iStructId;
   char pszIpAddr[BFSIZE];                 /* Required */
   long nPort[3];
   long nClusterSeq;
   struct Sm_PolicyApi_Server_s* next;
} Sm_PolicyApi_Server_t;
```

Field	Description
iStructId	Policy server structure ID, defined in Sm_PolicyApi_Structs_t.
pszIpAddr	The TCP/IP address of a Policy Server.
nPort	Prior to v6.0 of SiteMinder, this is an array of TCP/IP ports for Accounting, Authentication, and Authorization services. Beginning with SiteMinder v6.0, only the Policy Server port needs to be specified.
nClusterSeq	The cluster sequence number for this server. For a non-cluster server, omit the parameter. For a cluster server, specify the cluster sequence number, starting from one. Specifying a sequence number that is equal to an existing sequence number will result in an error.
next	Pointer to the next server structure.

Sm_PolicyApi_TrustedHost_t

Defines a trusted host object.

Syntax

```
typedef struct Sm_PolicyApi_TrustedHost_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];
   char pszDesc[BFSIZE];
   char pszIPAddr[BFSIZE];
   char pszSecret[BFSIZE];
   bool bIs4xHost;
   struct Sm_PolicyApi_TrustedHost_s* next;
   bool bRolloverEnabled;
} Sm_PolicyApi_TrustedHost_t;
```

Field	Description
iStructId	Data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the trusted host object.
pszName	The trusted host name.
pszDesc	Brief description of the trusted host.
pszIPAddr	The IP address of the trusted host.
pszSecret	The shared secret of the trusted host.
bIs4xHost	Internal use only.
next	Pointer to the next trusted host structure.
bRolloverEnabled	Indicates whether or not shared secret rollover is enabled for this trusted host.

Sm_PolicyApi_User_t

Defines a SiteMinder User object.

Syntax

```
typedef struct Sm_PolicyApi_User_s
{
   int iStructId;
   char pszUserPolicyOid[BFSIZE];            /* Required */
   char pszUserDirOid[BFSIZE];               /* Required */
   char pszPath[BFSIZE];                     /* Required */
   char pszClass[BFSIZE];                    /* Required */
   Sm_PolicyResolution_t nPolicyResolution;  /* Required */
   int nFlags;                               /* Required */
   struct Sm_PolicyApi_User_s* next;
} Sm_PolicyApi_User_t;
```

Field	Description
iStructId	User data structure ID, defined in Sm_PolicyApi_Structs_t.
pszUserPolicyOid	The object identifier of the user policy.
pszUserDirOid	The object identifier of the user directory
pszPath	User's distinguished name (DN).
pszClass	The object class as understood by a user directory, such as organizationalUnit in LDAP or User in NT.
nPolicyResolution	The relationship between two policy objects. Policy resolutions are enumerated in Sm_PolicyResolution_t.
nFlags	A bitmask using the bits defined in Policy Flags.
next	Pointer to the next User structure.

Sm_PolicyApi_UserContext_t

Makes user context information available to callers of the Policy Management API.

Syntax

```
typedef struct Sm_PolicyApi_UserContext_s
{
   int iStructId;
   Sm_Api_UserContext_t *pUserContext;
   struct Sm_PolicyApi_UserContext_s* next;
} Sm_PolicyApi_UserContext_t;
```

Field	Description
iStructId	User data structure ID, defined in Sm_PolicyApi_Structs_t.
pUserContext	Pointer to the user context information.
next	This field is always set to null.

More Information:

Sm_PolicyApi_GetTrustedHostByName() (see page 456)

Sm_PolicyApi_UserDir_t

Defines a SiteMinder user directory object.

Syntax

```
typedef struct Sm_PolicyApi_UserDir_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];                    /* Required */
   char pszDesc[BFSIZE];
   char pszNamespace[BFSIZE];               /* Required */
   char pszServer[BFSIZE];                  /* Required */
   char pszSearchRoot[BFSIZE];
   char pszUserLookupStart[BFSIZE];
   char pszUserLookupEnd[BFSIZE];
   char pszUsername[BFSIZE];
   char pszPassword[BFSIZE];
   int nSearchResults;                      /* Required */
   int nSearchScope;                        /* Required */
   int nSearchTimeout;                      /* Required */
   bool bSecureConnection;                  /* Required */
   bool bRequireCredentials;                /* Required */
   char pszDisabledAttr[BFSIZE];
   char pszUniversalIDAttr[BFSIZE];
   char pszODBCQuerySchemeOid[BFSIZE];
   char pszAnonymousId[BFSIZE];
   char pszPasswordData[BFSIZE];
   char pszPasswordAttribute[BFSIZE];
   char pszEmailAddressAttr[BFSIZE];
   char pszChallengeRespAttr[BFSIZE];
   struct Sm_PolicyApi_UserDir_s* next;
} Sm_PolicyApi_UserDir_t;
```

Field	**Description**
iStructId	User directory data structure ID, defined in Sm_PolicyApi_Structs_t.
pszOid	The object identifier of the user directory object.
pszName	Name of the user directory.
pszDesc	Brief description of the user directory.

Field	Description
pszNamespace	Mandatory field that designates the specific directory service being connected to (for example, LDAP:, ODBC:, WinNT:, AD:, or Custom:).
pszServer	Mandatory field. This is an overloaded field whose contents depend upon the namespace: ■ ODBC - Data source name. ■ NT - Domain name. ■ LDAP or AD - An IP address or an IP address and port number in the format *IP_address:port_number*. The port number 389 is assumed if no port number is specified. ■ Custom - Library name.
pszSearchRoot	One of the following values: ■ With LDAP directories, the location in the LDAP tree that serves as the starting point for the directory connection-typically, an organization (o) or organizational unit (ou). The Policy Server begins searching at the root when locating a user. ■ With custom directories, any parameters to pass to the custom library.
pszUserLookupStart	The User DN Lookup Start allows users to authenticate by entering only a part of the user name, without having to enter an entire DN string. Identifying unique and non-unique segments of the user DN string does this. Use this field with LDAP directories only.
pszUserLookupEnd	The User DN Lookup End allows users to authenticate by entering only a part of the user name, without having to enter an entire whole DN string. Use this field with LDAP directories only.
pszUsername	The user name needed to access a user directory.
pszPassword	The password needed to access a user directory.

Field	**Description**
nSearchResults	The maximum number of records that can be returned from a search of an LDAP or custom directory.
nSearchScope	The extent to which SiteMinder looks for users and user groups below *pszSearchRoot* in an LDAP directory - all levels below the root (subtree) or just one level below the root. Specify 1 for one level down or 2 for subtree.
nSearchTimeout	The maximum amount of time, in seconds, that SiteMinder will query an LDAP or custom directory.
bSecureConnection	This flag must be enabled when accessing an LDAP or custom directory over SSL. Enabling Secure Connect means that SiteMinder performs secure authentication and encrypted transmissions.
bRequireCredentials	Flag to specify credentials necessary to authenticate against a user directory
pszDisabledAttr	Name of the user directory attribute that SiteMinder uses to keep track of a user's enabled or disabled state. Applies to LDAP and ODBC directories, and possibly to custom directories.
pszUniversalIDAttr	Name of the user directory attribute that has been designated as the Universal ID. Typically, the Universal ID differs from the user's login ID, and the Universal ID is used to look up user information. Applies to LDAP, ODBC, and WinNT directories, and possibly to custom directories.
pszODBCQuerySchemeOid	The object identifier for a set of ODBC queries that SiteMinder uses to query the ODBC directory.

Field	Description
pszAnonymousId	Name of the user directory attribute that is designated as the anonymous user DN. This DN is defined in the anonymous authentication scheme. Anonymous users impersonate this DN to gain access to the resources associated with the anonymous authentication scheme. Applies to LDAP directories, and possibly to custom directories.
pszPasswordData	Name of the user directory attribute that SiteMinder uses to store password policy information. Applies to LDAP and ODBC directories, and possibly to custom directories.
pszPasswordAttribute	Name of the user directory attribute that contains the user's password, as defined using Password Services. Applies to LDAP and ODBC directories, and possibly to custom directories.
pszEmailAddressAttr	Reserved for future use.
pszChallengeRespAttr	Name of the user directory attribute that contains a response to return to the user, such as a hint for a forgotten password. Applies to LDAP directories, and possibly to custom directories.
next	Pointer to the next directory structure.

Remarks

Fields apply to all types of directories (LDAP, ODBC, WinNT, and custom) unless individual directory types are specified.

Fields that apply to LDAP directories also apply to Active Directories.

Sm_PolicyApi_UserPasswordState_t

Information regarding all PasswordState virtual attributes is returned using the SmPolicyApi_UserPasswordState_t structure. This structure coexists with a User object, which is restricted by the UserDirectory OID and a User DN string. The structure can be retrieved, created, or updated through the C Policy Management API.

Syntax

```
typedef struct Sm_PolicyApi_UserPasswordState_s
{
   int iLoginFailures;
   time_t tLastLogin;
   time_t tPrevLogin;
   time_t tDisabled;
   time_t tLastPWChange;
} Sm_PolicyApi_UserPasswordState_t;
```

Field	Description
iLoginFailures	Specifies how many times the user has failed to log in since the last successful login.
tLastLogin	Specifies the last time the user successfully logged in.
tPrevLogin	Specifies the second-to-last time the user successfully logged in.
tDisabled	Specifies the time the user was disabled.
tLastPWChange	Specifies the last time the user changed his password. If this value updates the user directory setting for the last time the password was changed, and the password is reset outside of SiteMinder, the password policy preventing password reuse may not work as expected. The value 0 may be returned in this field in the following cases: ■ The user begins the procedure to change his password but does not complete it. ■ Password history is cleared through a call to Sm_PolicyApi_SetUserPasswordState().

Sm_PolicyApi_Variable_t

Defines a variable object that can be used in a variable expression for a policy or a response. Variable objects are managed by the Variable Functions.

A 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.

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 on 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.

Syntax

```
typedef struct Sm_PolicyApi_Variable_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];
   char pszDesc[BFSIZE];
   char pszVariableTypeOid[BFSIZE];
   char pszDefinition[BFSIZE];
   char pszMetaData[BFSIZE];
   int nReturnType;
   bool bPreFetchFlag;
   char pszDomainOid[BFSIZE];
   Sm_PolicyApi_Oid_t* pNestedVariableList;
   struct Sm_PolicyApi_Variable_s*   next;
} Sm_PolicyApi_Variable_t;
```

Field	Description
iStructId	Data structure ID, defined in Sm_PolicyApi_Structs_t.

Field	Description
pszOid	The unique object ID of the variable object.
pszName	The user-defined name of the variable object.
pszDesc	Optional text describing the variable object.
pszVariableTypeOid	The unique object ID of the variable type.
pszDefinition	Information needed to obtain the value of the variable at runtime.
pszMetaData	Reserved for use by the optional CA TransactionMinder product.
nReturnType	The data type of the variable value: ■ Sm_PolicyApi_VarReturnTypes_Boolean ■ Sm_PolicyApi_VarReturnTypes_Number ■ Sm_PolicyApi_VarReturnTypes_String ■ Sm_PolicyApi_VarReturnTypes_Date
bPreFetchFlag	Not currently used.
pszDomainOid	The unique object ID of the associated domain.
pNestedVariableList	A linked list of nested variable OIDs that are part of the definition of this variable.
next	Pointer to the next variable object structure.

Variable Definition

You define a variable by specifying where the variable's value can be found. You do so through the *pszDefinition* field.

The value of this field 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 *pszDefinition* field for each type:

- **Post**

  The *pszDefinition* field 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 *pszDefinition* field 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. With this item name, the variable value is the type of action specified in the request (for example, GET or POST).
  - Resource. With this item name, the variable value is the target resource (for example, /directory_name/).
  - Server. With this item name, the variable value is the full server name specified in the request (for example, server.company.com).

- **Static**

  The *pszDefinition* field contains the actual value that will be compared against the user-supplied data at runtime. For example, a Static variable of return type Sm_PolicyApi_VarReturnTypes_Date might be assigned the string value 2004-01-01. During authorization, this assigned date is compared against a user-supplied date.

- **UserContext**

  The *pszDefinition* field 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 ItemName element can contain one of the following values:

  - DirectoryEntryProperty
  - DirectoryNameSpace
  - DirectoryPath
  - DirectoryServer
  - IsUserContext
  - SessionId
  - UserPath

- UserProperty
- UserName

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.

- **WebService**

  The *pszDefinition* field 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 *pszDefinition*, 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.

Element	Description
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.
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 *pszDefinition* field should not be formatted with spaces, tabs, and return characters. For example, a RequestContext variable for a Resource attribute would be passed in *pszDefinition* as follows:

```
<RequestContextVariableDef><ItemName>Resource</ItemName></RequestContextVariableD
ef>
```

Sm_PolicyApi_VariableType_t

Defines a supported variable object type. Variable types are read-only. They cannot be created or deleted through the Policy Management Variable Functions.

Syntax

```
typedef struct Sm_PolicyApi_VariableType_s
{
   int iStructId;
   char pszOid[BFSIZE];
   char pszName[BFSIZE];
   char pszDesc[BFSIZE];
   char pszFilter[BFSIZE];
   struct Sm_PolicyApi_VariableType_s*    next;
} Sm_PolicyApi_VariableType_t;
```

Field	Description
iStructId	Data structure ID defined in Sm_PolicyApi_Structs_t.
pszOid	The unique object ID of the variable type object.
pszName	One of the following object type names: ■ Post ■ RequestContext ■ Static ■ UserContext ■ WebService If you have installed the optional CA SOA Security Manager product, the following variable types are also available: ■ SAMLAssertion ■ Transport ■ XMLAgent ■ XMLBody ■ XMLEnvelopeHeader You cannot create these variables using the SDK; you must use the Administrative UI.
pszDesc	The description of the variable type object-for example, Form Post Variables.
pszFilter	Not currently used.

Field	Description
next	Pointer to the next variable type object structure.

Sm_PolicyApi_WSFEDProviderProp_t

Defines a linked list of WS-Federation Provider properties, that is, name/value pairs.

An Sm_PolicyApi_WSFEDProviderProp_t structure consists of a single name/value pair. You define a set of properties for a given WS-Federation object through a linked list of Sm_PolicyApi_WSFEDProviderProp_t structures.

Syntax

```
typedef struct Sm_PolicyApi_WSFEDProviderProp_s
{
        int iStructId;
        char pszName[BFSIZE];
        char pszValue[BFSIZE];
        Sm_PolicyApi_WSFEDProviderProp_t* next;
} Sm_PolicyApi_WSFEDProviderProp_t;
```

Parameters

iStructId

ID of the structure in Sm_PolicyAp_Structs_t. Should be set to Sm_PolicyApi_WSFEDProviderProp_ID.

pszName

Name of the WS-Federation Provider property.

pszValue

Value of the WS-Federation Provider property.

next

Pointer to the next WS-Federation Provider property data in the linked list.

Each Sm_PolicyApi_WSFEDProviderProp_t structure contains a WS-Federation metadata property defined as a name/value pair. A complete set of properties for a particular object is defined as a linked list of Sm_PolicyApi_WSFEDProviderProp_t structures.

The following metadata properties apply to WS-Federation objects types:

- Common properties
- Properties for defining a Resource Partner
- Properties for defining an Account Partner

Optional properties are specified in square brackets.

For Boolean values, a value of 1 denotes true; any other value denotes false.

The Property Name column also includes the corresponidng C Policy Management API macro name.

Common Properties

The following table specifies the metadata properties that are common to defining a Resource Partner or an Account Partner:

Property Name	Type	Description
General		
Name WSFED_NAME	String	Name of the provider.
[Description] WSFED_DESCRIPTION	String	Brief description of the provider.
[SkewTime] WSFED_SKEW_TIME	String	The skew time between consumer and producer sides in seconds. This value is used to calculate validity duration of assertions and of SLO requests. The default value is 30.
Versioning		
[WSFEDMajorVersion] WSFED_MAJOR_VERSION	Int	Version of WSFED protocol supported by this provider. The value of this property has to be set to 1.
[WSFEDMinorVersion] WSFED_MINOR_VERSION	Int	Version of WSFED protocol supported by this provider. The value of this property has to be set to 0.
[WSFEDSAMLMajorVersion] WSFED_SAML_MAJOR_ VERSION	Int	Version of SAML protocol supported by this provider. The value of this property has to be set to 1.
[WSFEDSAMLMinorVersion] WSFED_SAML_MINOR_ VERSION	Int	Version of WSFED protocol supported by this provider. The value of this property has to be set to 1.

Resource Partner Properties

The following table lists the metadata properties used to define a Resource Partner:

Property Name	Type	Description
Domain WSFED_RP_DOMAIN	OID	The Domain OID where this Resource Partner is defined
[Enabled] WSFED_ENABLED	Bool	Boolean indicating if the provider is enabled. If not provided, defaults to true. This property does not get stored physically to the property collection but is used to enable underlying policy.
NetegrityAffiliateMinderAuthURL WSFED_RP_AUTHENTICATION_URL	String	The protected URL used to authenticate Resource Partner users.
NameID		
[NameIdFormat] WSFED_RP_NAMEID_FORMAT	String	The URI for a WSFED name identifier.
[NameIdType] WSFED_RP_NAMEID_TYPE	Int	Represents the type of name identifier: 0 - Static Text 1 - User Attribute 2 - DN Attribute Defaults to 1
[NameIdStatic] WSFED_RP_NAMEID_STATIC	String	The static text to be used as the name identifier when the NameIdType == 0. The Policy Management API will return an error if no value is specified for this property and NameIdType==0.

Property Name	Type	Description
[NameIdAttrName] WSFED_RP_NAMEID_ATTR_NAME	String	The attribute name (user or DN) which holds the name identifier when NameIdType == 1 or NameIdType == 2. If "NameIdType" is set to "1" or "2", "NameIdAttrName" property should have a value, otherwise the Policy Management API will return an error.
[NameIdDNSpec] WSFED_RP_NAMEID_DN_SPEC	String	The DN spec used when the NameIdType == 2. If "NameIdType" is set to "2", "NameIdDNSpec" property should have a value, otherwise the Policy Management API will return error.
[NameIdAllowNested] WSFED_RP_NAMEID_ALLOWED_ NESTED	Bool	Flag indicating whether nested groups are allowed when selecting a DN attribute for the name identifier. Defaults to zero.
General		
KEY_RPID WSFED_KEY_RPID	String	The Resource Partner ID for WSFED Assertion Consumer. Must be a URI less than 1024 characters in length. Also this is the key using which properties associated to a provider can be looked up.
APID WSFED_APID	String	The Resource Partner ID of the WSFED Assertion Producer.
SSO		
[AuthenticationMethod] WSFED_RP_AUTHENTICATION_METHOD	String	The authentication method to use in the assertion.

Property Name	Type	Description
[ValidityDuration] WSFED_RP_VALIDITY_DURATION	Int	An integer number of seconds for which a generated assertion is valid. If not provided during Resource Partner creation, the default is 60 seconds.
AssertionConsumerDefaultURL WSFED_RP_ASSERTION_CONSUMER_ DEFAULT_URL	String	The default WSFED Assertion Consumer to use.
[AuthenticationLevel] WSFED_RP_AUTHENTICATION_LEVEL	Int	The principal must have authenticated in a realm by an authentication scheme of at least this level or greater. If not supplied during Resrource Partner creation, this will default to 5.
Signout		
[SLOEnabled] WSFED_RP_SLO_ENABLED	Bool	Boolean indicating if Signout is enabled for the Resource Partner.
[SignOutCleanupURL] WSFED_RP_SIGNOUT_CLEANUP_URL	String	Sign-out cleanup URL of the Resource Partner. This property is mandatory if SLOEnabled is true.
[SignOutConfirmURL] WSFED_RP_SIGNOUT_CONFIRM_URL	String	URL where the user will be redirected once the Sign-out at Account Partner is complete. (If there are multiple Resource Partners available then Sign-out confirm URL of the last Resource Partner is applicable.)
Advanced		
[AssertionPluginClass] WSFED_RP_ PLUGIN_CLASS	String	The fully qualified Java class name for the Assertion Generator Plugin class to be used.

Property Name	Type	Description
[AssertionPluginParameters] WSFED_RP_ PLUGIN_PARAMS	String	The string containing parameters to be passed to the Assertion Generator Plugin.

Account Partner Properties

The following table lists the metadata properties used to define an Account Partner:

Property Name	Type	Description
General		
KEY_APID WSFED_KEY_APID	String	Identifier for the account partner. Among other things this identifier is used to identify assertion issuer. Also this is the key using which properties associated to a Account Partner can be looked up.
RPID WSFED_RPID	String	Identifier of the Resource Partner.
Signing		
[DisableSignatureProcessing] WSFED_DISABLE_SIGNATURE_ PROCESSING	Bool	Specifies whether signature processing is disabled. This setting is useful during initial setup of a Account Partner. When a provider is up and running, this setting will need to be set to false, to avoid security implications. Default value is zero.
[DsigVerInfoIssuerDN] WSFED _DSIG_VERINFO_ALIAS	String	Used to locate the certificate of the provider in the key store if it is not provided inline.

Property Name	Type	Description
Users		
[XPath] WSFED_AP_XPATH	String	XPath query for disambiguating the principal.
[LDAPSearchSpec] WSFED_AP_LDAP_SEARCH_SPEC	String	Search specification for LDAP directory.
[ODBCSearchSpec] WSFED_AP_ODBC_SEARCH_SPEC	String	Search specification for ODBC directory.
[WinNTSearchSpec] WSFED_AP_WINNT_SEARCH_SPEC	String	Search specification for WinNT directory.
[CustomSearchSpec] WSFED_AP_CUSTOM_SEARCH_SPEC	String	Search specification for a custom directory.
[ADSearchSpec] WSFED_AP_AD_SEARCH_SPEC	String	Search specification for AD directory.
SSO		
[RedirectMode] WSFED_AP_SSO_REDIRECT_MODE	Int	Redirect mode for assertion attributes. The following values are valid: 0—302 No Data 1—302 Cookie Data 2—Server Redirect 3—Persist Attributes The default is zero.
[SSODefaultService] WSFED_AP_SSO_DEFAULT_SERVICE	String	The default location of the Single Sign-on service.
[Target] WSFED_AP_SSO_TARGET	String	Target resource at the destination site.
[EnforceSingleUsePolicy] ENFORCE_SINGLE_USE_POLICY	Bool	If 1, the single use policy for POST assertions will be enforced, if 0, single use policy for POST assertions will not be enforced. Default set to 1.

Property Name	Type	Description
Signout		
[SLOEnabled] WSFED_AP_SLO_ENABLED	Bool	Boolean indicating if Signout is enabled for the Account Partner. If not supplied during Account Partner creation, this will default to disabled.
[SignOutURL] WSFED_AP_SIGNOUT_URL	String	Sign-out URL of the Account Partner. This property is mandatory if SLOEnabled is true.
Message Consumer Plug-in		
[APPluginClass] WSFED_AP_ PLUGIN_CLASS	String	Name of a Java class that implements customization of assertion consumption.
[APPluginParameters] WSFED_AP_ PLUGIN_PARAMS	String	Parameters of the Java class that implements customization of assertion consumption. All parameters are concatenated into one line.
Post Processing URL Support		
[UserNotFoundRedirectURL] WSFED_AP_USER_NOT_FOUND_ REDIRECT_URL	String	Contains an optional redirect URL to be used when - Auth Scheme cannot obtain a LoginID from the federation Message, given the configured query string - Auth Scheme can not find a user in the specific user directory, given the configured user store search string.

Property Name	Type	Description
[UserNotFoundRedirectMode] WSFED_AP_USER_NOT_FOUND_ REDIRECT_MODE	0/1	Default is 0. 0: Http 302 redirect without passing federation messages 1: Http Form Post Redirect
[FailureRedirectURL] WSFED_AP_FAILURE_REDIRECT_URL	String	Contains an optional redirect URL to be used when assertion processsing has failed.
[FailureRedirectMode] WSFED_AP_FAILURE_REDIRECT_MODE	0/1	Default is 0. 0: Http 302 redirect without passing federation messages 1: Http Form Post Redirect
[InvalidRedirectURL] WSFED_AP_INVALID_REDIRECT_URL	String	Contains an optional redirect URL to be used when the assertion is invalid.
[InvalidRedirectMode] WSFED_AP_INVALID_REDIRECT_MODE	0/1	Default is 0. 0: Http 302 redirect without passing federation messages 1: Http Form Post Redirect

Sm_PolicyApi_WSFEDResourcePartner_t

Defines WS-Federation Resource Partner data.

Syntax

```
typdef struct Sm_PolicyApi_WSFEDResourcePartner_s
{
        int iStructId;
        Sm_PolicyApi_WSFEDProviderProp_t* pProps;
        Sm_PolicyApi_WSFEDResourcePartner_t* next;
} Sm_PolicyApi_WSFEDResroucePartner_t;
```

Parameters

iStructId

ID of the structure in Sm_PolicyApi_Structs_t. Should be set to Sm_PolicyApi_WSFEDResourcePartner_ID.

pProp

Pointer to the linked list of Resource Partner properties.

next

Pointer to the next Resource Partner data in the linked list.

Exported Types

Administrator Rights

Sm_PolicyApi_AdminRights_t enumerates the rights of the administrator. These values may be used individually or combined to set multiple rights. The resulting value is passed to Sm_PolicyApi_AddAdmin() as one of the attributes in a Sm_PolicyApi_Admin_t structure.

Name	Value
Sm_PolicyApi_AdminRights_ManageAllDomains	0x01
Sm_PolicyApi_AdminRights_ManageObjects	0x02
Sm_PolicyApi_AdminRights_ManageUsers	0x04
Sm_PolicyApi_AdminRights_ManageKeys	0x08

Name	Value
Sm_PolicyApi_AdminRights_ManagePasswordPolicy	0x08
Sm_PolicyApi_AdminRights_ManageReports	0x10

The following table shows how these values are used to set administrative privileges:

Scope	Task	Setting and Privilege(s)
System	Manage System & Domain Objects	To set the privileges below, set administrator rights to both of the following: Sm_PolicyApi_AdminRights_ManageAllDomains Sm_PolicyApi_AdminRights_ManageObjects Privileges: Create/edit/delete agents, agent groups, directories, policy domains, authentication schemes, agent types, ODBC setup, directory mappings, certificate mappings, and registration schemes. Create/delete parent realms in all domains. Create/edit/delete administrators. Flush all caches, including cached resources. Change global settings. All the privileges for Manage Domain Objects listed below.
Domains	Manage Domain Objects	To set the privileges below, set administrator rights to: Sm_PolicyApi_AdminRights_ManageObjects Privileges: In managed domains: create/edit/delete rules, rule groups, responses, response groups, policies. Edit top level realms in managed domains (not resource filters). Create/edit/delete nested realms in managed domains. Flush specific realms from the resource cache, and flush all resources (in privileged domains) from the cache.
System	View Reports	To set the privilege below, set administrator rights to both of the following: Sm_PolicyApi_AdminRights_ManageAllDomains Sm_PolicyApi_AdminRights_ManageUsers Privilege: View all system and domain reports.

Scope	Task	Setting and Privilege(s)
Domains	View Reports	To set the privilege below, set administrator rights to: Sm_PolicyApi_AdminRights_ManageUsers Privilege: View reports for managed domains.
System	Manage Keys and Password Policies	To set the privileges below, set administrator rights to both of the following: Sm_PolicyApi_AdminRights_ManageAllDomains Sm_PolicyApi_AdminRights_ManageKeys Privileges: Create/edit/delete password policies. Manage keys.
Domains	Manage Password Policies	To set the privilege below, set administrator rights to: Sm_PolicyApi_AdminRights_ManagePasswordPolicy Privilege: Create/edit/delete password policies for users in directories attached to managed domains.
System	Manage Users	To set the privileges below, set administrator rights to both of the following: Sm_PolicyApi_AdminRights_ManageAllDomains Sm_PolicyApi_AdminRights_ManageReports Privileges: Flush all user session caches, or flush the user session cache of any individual user cache from any directory. Enable/disable users in any directory. Force password change on any user in any directory.
Domains	Manage Users	To set the privileges below, set administrator rights to: Sm_PolicyApi_AdminRights_ManageReports Privileges: Flush user session caches for individual users in directories attached to managed domains. Enable/disable users in directories attached to managed domains. Force password change on users in directories attached to managed domains.

Affiliate Attribute Types

Sm_PolicyApi_AffiliateAttrType_t enumerates the valid affiliate attribute types, for use in the affiliate functions to manipulate affiliate attributes.

Name	Value
Sm_PolicyApi_Affiliate_HTTP_Header_Variable	1
Sm_PolicyApi_Affiliate_HTTP_Cookie_Variable	2

Attribute Mode Types

Sm_PolicyApi_SAMLSPAttrMode_t enumerates the valid attribute retrieval types for use in SAML 2.0 Attribute Authority support:

Name	Value
Sm_PolicyApi_SAMLSP_SSO_Only	0
Sm_PolicyApi_SAMLSP_Attribute_Only	1

One of these values should be provided in the nMode element of the Sm_PolicyApi_SAMLSPAttr_t structure.

Authentication and Authorization Mapping Types

Sm_PolicyApi_AuthAzMapType_t enumerates the authentication and authorization mapping types.

Name	Value
Sm_PolicyApi_AuthAzMapType_DN	1
Sm_PolicyApi_AuthAzMapType_UniversalId	2
Sm_PolicyApi_AuthAzMapType_Attr	3

Certificate Mapping Attribute Types

Sm_PolicyApi_CertMapAttrType_t enumerates types of mapping that determine how an X.509 client certificate will map to the user information in the authentication directory.

Name	Value
Sm_PolicyApi_CertMapAttrType_Single	1
Sm_PolicyApi_CertMapAttrType_Custom	2
Sm_PolicyApi_CertMapAttrType_Exact	3

Certificate Mapping Directory Types

Sm_PolicyApi_DirType_t enumerates the types of directories that can be used to authenticate users.

Name	Value
Sm_PolicyApi_DirType_LDAP	1
Sm_PolicyApi_DirType_WinNT	2
Sm_PolicyApi_DirType_ODBC	3

Certificate Mapping Flags Definitions

Sm_PolicyApi_CertMapFlags_t enumerates flags that represent certificate mapping properties.

Flag	Value
Sm_PolicyApi_CertMapFlags_CertRequired Setting this flag causes SiteMinder to verify that the certificate presented by the user matches the certificate stored in the user's entry in the authentication directory. The authentication directory must be an LDAP user directory.	0x01

Flag	Value
Sm_PolicyApi_CertMapFlags_UseDistributionPoints Set this flag if your Certificate Revocation List (CRL) uses distribution points. 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. The distribution point provides a starting point for a CRL check and saves the processing time that it would take to search the entire CRL for a particular user. When this flag is set, SiteMinder retrieves the distribution point from the user's certificate, then uses it to find the appropriate LDAP directory entry point for the CRL.	0x02
Sm_PolicyApi_CertMapFlags_VerifySignature Set this flag to enable signature verification, where the Policy Server checks the Certificate Authority's public certificate against a signature stored in the policy database.	0x04
Sm_PolicyApi_CertMapFlags_CRLCheck Set this flag to make SiteMinder perform a Certificate Revocation List check. A Certificate Revocation List (CRL) is a list of revoked X.509 client certificates published by the Certificate Authority. Comparing certificates against CRLs 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.	0x08
Sm_PolicyApi_CertMapFlags_Cache Setting this flag causes SiteMinder to use cached CRL information until the date specified in the NextUpdate field in the CRL.	0x10

Directory Capabilities

Sm_PolicyApi_GetUserDirCapabilities() uses the values that are enumerated in Sm_DirectoryCapability_t, which is defined in SmApi.h.

Directory Capability	Value
Sm_DirCapability_CreatePasswordPolicy Capable of creating password policy. The following attributes are affected in the user directory (Sm_PolicyApi_UserDir_t): *pszPasswordData*, *pszDisabledAttr*, and *pszPasswordAttribute*.	0x00000001

Directory Capability	Value
Sm_DirCapability_CreateRegistrationPolicy Capable of creating registration policy. The following attributes are affected in the user directory (Sm_PolicyApi_UserDir_t): *pszAnonymousId*, *pszEmailAddressAttr*, *pszChallengeRespAttr*, and *pszPasswordAttribute*.	0x00000002
Sm_DirCapability_ResetUserPassword Capable of resetting the user password. This affects *pszPasswordAttribute*.	0x00000004
Sm_DirCapability_ChangeUserPassword Capable of changing the user password. This affects *pszPasswordAttribute*.	0x00000008
Sm_DirCapability_DisableUser Capable of disabling the user account. This affects *pszDisabledAttr*.	0x00000010
Sm_DirCapability_DmsCapable Capable of being written by the Delegated Management System (DMS).	0x00000020
Sm_DirCapability_Recursive Capable of supporting recursion.	0x00000040
Sm_DirCapability_DisabledAttr Read-Write disabled attribute. This attribute is configured for the user directory.	0x00100000
Sm_DirCapability_UniversalIdAttr Read-only Universal ID. This attribute is configured for the user directory.	0x00200000
Sm_DirCapability_AnonymousIdAttr Read-Write anonymous ID attribute. This attribute is configured for the user directory.	0x00400000
Sm_DirCapability_PasswordDataAttr Read-Write password data attribute. This attribute is configured for the user directory.	0x00800000
Sm_DirCapability_UserPasswordAttr Read-Write password attribute. This attribute is configured for the user directory.	0x01000000

Directory Capability	Value
Sm_DirCapability_EmailAddressAttr Read-only E-mail attribute. This attribute is configured for the user directory.	0x02000000
Sm_DirCapability_ChallengeRespAttr Read-Write Challenge and Response attribute. This attribute is configured for the user directory.	0x04000000

Note: Attribute masks are directory user profile attributes. They are available in the directory. Each attribute is read-only or read-write. Read-write attributes are not used by other applications.

Domain Flags

Sm_PolicyApi_DomainFlags_t enumerates flags pertaining to domain-wide influence.

Name	Value
Sm_PolicyApi_DomainFlags_GlobalPoliciesApply When this flag is set, the domain processes global policies for all realms in the domain. When this flag is not set, the domain does not process global policies.	0x02

Group Types

Sm_PolicyApi_Groups_t enumerates the type of group for which you can perform group functions.

Name	Value
Sm_PolicyApi_NULL_Group_Prop	0
Sm_PolicyApi_Rule_Group_Prop	1
Sm_PolicyApi_Response_Group_Prop	2
Sm_PolicyApi_Agent_Group_Prop	3

IP Address Types

Sm_PolicyApi_IPAddressType_t enumerates the type of IP address restrictions that are defined for an object in Sm_PolicyApi_IPAddress_t.

IP Address Type	Value
Sm_PolicyApi_IPAddressType_SingleHost A single host IP address requires the following fields to be set: ■ *iStructId*. IP Address data structure ID defined in Sm_PolicyApi_Structs_t. ■ *iIPAddressType*. Set IP address type to be Sm_PolicyApi_IPAddressType_SingleHost. ■ *nIPAddress*. The valid IP address. This IP address is specified in the long format.	1
Sm_PolicyApi_IPAddressType_HostName A host name IP address requires the following fields to be set: ■ *iStructId*. IP Address data structure ID defined in Sm_PolicyApi_Structs_t. ■ *iIPAddressType*. Set IP address type to be Sm_PolicyApi_IPAddressType_HostName. ■ *pszHostName*[BFSIZE]. Host name of the machine that a user must be using for an action to occur-for example, for a policy to fire.	2
Sm_PolicyApi_IPAddressType_AddressAndSubNetMask A subnet mask requires the following fields to be set: ■ *iStructId*. IP Address data structure ID defined in Sm_PolicyApi_Structs_t. ■ *iIPAddressType*. Set IP address type to be Sm_PolicyApi_IPAddressType_AddressAndSubnetMask. ■ *nIPAddress*. The valid IP address. This IP address is specified in the long format. ■ *nSubnetMask*. Specify the subnet mask.	3

IP Address Type	Value
Sm_PolicyApi_IPAddressType_Range A range of IP addresses requires the following fields to be set: ■ *iStructId*. IP Address data structure ID defined in Sm_PolicyApi_Structs_t. ■ *iIPAddressType*. Set IP address type to be Sm_PolicyApi_IPAddressType_Range. ■ *nIPAddress*. Starting IP address. This IP address is specified in the long format. ■ *nEndIPAddress*. Ending IP address. This IP address is specified in the long format.	4

Management Commands

Sm_PolicyApi_ManagementCommands_t enumerates the values that can be passed to Sm_PolicyApi_ManagementCommand() for flushing caches, for managing agent encryption keys, and for shared secret rollover.

Initialize the structure to zero (memset) prior to setting any values. Use the symbolic enumerated values, rather than hard-coding integer command values.

The value is passed in the *iCommand* field of the structure Sm_PolicyApi_ManagementCommand_t.

Management Command	Value
Sm_PolicyApi_ManagementCommand_FlushAll Flushes all SiteMinder caches. Policy store cache, resource cache, and user information cache are flushed by this command. It does not require any data in the *pszData* field of Sm_PolicyApi_ManagementCommand_t.	1
Sm_PolicyApi_ManagementCommand_FlushUsers Flushes user information cache. It does not require any data in the *pszData* field of Sm_PolicyApi_ManagementCommand_t.	2
Sm_PolicyApi_ManagementCommand_FlushRealms Flushes resource cache. It does not require any data in the *pszData* field of Sm_PolicyApi_ManagementCommand_t.	3

Management Command	Value
Sm_PolicyApi_ManagementCommand_ChangeDynamicKeys Changes the dynamic agent key. It does not require any data in the *pszData* field of Sm_PolicyApi_ManagementCommand_t. Before you change a dynamic agent key through the C API, the Agent Key setting in the Policy Server Key Management dialog box must be set to Use dynamic Agent Key. To access this dialog box in the Policy Server UI, click Tools > Manage Keys. Then, in the Agent Key tab, select Use dynamic Agent Key.	4
Sm_PolicyApi_ManagementCommand_ChangePersistentKey Changes the persistent or static key. The data field *pszData* of Sm_PolicyApi_ManagementCommand_t structure may contain an optional key value. If *pszData* is empty, the persistent key is randomly generated.	5
Sm_PolicyApi_ManagementCommand_ChangeSessionKey Changes the session key. The data field *pszData* of Sm_PolicyApi_ManagementCommand_t structure may contain an optional key value. If *pszData* is empty, the session key is randomly generated.	6
Sm_PolicyApi_ManagementCommand_RolloverSharedSecrets Rolls over shared secrets for rollover-enabled trusted hosts.	7

Password Messages

Sm_PolicyApi_PasswordMsgId_t enumerates password message IDs.

Password messages describe the encoded error message returned to Sm_PolicyApi_SetPassword() when a new password does not satisfy the password policy requirements of the specified directory.

Password Message ID	Value
Sm_PolicyApi_PasswordMsgId_None	0
Sm_PolicyApi_PasswordMsgId_ChangePassword	1
Sm_PolicyApi_PasswordMsgId_PassswordGeneralFailure	1000
Sm_PolicyApi_PasswordMsgId_PasswordShort	1001
Sm_PolicyApi_PasswordMsgId_PasswordLong	1002
Sm_PolicyApi_PasswordMsgId_PasswordOldPasswordBad	1003
Sm_PolicyApi_PasswordMsgId_PasswordReuse	1004

Password Message ID	Value
Sm_PolicyApi_PasswordMsgId_PasswordSimilar	1005
Sm_PolicyApi_PasswordMsgId_PasswordRepeatingChars?	1006
Sm_PolicyApi_PasswordMsgId_PasswordDictionaryMatch	1007
Sm_PolicyApi_PasswordMsgId_PasswordContentLetters	1008
Sm_PolicyApi_PasswordMsgId_PasswordContentDigits	1009
Sm_PolicyApi_PasswordMsgId_PasswordContentAlphaNum	1010
Sm_PolicyApi_PasswordMsgId_PasswordContentPunctuation	1011
Sm_PolicyApi_PasswordMsgId_PasswordContentNonPrintable	1012
Sm_PolicyApi_PasswordMsgId_PasswordContentNonAlphaNum	1013
Sm_PolicyApi_PasswordMsgId_PasswordProfileMatch	1014
Sm_PolicyApi_PasswordMsgId_PasswordGraceDays	1015
Sm_PolicyApi_PasswordMsgId_PasswordSystemPIN	1016
Sm_PolicyApi_PasswordMsgId_PasswordUserMaxNumPIN	1017
Sm_PolicyApi_PasswordMsgId_PasswordUserMinMaxNumPIN	1018
Sm_PolicyApi_PasswordMsgId_PasswordUserMaxAlphaPIN	1019
Sm_PolicyApi_PasswordMsgId_PasswordUserMinMaxAlphaPIN	1020
Sm_PolicyApi_PasswordMsgId_PasswordAcceptPIN	1021
Sm_PolicyApi_PasswordMsgId_PasswordContentLowerAlpha	1022
Sm_PolicyApi_PasswordMsgId_PasswordContentUpperAlpha	1023
Sm_PolicyApi_PasswordMsgId_PasswordContentNoLowerAlpha	1024
Sm_PolicyApi_PasswordMsgId_PasswordContentNoUpperAlpha	1025
Sm_PolicyApi_PasswordMsgId_PasswordContentNoDigits	1026
Sm_PolicyApi_PasswordMsgId_PasswordContentNoPunctuation	1027
Sm_PolicyApi_PasswordMsgId_PasswordContentNoNonPrintable	1028
Sm_PolicyApi_PasswordMsgId_PasswordContentNoNonAlphaNum	1029
Sm_PolicyApi_PasswordMsgId_PasswordContentNoAlphaNum	1030
Sm_PolicyApi_PasswordMsgId_PasswordContentMatchRegExp	1031
Sm_PolicyApi_PasswordMsgId_PasswordContentNoMatchRegExp	1032
Sm_PolicyApi_PasswordMsgId_PasswordUserMinNumPIN	1033

Password Message ID	Value
Sm_PolicyApi_PasswordMsgId_PasswordUserDigitsPIN	1034
Sm_PolicyApi_PasswordMsgId_PasswordUserAlphaNumPIN	1035

Additional information about the error message is available in the password message field associated with the password message.

Password Message Fields

Sm_PolicyApi_PasswordMsgFieldId_t enumerates password message field IDs.

Password message fields contain additional information about the password messages described in the previous section. You can find this additional information in the structure Sm_PolicyApi_PasswordMsgField_t.

Password Message Field ID	Value
Sm_PolicyApi_PasswordMsgFieldId_None	0
Sm_PolicyApi_PasswordMsgFieldId_Min	1
Sm_PolicyApi_PasswordMsgFieldId_Max	2
Sm_PolicyApi_PasswordMsgFieldId_OldPW	3
Sm_PolicyApi_PasswordMsgFieldId_NewPW	4
Sm_PolicyApi_PasswordMsgFieldId_Days	5
Sm_PolicyApi_PasswordMsgFieldId_Token	6

Fields can be of type integer or string, or they can have no type.

Password Message Field Types

Sm_PolicyApi_FieldType_t enumerates the possible data types for the password message fields.

Password Message Field Type	Value
Sm_PolicyApi_FieldType_None	0
Sm_PolicyApi_FieldType_Int	1
Sm_PolicyApi_FieldType_String	2

Password Policy Behavior Flags

Sm_PasswordPolicyBehavior_t enumerates the behavioral characteristics of a password policy.

Password Policy Behavior Flag	Value
Sm_PasswordPolicy_DontTrackLogins This flag has been replaced in SiteMinder v6.0 SP3 by: ■ Sm_PasswordPolicy_DontTrackSuccessLogins ■ Sm_PasswordPolicy_DontTrackFailedLogins The new flags allow successful and failed logins to be tracked separately. Sm_PasswordPolicy_DontTrackLogins is currently maintained for backwards compatibility. If this flag is set, login tracking for successful and failed logins will not occur.	0x00000004
Sm_PasswordPolicy_AllowFailedWrites Allows users to log in even if password data cannot be written to the user directory.	0x00000008
Sm_PasswordPolicy_InactivityForcePWChange Forces a password change on the next login attempt after a user's password becomes invalid due to inactivity.	0x00000010
Sm_PasswordPolicy_PWExpiredForcePWChange Forces a password change on the next login attempt after a user's password expires.	0x00000020
Sm_PasswordPolicyBehavior_FullReenable If a user's account is disabled due to successive incorrect password entries, this flag re-enables the account after a given time period. Specify the time in the *nReenablement* field of Sm_PolicyApi_PasswordPolicy_t. If this flag is not set, the user is allowed another login attempt after the given *nReenablement* time period.	0x00000040
Sm_PasswordPolicy_StopPriorityChaining Prevents the evaluation of password policies with lower priority ratings than the current password policy.	0x00000080
Sm_PasswordPolicy_ExpireDisablePassword When the password expires, disable just the password and not the user account.	0x00000100

Password Policy Behavior Flag	Value
Sm_PasswordPolicy_FailuresDisablePassword When the maximum number of authentication failures are exceeded, disable just the password and not the user account.	0x00000200
Sm_PasswordPolicy_ForceCase Force the password's case that is specified through bit Sm_PasswordPolicy_CaseSelect.	0x00000400
Sm_PasswordPolicy_CaseSelect If Sm_PasswordPolicy_ForceCase is set, Sm_PasswordPolicy_ForceCase forces upper case passwords when set, and forces lower case passwords when cleared.	0x00000800
Sm_PasswordPolicy_CaseBits Sets both of the following bits (forces upper case passwords): ■ Sm_PasswordPolicy_ForceCase ■ Sm_PasswordPolicy_CaseSelect	0x00000c00
Sm_PasswordPolicy_StripLeadingWhiteSpace Removes any leading white space from the password.	0x00001000
Sm_PasswordPolicy_StripTrailingWhiteSpace Removes any trailing white space from the password.	0x00002000
Sm_PasswordPolicy_StripFlankingWhiteSpace Sets both of the following bits (strips leading and trailing white space): ■ Sm_PasswordPolicy_StripLeadingWhiteSpace ■ Sm_PasswordPolicy_StripTrailingWhiteSpace	0x00003000
Sm_PasswordPolicy_StripEmbeddedWhiteSpace Removes all white space within the password.	0x00004000
Sm_PasswordPolicy_WhiteSpaceBits Sets all of the following bits (strips leading, trailing, and embedded white space): ■ Sm_PasswordPolicy_StripLeadingWhiteSpace ■ Sm_PasswordPolicy_StripTrailingWhiteSpace ■ Sm_PasswordPolicy_StripEmbeddedWhiteSpace	0x00007000

Password Policy Behavior Flag	Value
Sm_PasswordPolicy_PreProcessBits Sets all of the following bits (forces upper case passwords and strips leading, trailing, and embedded white space): ■ Sm_PasswordPolicy_ForceCase ■ Sm_PasswordPolicy_CaseSelect ■ Sm_PasswordPolicy_StripLeadingWhiteSpace ■ Sm_PasswordPolicy_StripTrailingWhiteSpace ■ Sm_PasswordPolicy_StripEmbeddedWhiteSpace	0x00007c00
Sm_PasswordPolicy_DontTrackSuccessLogins Performs directory updates at login time. When this flag is not set, the password policy tracks successful user logins, including the time of the last login.	0x00008000
Sm_PasswordPolicy_DontTrackFailedLogins Performs directory updates at login time. When this flag is not set, the password policy tracks unsuccessful user login attempts.	0x00010000

Note: Values 0x00000400 through 0x00007c00 apply to password preprocessing. During preprocessing, the password is checked before it is processed or stored.

Policy Flags

Sm_PolicyApi_AddUsersToPolicy() uses the following values (which are defined in SmApi.h):

Flag	Value
Sm_PolicyBehavior_Exclude_Mask Bit 0x01 determines whether user policy excludes or includes 'users.'	0x01
Sm_PolicyBehavior_Exclude_No	0x00
Sm_PolicyBehavior_Exclude_Yes	0x01
Sm_PolicyBehavior_Recursive_Mask Bit 0x02 determines whether user policy is recursive. This is applicable to directory object classes that can be nested.	0x02
Sm_PolicyBehavior_Recursive_No	0x00

Flag	Value
Sm_PolicyBehavior_Recursive_Yes	0x02
Sm_PolicyBehavior_AND_Mask Bit 0x04 determines whether the user policy has an AND relationship between user policies. This is applicable to user policies that are members of a particular user directory within the policy.	0x04
Sm_PolicyBehavior_AND_No	0x00
Sm_PolicyBehavior_AND_Yes	0x04

Policy Management API Initialization Flags

Sm_PolicyApi_InitFlags_t enumerates the initialization flags used by Sm_PolicyApi_Init(). These flags affect API behavior.

Flag	Value
Sm_PolicyApi_InitFlags_EnableCache Enables caching of policy store, resource, and user information to ensure that SiteMinder responds quickly to user requests.	0x01
Sm_PolicyApi_InitFlags_PreLoadCache Enables the Policy Management API to preload the SiteMinder caches. **Note:** By omitting this flag, you can reduce the time it takes for custom Policy Management applications to make policy store changes.	0x02
Sm_PolicyApi_InitFlags_LoadAgentTypeDictionary Enables the Policy Management API to preload the SiteMinder agent type dictionary.	0x04
Sm_PolicyApi_InitFlags_DisableValidation Disables validation of policy objects.	0x08
Sm_PolicyApi_InitFlags_DisableAudit Disables: ■ Auditing of user activity, including authentication, authorization, and administration activities. (Administration activities include changes to the policy store.) ■ Monitoring of user sessions.	0x10

Flag	Value
Sm_PolicyApi_InitFlags_DisableCacheUpdates Disables cache updates. If cache updates are not disabled and Sm_PolicyApi_InitFlags_EnableCache is turned off, the Policy Management API will still issue the cache updates.	0x20
Sm_PolicyApi_InitFlags_DisableManagementWatchDog Disables the SiteMinder management watchdog. The watchdog is enabled by default. The watchdog is used internally and should not be disabled.	0x40

Policy Object IDs

Sm_PolicyApi_Objects_t describes the policy store properties that can be retrieved, set, and removed.

Note: Sm_PolicyApi_NULL_Domain_Props, value 0, is reserved.

The following table lists the domain object type values that can be passed to Sm_PolicyApi_GetDomainObjects():

Name	Value
Sm_PolicyApi_Rule_Prop	1
Sm_PolicyApi_RuleGroup_Prop	2
Sm_PolicyApi_Policy_Prop	3
Sm_PolicyApi_PolicyLink_Prop	4
Sm_PolicyApi_UserPolicy_Prop	5
Sm_PolicyApi_Realm_Prop	6
Sm_PolicyApi_ResponseGroup_Prop	7
Sm_PolicyApi_Response_Prop	8
Sm_PolicyApi_ResponseAttr_Prop	9
Sm_PolicyApi_UserDir_Prop	10
Sm_PolicyApi_Admins_Prop	17
Sm_PolicyApi_ActiveExpr_Prop	23
Sm_PolicyApi_Variable_Prop	25
Sm_PolicyApi_Affiliate_Prop	33

Name	Value
Sm_PolicyApi_SAMLSP_Prop	35

The following table lists the global object type names that can be passed to Sm_PolicyApi_GetGlobalObjects():

Name	Value
Sm_PolicyApi_Rule_Prop	1
Sm_PolicyApi_Policy_Prop	3
Sm_PolicyApi_Response_Prop	8
Sm_PolicyApi_UserDir_Prop	10
Sm_PolicyApi_Scheme_Prop Object ID for an authentication scheme.	11
Sm_PolicyApi_Agent_Prop	12
Sm_PolicyApi_AgentGroup_Prop	13
Sm_PolicyApi_AgentType_Prop	14
Sm_PolicyApi_AgentTypeAttr_Prop	15
Sm_PolicyApi_Domain_Prop	16
Sm_PolicyApi_Admins_Prop	17
Sm_PolicyApi_ODBCQueryScheme_Prop	18
Sm_PolicyApi_RegistrationScheme_Prop	19
Sm_PolicyApi_PasswordPolicy_Prop	20
Sm_PolicyApi_AuthAzMap_Prop Object ID for an authentication-authorization object.	21
Sm_PolicyApi_CertMap_Prop Object ID for a certification-mapping object.	22
Sm_PolicyApi_VariableType_Prop	24
Sm_PolicyApi_TrustedHost_Prop	26
Sm_PolicyApi_HostConfig_Prop	27
Sm_PolicyApi_AgentConfig_Prop	28
Sm_PolicyApi_Association_Prop Object ID for a configuration name/value pair in an agent configuration object.	29

Name	Value
Sm_PolicyApi_AffiliateDomain_Prop	32
Sm_PolicyApi_SharedSecretPolicy_Prop	34
Sm_PolicyApi_SAMLIdP_Prop	36
Sm_PolicyApi_SAMLAffiliation_Prop	37
Sm_PolicyApi_WSFEDResourcePartner_Prop	38

Policy Resolutions

Sm_PolicyResolution_t, defined in SmApi.h, enumerates the values that describe the relationship between two policy objects.

More Information:

Sm_PolicyResolution_t (see page 757)

Return Codes

The value codes that can be returned by the API are enumerated in Sm_PolicyApi_Status_t. The values have the following significance:

- A zero return code indicates success.
- Negative return codes indicate failure.

Most of the code names are self-explanatory. However, note that Sm_PolicyApi_BadArgument (-10) is returned when one or more of the required input parameters is not supplied. For example, if an argument such as a domain OID is null or represents a string of zero length, Sm_PolicyApi_BadArgument is returned to the caller.

Return codes with values less than -100 (except for Sm_PolicyApi_NotUnique, value -105) will rarely be returned by this API. They are included for completeness.

Return Code	Value
Sm_PolicyApi_Success	0
Sm_PolicyApi_Failure	-1
Sm_PolicyApi_InvalidHandle	-2

Sm_PolicyApi_ErrorLogin	-3
Sm_PolicyApi_NoPrivilege	-4
Sm_PolicyApi_InvalidPasswordSyntax	-5
Sm_PolicyApi_InvalidPassword	-6
Sm_PolicyApi_DuplicateEntry	-7
Sm_PolicyApi_DoesNotExist	-8
Sm_PolicyApi_NotFound	-9
Sm_PolicyApi_BadArgument	-10
Sm_PolicyApi_WrongNumberOfElements	-11
Sm_PolicyApi_UserDirNotPartOfDomain	-12
Sm_PolicyApi_UserDirNotValid	-13
Sm_PolicyApi_ErrorUserDir	-14
Sm_PolicyApi_AgentNotFound	-15
Sm_PolicyApi_AgentTypeNotFound	-16
Sm_PolicyApi_AgentTypeAttrNotFound	-17
Sm_PolicyApi_AgentTypeMismatch	-18
Sm_PolicyApi_ODBCQuerySchemeNotFound	-19
Sm_PolicyApi_UserDirNotFound	-20
Sm_PolicyApi_DomainNotFound	-21
Sm_PolicyApi_AdminNotFound	-22
Sm_PolicyApi_SchemeNotFound	-23
Sm_PolicyApi_RegistrationSchemeNotFound	-24
Sm_PolicyApi_PasswordPolicyNotFound	-25
Sm_PolicyApi_SchemeIsRequired	-26
Sm_PolicyApi_PasswordPolicyConfig	-27
Sm_PolicyApi_RealmNotFound	-28
Sm_PolicyApi_NoChildren	-29
Sm_PolicyApi_RuleNotFound	-30
Sm_PolicyApi_ResponseNotFound	-31
Sm_PolicyApi_ResponseAttrNotFound	-32
Sm_PolicyApi_PolicyNotFound	-33

Sm_PolicyApi_PolicyLinkNotFound	-34
Sm_PolicyApi_UserPolicyNotFound	-35
Sm_PolicyApi_BadGroup	-36
Sm_PolicyApi_GroupNotFound	-37
Sm_PolicyApi_Invalid	-38
Sm_PolicyApi_InvalidHandleVersion	-39
Sm_PolicyApi_DomainNotAffiliate	-41
Sm_PolicyApi_InvalidOid	-100
Sm_PolicyApi_NotImplemented	-101
Sm_PolicyApi_NotSearchable	-102
Sm_PolicyApi_NotStorable	-103
Sm_PolicyApi_NotCollection	-104
Sm_PolicyApi_NotUnique	-105
Sm_PolicyApi_InvalidProp	-106
Sm_PolicyApi_NotInitted	-107
Sm_PolicyApi_NoSession	-108
Sm_PolicyApi_OidInUseByRealm	-109
Sm_PolicyApi_OidInUseByRule	-110
Sm_PolicyApi_OidInUseByAdmin	-111
Sm_PolicyApi_MissingProperty	-112
Sm_PolicyApi_GroupMemberName	-113
Sm_PolicyApi_RadiusIpAddrNotUnique	-114
Sm_PolicyApi_GroupAgentType	-115
Sm_PolicyApi_RadiusRealmNotUnique	-116
Sm_PolicyApi_RealmFilterNotUnique	-117
Sm_PolicyApi_InvalidCharacters	-118
Sm_PolicyApi_AgentTypeCantBeDeleted	-119
Sm_PolicyApi_ProvNotImplemented	-120
Sm_PolicyApi_ProvNotUnique	-121
Sm_PolicyApi_RealmCantBeUsedInRule	-122
Sm_PolicyApi_OidInUserByCertMap	-123

Sm_PolicyApi_OidInUseBySelfReg	-124
Sm_PolicyApi_OidInUseByUserDirectory	-125
Sm_PolicyApi_SchemeCantBeDeleted	-126
Sm_PolicyApi_BasicSchemeUpdate	-127
Sm_PolicyApi_NonHtmlForm	-128
Sm_PolicyApi_IllegalRealmOperation	-129
Sm_PolicyApi_NameNotUnique	-130
Sm_PolicyApi_FeatureNotSupported	-132
Sm_PolicyApi_AssertionConsumerDefaultMissing	-133
Sm_PolicyApi_SAMLSP_AuthenticationURLMissing	-134
Sm_PolicyApi_SAMLSP_DomainOidMissing	-135
Sm_PolicyApi_SAMLSP_IdPIDMissing	-136
Sm_PolicyApi_SAMLSP_NameMissing	-137
Sm_PolicyApi_SAMLSP_NameIdFormatMissing	-138
Sm_PolicyApi_SAMLSP_NameIdTypeMissing	-139
Sm_PolicyApi_SAMLSP_NameIdStaticMissing	-140
Sm_PolicyApi_SAMLSP_NameIdAttrNameMissing	-141
Sm_PolicyApi_SAMLSP_NameIdDNSpecMissing	-142
Sm_PolicyApi_SAMLSP_ProviderIDMissing	-143
Sm_PolicyApi_SAMLSP_ProviderIDNotUnique	-144
Sm_PolicyApi_SAML_UnSupportedSAMLVersion	-145
Sm_PolicyApi_SAMLIDP_IncorrectParameters	-146
Sm_PolicyApi_SAMLIDP_ProviderIDNotUnique	-147
Sm_PolicyApi_SAMLAFF_NameMissing	-148
Sm_PolicyApi_SAMLAFF_NameIdFormatMissing	-149
Sm_PolicyApi_SAMLAFF_NameIdTypeMissing	-150
Sm_PolicyApi_SAMLAFF_NameIdStaticMissing	-151
Sm_PolicyApi_SAMLAFF_NameIdAttrNameMissing	-152
Sm_PolicyApi_SAMLAFF_NameIdDNSpecMissing	-153
Sm_PolicyApi_SAMLAFF_AffiliationIDMissing	-154
Sm_PolicyApi_SAMLAFF_AffiliationIDNotUnique	-155

Sm_PolicyApi_SAMLAFF_AffiliationHasMembers	-156
Sm_PolicyApi_SAML_UnknownProperty	-157
Sm_PolicyApi_WSFEDRP_AssertionConsumerDefaultMissing	-158
Sm_PolicyApi_WSFEDRP_AuthenticationURLMissing	-159
Sm_PolicyApi_WSFEDRP_DomainOidMissing	-160
Sm_PolicyApi_WSFEDRP_APIDMissing	-161
Sm_PolicyApi_WSFEDRP_NameMissing	-162
Sm_PolicyApi_WSFEDRP_NameIdFormatMissing	-163
Sm_PolicyApi_WSFEDRP_NameIdTypeMissing	-164
Sm_PolicyApi_WSFEDRP_NameIdStaticMissing	-165
Sm_PolicyApi_WSFEDRP_NameIdAttrNameMissing	-166
Sm_PolicyApi_WSFEDRP_NameIdDNSpecMissing	-167
Sm_PolicyApi_WSFEDRP_ProviderIdMissing	-168
Sm_PolicyApi_WSFEDRP_ProviderIdNotUnique	-169
Sm_PolicyApi_WSFEDRP_UnsupportedSAMLVersion	-170
Sm_PolicyApi_WSFEDRP_UnkownProperty	-171
Sm_PolicyApi_WSFEDAP_IncorrectParameters	-172
Sm_PolicyApi_WSFEDAP_ProviderIDNotUnique	-173
Sm_PolicyAPI_InsufficientRPData	-174
Sm_PolicyAPI_WSFED_UnSupportedWSFEDVersion	-175
Sm_PolicyAPI_DuplicateAttribute	-176
Sm_PolicyAPI_SAMLSP_ACSDuplicateIndex	-177
Sm_PolicyAPI_SAMLSP_ACSIndexedEndpointInUse	-178
Sm_PolicyAPI_SAMLSP_ACSIndexedEndpointNotFound	-179
Sm_PolicyAPI_SAMLSP_CantDeleteDefaultACSIndex	-180
Sm_PolicyAPI_SAMLSP_ACSMaxExceeded	-181
Sm_PolicyAPI_InConsistentANDBitMask	-182

SAML1x Redirect URL Types

Sm_PolicyApi_SAML1_STATUS_REDIRECT_URL_TYPE_t defines the type of redirection specified in Sm_PolicyApi_AddRedirectURLToSAML1xScheme() and Sm_PolicyApi_GetRedirectURLFromSAML1xScheme().

Sm_PolicyApi_SAML1_STATUS_REDIRECT_URL_TYPE_t is listed in SmPolicyApi45.h.

Name	Value
Sm_PolicyApi_SAML1_STATUS_REDIRECT_URL_USER_NOT_FOUND_ TYPE	0
Sm_PolicyApi_SAML1_STATUS_REDIRECT_URL_INVALID_SSO	1
Sm_PolicyApi_SAML1_STATUS_REDIRECT_URL_UNACCEPTABLE_USE R_ CREDENTIALS	2

SAML Assertion Consumer Service Bindings

The following values are the SAML Protocol Bindings that can be specified for each row of the Assertion Consumer Service:

Name	Value
Sm_PolicyApi_SAMLSP_HTTP_Post	0
Sm_PolicyApi_SAMLSP_HTTP_Artifact	1
Sm_PolicyApi_SAMLSP_PAOS	2

SAML Attribute Name Format Identifiers

Sm_PolicyApi_SAMLSPAttrNameFormat_t defines the format to use for specifying attributes that apply to a principal. The format specification is made within the structure Sm_PolicyApi_SAMLSPAttr_t.

The format identifiers are defined by the SAML 2.0 standard.

Sm_PolicyApi_SAMLSPAttrNameFormat_t is listed in SmPolicyApi45.h.

Name	Value
Sm_PolicyApi_SAMLSP_Unspecified	0
Sm_PolicyApi_SAMLSP_URI	1
Sm_PolicyApi_SAMLSP_Basic	2

SAML Profiles

Sm_PolicyApi_SAML_Profile_t specifies the communication profile used to send and receive a SAML assertion for a particular affiliate object. The profile is specified as one of the attributes of a Sm_PolicyApi_Affiliate_t structure. Sm_PolicyApi_SAML_Profile_t is listed in SmPolicyApi45.h.

Name	Value
Sm_PolicyApi_SAML_Profile_Artifact	1
Sm_PolicyApi_SAML_Profile_POST	2

Scheme Types

Sm_Api_SchemeType_t describes the values that may be passed to Sm_PolicyApi_AddScheme() as one of the attributes of a SmPolicyApi_Scheme_t structure. Sm_Api_SchemeType_t is listed in SmApi.h.

Scheme Type	Value
Sm_Api_SchemeType_Basic	1
Sm_Api_SchemeType_CryptoCard	2
Sm_Api_SchemeType_Encotone	3
Sm_Api_SchemeType_HTMLForm	4

Sm_Api_SchemeType_BasicOverSSL	5
Sm_Api_SchemeType_RadiusServer	6
Sm_Api_SchemeType_SafeWordServer	7
Sm_Api_SchemeType_ACEServer	8
Sm_Api_SchemeType_X509ClientCert	9
Sm_Api_SchemeType_X509ClientCertAndBasic	10
Sm_Api_SchemeType_X509ClientCertOrBasic	11
Sm_Api_SchemeType_RadiusChapPap	12
Sm_Api_SchemeType_Anonymous	13
Sm_Api_SchemeType_NTLM	14
Sm_Api_SchemeType_Custom	15
Sm_Api_SchemeType_ACEServerHTMLForm	16
Sm_Api_SchemeType_SafeWordHTMLForm	17
Sm_Api_SchemeType_XMLDsig	18
Sm_Api_SchemeType_X509ClientCertOrForm	19
Sm_Api_SchemeType_X509ClientCertAndForm	20
Sm_Api_SchemeType_MSPassport	21
Sm_Api_SchemeType_XMLDocumentCredentialCollector	22
Sm_Api_SchemeType_SAMLSessionTicket	25
Sm_Api_SchemeType_SAMLArtifact	26
Sm_Api_SchemeType_Impersonation	27
Sm_Api_SchemeType_SAMLPOST	28
Sm_Api_SchemeType_SAML2	29
Sm_Api-SchemeType_WSFED	30

Shared Secret Rollover Parameters

Sm_PolicyApi_SecretRolloverPeriod_t enumerates the units of time which, when combined with the rollover frequency setting, determines how often shared secret rollover occurs. For example a rollover period of RolloverHOURS and a frequency of 12 means that the shared secret is changed every 12 hours.

The rollover period is defined in the *iRolloverPeriod* field of structure Sm_PolicyApi_SharedSecretPolicy_t, and the frequency is defined in the *iRolloverFrequency* field of the structure.

Name	Value
RolloverNEVER	0
RolloverHOURS	1
RolloverDAYS	2
RolloverWEEKS	3
RolloverMONTHS	4

Structure IDs

Sm_PolicyApi_Structs_t enumerates the data structures that can be passed to and from the Policy Management API as follows:

Name	Value
Sm_PolicyApi_NULL_ID	0
Sm_PolicyApi_Rule_ID	1
Sm_PolicyApi_Policy_ID	2
Sm_PolicyApi_Realm_ID	3
Sm_PolicyApi_Response_ID	4
Sm_PolicyApi_UserDir_ID	5
Sm_PolicyApi_Agent_ID	6
Sm_PolicyApi_Domain_ID	7
Sm_PolicyApi_PolicyLink_ID	8
Sm_PolicyApi_ResponseAttr_ID	9
Sm_PolicyApi_User_ID	10

Name	Value
Sm_PolicyApi_Scheme_ID	11
Sm_PolicyApi_Admin_ID	12
Sm_PolicyApi_Group_ID	13
Sm_PolicyApi_ODBCQueryScheme_ID	14
Sm_PolicyApi_Object_ID	15
Sm_PolicyApi_AgentType_ID	16
Sm_PolicyApi_AgentTypeAttr_ID	17
Sm_PolicyApi_RegistrationScheme_ID	18
Sm_PolicyApi_PasswordPolicy_ID	19
Sm_PolicyApi_IPAddress_ID	20
Sm_PolicyApi_AuthAzMap_ID	21
Sm_PolicyApi_CertMap_ID	22
Sm_PolicyApi_PasswordMsgField_ID	23
Sm_PolicyApi_VariableType_ID	25
Sm_PolicyApi_Variable_ID	26
Sm_PolicyApi_TrustedHost_ID	27
Sm_PolicyApi_HostConfig_ID	28
Sm_PolicyApi_AgentConfig_ID	29
Sm_PolicyApi_Association_ID	30
Sm_PolicyApi_UserContext_ID	31
Sm_PolicyApi_Affiliate_ID	36
Sm_PolicyApi_AffiliateAttr_ID	37
Sm_PolicyApi_SharedSecretPolicy_ID	38
Sm_PolicyApi_UserContext_ID	40
Sm_PolicyApi_SAMLSP_ID	41
Sm_PolicyApi_SAMLProviderProp_ID	42
Sm_PolicyApi_SAMLAffiliation_ID	43
Sm_PolicyApi_SAMLSPAttr_ID	44

Name	Value
Sm_PolicyApi_WSFEDResourcePartner_ID	45
Sm_PolicyApi_WSFEDProviderProp_ID	46
Sm_PolicyApi_WSFEDRPAttr_ID	47
Sm_PolicyApi_SAMLRequesterAttr_ID	48
Sm_PolicyApi_SAMLSPAssertionConsumerService_ ID	49

Structure of a Policy Application

Policy applications must perform the following operations:

- **Initialization:** Sm_PolicyApi_Init() or Sm_PolicyApi_InitEx() must be the first function called by the API client session. The function initializes the connection to the SiteMinder policy store and establishes the *init handle*. (The *init handle* is passed in calls to Sm_PolicyApi_Login() and Sm_PolicyApi_Release().)
- **Login:** Sm_PolicyApi_Login() must be called after initialization and before making calls to the other functions in the API. This function can check the administrator's login credentials (username and password) or the administrator's validity.
  If the administrator is authenticated, the function initializes internal data structures and resources. Once the administrator is logged in, the Policy Server initializes a handle that is used as an input parameter to subsequent Policy Management API functions. Sm_PolicyApi_Login() can be called more than once during the client session.
- **Logout:** Sm_PolicyApi_Logout() logs out the administrator.
- **Release:** Sm_PolicyApi_Release() must be the last function called by the API client session. It disconnects from the policy store and releases memory and resources held by the API. This function must be called once per client session. Failure to call this function will result in a memory leak.

Functions by Category in the Policy Management API

Most of the functions in the Policy Management API are categorized according to the SiteMinder policy store object (for example, an agent, policy, or rule) that a given function acts upon. There are additional categories-required functions, user state functions, and utility functions such as cache and agent encryption key management-that are categorized by the type of service that certain functions perform.

Use these categories to help you find a particular Policy Management API function to use in your custom policy management applications.

The categories of Policy Management API functions include:

- Required Functions
- Administrator Functions
- Agent Functions
- Agent Configuration Functions
- Authentication/Authorization Map Functions
- Authentication Scheme Functions
- Certificate Mapping Functions
- Domain Functions
- Federation Functions
- General Object Functions
- Group Functions
- ODBC Query Scheme Functions
- Password Policy Functions
- Policy Functions
- Realm Functions
- Registration Scheme Functions
- Regular Expression Functions
- Response Functions
- Rule Functions
- SAML 2.0 Attribute Authority Functions
- SAML 2.0 Configuration Functions
- SAML 2.0 Indexed Endpoint Functions
- User Directory Functions
- User and User State Functions

- User Password State Functions
- Utility Functions
- Variable Functions
- WS-Federation Functions

Required Functions

The following functions must be used in all policy management applications:

Function	Description
Sm_PolicyApi_Init() (see page 475)	Initializes the connection to the Policy Server.
Sm_PolicyApi_InitEx() (see page 476)	Initializes a connection to the SiteMinder policy store and establishes the init handle based on a supplied version. Required for clients starting at version SM_POLICY_API_VERSION_6_0.
Sm_PolicyApi_Login() (see page 479)	Authenticates the administrator.
Sm_PolicyApi_Logout() (see page 481)	Logs out the administrator.
Sm_PolicyApi_Release() (see page 486)	Disconnects from the Policy Server and releases memory and resources held by the API.

Administrator Functions

The following functions manage SiteMinder administrator objects:

Function	Description
Sm_PolicyApi_AddAdmin() (see page 266)	Creates or updates an administrator object.
Sm_PolicyApi_AddAdminToDomain() (see page 268)	Gives the administrator permission to administer the specified domain.
Sm_PolicyApi_DeleteAdmin() (see page 336)	Deletes an administrator.
Sm_PolicyApi_GetAdmin() (see page 368)	Gets the contents of an administrator by object identifier.

Function	Description
Sm_PolicyApi_GetAdminByName() (see page 369)	Gets the contents of an administrator by name.
Sm_PolicyApi_GetGlobalPolicyByName() (see page 408)	Gets a specified global policy by name.
Sm_PolicyApi_GetGlobalResponseByName() (see page 409)	Gets a specified global response by name.
Sm_PolicyApi_GetGlobalRuleByName() (see page 410)	Gets a specified global rule by name.
Sm_PolicyApi_RemoveAdminFromDomain() (see page 488)	Disassociates the administrator from the specified domain.

Agent Functions

The following functions manage SiteMinder agent objects.

Note: There is no facility for creating Agent Types or Agent Type attributes.

Function	Description
Sm_PolicyApi_AddAgent() (see page 272)	Creates or updates an agent object.
Sm_PolicyApi_ConvertFromLegacyAgent() (see page 328)	Converts a v4.x agent to a v5.x agent.
Sm_PolicyApi_ConvertToLegacyAgent() (see page 329)	Converts a v5.x agent to a v4.x agent.
Sm_PolicyApi_DeleteAgent() (see page 339)	Deletes an agent.
Sm_PolicyApi_GetAgent() (see page 380)	Gets the contents of an agent by OID.
Sm_PolicyApi_GetAgentByName() (see page 381)	Gets the contents of an agent by name.
Sm_PolicyApi_GetAgentType() (see page 385)	Gets the contents of an agent type object by OID.
Sm_PolicyApi_GetAgentTypeByName() (see page 386)	Gets the contents of an agent type object by name.
Sm_PolicyApi_GetAgentTypeAttr() (see page 387)	Returns one or all agent type attributes.

Function	Description
Sm_PolicyApi_GetAgentTypeAttrByName() (see page 388)	Returns named agent type attribute object.

Agent Configuration Functions

The following functions manage configuration objects (agent configuration objects, host configuration objects, and trusted host objects) for centrally configuring agents.

Trusted hosts are created in any of the following ways:

- During agent installation
- Through the command line using the smreghost tool. For information about using this tool, see *Web Agent Installation Guide*.
- Through the Sm_PolicyApi_AddTrustedHost() function.

Function	Description
Sm_PolicyApi_AddAgentConfig() (see page 273)	Adds or modifies an agent configuration object.
Sm_PolicyApi_AddAgentConfigAssociation() (see page 274)	Adds or modifies a configuration parameter name and corresponding value in a specified agent configuration object.
Sm_PolicyApi_AddHostConfig() (see page 286)	Adds or modifies a host configuration object.
Sm_PolicyApi_AddTrustedHost() (see page 312)	Creates or modifies a trusted host object in the object store when there is no connection between the agent and the Policy Server.
Sm_PolicyApi_DeleteAgentConfig() (see page 340)	Deletes an agent configuration object.
Sm_PolicyApi_DeleteHostConfig() (see page 344)	Deletes a host configuration object.
Sm_PolicyApi_DeleteTrustedHost() (see page 357)	Deletes a trusted host object.

Function	Description
Sm_PolicyApi_GetAgentConfig() (see page 382)	Retrieves an agent configuration object.
Sm_PolicyApi_GetAgentConfigAssociations() (see page 384)	Retrieves a list of configuration parameters for an agent configuration object.
Sm_PolicyApi_GetAgentConfigByName() (see page 383)	Retrieves an agent configuration object by name.
Sm_PolicyApi_GetHostConfig() (see page 416)	Retrieves a host configuration object.
Sm_PolicyApi_GetHostConfigByName() (see page 417)	Retrieves a host configuration object by name.
Sm_PolicyApi_GetSharedSecretPolicy() (see page 452)	Retrieves the current shared secret policy.
Sm_PolicyApi_GetTrusted (see page 455) Host() (see page 455)	Retrieves a trusted host object by OID.
Sm_PolicyApi_GetTrustedHostByName() (see page 456)	Retrieves a trusted host object by name.
Sm_PolicyApi_RemoveAgentConfigAssociation() (see page 489)	Removes a configuration parameter name/value pair from a specified agent configuration object.
Sm_PolicyApi_SetSharedSecretPolicy() (see page 517)	Sets the current SharedSecretPolicy.

Authentication/Authorization Map Functions

The following functions manage SiteMinder authentication and authorization directory mapping objects:

Function	Description
Sm_PolicyApi_CreateAuthAzMap() (see page 330)	Creates or updates an authentication and authorization directory mapping object.
Sm_PolicyApi_DeleteAuthAzMap() (see page 341)	Deletes an authentication and authorization directory map.

Function	Description
Sm_PolicyApi_GetAuthAzMap() (see page 398)	Gets the contents of an authentication and authorization directory map.

Authentication Scheme Functions

The following functions manage SiteMinder authentication schemes:

Function	Description
Sm_PolicyApi_AddScheme() (see page 308)	Creates or updates an authentication scheme.
Sm_PolicyApi_DeleteScheme() (see page 356)	Deletes an authentication scheme.
Sm_PolicyApi_GetScheme() (see page 451)	Gets the contents of an authentication scheme by OID.
Sm_PolicyApi_GetSchemeByName() (see page 452)	Gets the contents of an authentication scheme by name.

Certificate Mapping Functions

The following functions manage SiteMinder certificate mapping objects:

Function	Description
Sm_PolicyApi_CreateCertMap() (see page 331)	Creates or updates a certificate mapping object.
Sm_PolicyApi_DeleteCertMap() (see page 342)	Deletes a certificate map.
Sm_PolicyApi_GetCertMap() (see page 399)	Gets the contents of a certificate map.

Domain Functions

The following functions manage SiteMinder domain objects:

Function	Description
Sm_PolicyApi_AddDomain() (see page 280)	Creates or updates a domain.
Sm_PolicyApi_DeleteDomain() (see page 343)	Deletes the domain and any domain children (rules, responses, realms, and policies).
Sm_PolicyApi_GetDomain() (see page 404)	Gets the contents of the domain by OID.
Sm_PolicyApi_GetDomainByName() (see page 405)	Gets the contents of a specified domain by name.
Sm_PolicyApi_GetDomainObjects() (see page 406)	Gets the OIDs of domain objects for a specified object type within the specified domain.

Federation Functions

The following functions support the manipulation of Policy Store data (Affiliate Domain and Affiliate objects) required to generate SAML assertions.

Function	Description
Sm_PolicyApi_AddAdminToAffiliateDomain() (see page 267)	Adds an administrator to an affiliate domain.
Sm_PolicyApi_AddAffiliate() (see page 269)	Creates a new or update an existing affiliate object.
Sm_PolicyApi_AddAffiliateDomain() (see page 270)	Creates a new or updates an existing affiliate domain.
Sm_PolicyApi_AddAttributeToAffiliate() (see page 277)	Adds a new attribute to an affiliate.
Sm_PolicyApi_AddUserDirToAffiliateDomain() (see page 313)	Adds a user directory to an affiliate domain.
Sm_PolicyApi_AddUsersToAffiliate() (see page 317)	Adds a user directory entry to an affiliate.

Function	Description
Sm_PolicyApi_DeleteAffiliate() (see page 337)	Deletes an affiliate.
Sm_PolicyApi_DeleteAffiliateDomain() (see page 338)	Deletes an affiliate domain.
Sm_PolicyApi_GetAffiliate() (see page 370)	Gets an affiliate by OID.
Sm_PolicyApi_GetAffiliateByName() (see page 371)	Gets and affiliate by name.
Sm_PolicyApi_GetAffiliateDomain() (see page 372)	Gets an affiliate domain by OID.
Sm_PolicyApi_GetAffiliateDomainByName() (see page 373)	Gets an affiliate domain by name
Sm_PolicyApi_GetAffiliateDomainObjects() (see page 374)	Gets the OIDs of domain objects for a given object type within the affiliate domain.
Sm_PolicyApi_GetAffiliateDomainUserDirSearchOrder() (see page 376)	Gets the user directory search order for an affiliate domain.
Sm_PolicyApi_GetAffiliateUsers() (see page 379)	Gets the user directory entries for an affiliate.
Sm_PolicyApi_GetAllAffiliateAttributes() (see page 389)	Gets all attributes for an affiliate.
Sm_PolicyApi_GetAllAffiliates() (see page 390)	Gets all affiliates for an affiliate domain.
Sm_PolicyApi_GetGlobalObjects() (see page 410)	Gets affiliate domains. (Not exclusively a Federation function.)
Sm_PolicyApi_InitEx() (see page 476)	See Required Functions.
Sm_PolicyApi_RemoveAdminFromAffiliateDomain() (see page 487)	Removes an administrator from an affiliate domain.

Function	Description
Sm_PolicyApi_RemoveAttributeFromAffiliate() (see page 491)	Removes an attribute from an affiliate.
Sm_PolicyApi_RemoveUserDirFromAffiliateDomain() (see page 500)	Removes a user directory from an affiliate domain.
Sm_PolicyApi_RemoveUsersFromAffiliate() (see page 502)	Removes a user directory entry from an affiliate.
Sm_PolicyApi_SetAffiliateDomainUserDirSearchOrder() (see page 509)	Sets the user directory search order for an affiliate domain.

General Object Functions

The following functions act on multiple types of SiteMinder objects:

Function	Description
Sm_PolicyApi_GetGlobalObjects() (see page 410)	Gets the OIDs of global objects for a specified object type.
Sm_PolicyApi_RenameObject() (see page 508)	Renames an object.

Group Functions

The following functions manage SiteMinder group objects. You can create agent groups, response groups, and rule groups, as enumerated in Sm_PolicyApi_Groups_t.

Note: Groups of global objects are not supported.

A group can contain individual items or groups of its own type. For example, a rule group can contain rules and/or groups of rules.

Function	Description
Sm_PolicyApi_AddGroup() (see page 285)	Creates or updates an agent, response, or rule group object.

Function	Description
Sm_PolicyApi_AddToGroup() (see page 310)	Adds an agent, response, or rule item to a group.
Sm_PolicyApi_DeleteGroup() (see page 345)	Deletes a group.
Sm_PolicyApi_GetGroup() (see page 411)	Gets the contents of a group by OID.
Sm_PolicyApi_GetGroupByName() (see page 413)	Gets the contents of a group by name.
Sm_PolicyApi_GetGroupOids() (see page 415)	Gets the OIDs contained with a group.
Sm_PolicyApi_IsGroup() (see page 477)	Determines whether a specified item is contained within the group.
Sm_PolicyApi_RemoveFromGroup() (see page 495)	Removes the specified item from the group.

ODBC Query Scheme Functions

The following functions manage SiteMinder ODBC query schemes:

Function	Description
Sm_PolicyApi_CreateODBCQueryScheme() (see page 332)	Creates or updates an ODBC query scheme.
Sm_PolicyApi_DeleteODBCQueryScheme() (see page 347)	Deletes an ODBC query scheme.
Sm_PolicyApi_GetODBCQueryScheme() (see page 420)	Gets the contents of an ODBC query scheme by OID.
Sm_PolicyApi_GetODBCQuerySchemeByName() (see page 421)	Gets the contents of an ODBC query scheme by name.

Password Policy Functions

The following functions manage SiteMinder password policy objects:

Function	Description
Sm_PolicyApi_AddPasswordPolicy() (see page 290)	Creates or updates a password policy object.
Sm_PolicyApi_DeletePasswordPolicy() (see page 348)	Deletes a password policy.
Sm_PolicyApi_GetPasswordPolicy() (see page 425)	Gets the contents of a password policy by OID.
Sm_PolicyApi_GetPasswordPolicyByName() (see page 426)	Gets the contents of a password policy by name.

Policy Functions

The following functions manage SiteMinder policy and policy link objects. A policy link is an association of a policy, a rule, and optionally, a response.

Function	Description
Sm_PolicyApi_AddGlobalPolicy() (see page 281)	Creates or updates a global policy object.
Sm_PolicyApi_AddPolicy() (see page 291)	Creates or updates a policy object.
Sm_PolicyApi_AddPolicyLink() (see page 292)	Creates a policy link for the specified policy.
Sm_PolicyApi_DeletePolicy() (see page 349)	Deletes a policy.
Sm_PolicyApi_GetPasswordPolicyByName() (see page 426)	Gets the contents of a password policy by name.
Sm_PolicyApi_GetPolicy() (see page 427)	Gets the contents of a policy by OID.
Sm_PolicyApi_GetPolicyLinks() (see page 429)	Gets a linked list of all policy links associated with the specified policy.
Sm_PolicyApi_RemovePolicyLinkFromPolicy() (see page 497)	Removes a policy link from the specified policy.

Realm Functions

The following functions manage SiteMinder realms objects:

Function	Description
Sm_PolicyApi_AddRealm() (see page 294)	Creates or updates a realm object.
Sm_PolicyApi_DeleteRealm() (see page 350)	Deletes a realm.
Sm_PolicyApi_GetChildren() (see page 400)	Builds a hierarchical realm and rule tree.
Sm_PolicyApi_GetRealm() (see page 432)	Gets the contents of a realm by OID.
Sm_PolicyApi_GetRealmByName() (see page 433)	Gets the contents of a realm by name.

Registration Scheme Functions

The following functions manage SiteMinder registration schemes:

Function	Description
Sm_PolicyApi_AddRegistrationScheme() (see page 297)	Creates or updates a registration scheme.
Sm_PolicyApi_DeleteRegistrationScheme() (see page 351)	Deletes a registration scheme.
Sm_PolicyApi_GetRegistrationScheme() (see page 436)	Gets the contents of a registration scheme by OID.
Sm_PolicyApi_GetRegistration (see page 437) SchemeByName() (see page 437)	Gets the contents of a registration scheme by name.

Regular Expression Functions

The following functions manage SiteMinder regular expressions:

Function	Description
Sm_PolicyApi_AddRegularExpressionToPasswordPolicy() (see page 298)	Creates or updates a regular expression.

Function	Description
Sm_PolicyApi_RemoveRegularExpressionFromPasswordPolicy() (see page 498)	Deletes a regular expression.
Sm_PolicyApi_GetRegularExpressions() (see page 438)	Gets the regular expressions belonging to a given password policy.

Response Functions

The following functions manage SiteMinder response objects:

Function	Description
Sm_PolicyApi_AddGlobalResponse() (see page 282)	Creates or updates a global response object.
Sm_PolicyApi_AddResponse() (see page 299)	Creates or updates a response object.
Sm_PolicyApi_AddResponseAttr() (see page 300)	Creates a response attribute for the specified response.
Sm_PolicyApi_DeleteResponse() (see page 352)	Deletes a response.
Sm_PolicyApi_GetResponse() (see page 439)	Gets the contents of a response by OID.
Sm_PolicyApi_GetResponseAttrs() (see page 441)	Gets a linked list of response attributes for the specified response.
Sm_PolicyApi_GetResponseByName() (see page 440)	Gets the contents of a response by name.
Sm_PolicyApi_RemoveResponseAttr() (see page 499)	Disassociates a response attribute from the specified response.
Sm_PolicyApi_SetResponseInPolicyLink() (see page 516)	Sets a response or response group to a rule or rule group, or removes a response or response group from a rule or rule group.

Rule Functions

The following functions manage SiteMinder rule objects:

Function	Description
Sm_PolicyApi_AddGlobalRule() (see page 284)	Creates or updates a global rule object.
Sm_PolicyApi_AddRule() (see page 302)	Creates or updates a rule object.
Sm_PolicyApi_DeleteRule() (see page 353)	Deletes a rule.
Sm_PolicyApi_GetRule() (see page 442)	Gets the contents of a rule by OID.
Sm_PolicyApi_GetRuleByName() (see page 443)	Gets the contents of a rule by name.

SAML1.x Configuration Functions

The following functions provide support for SAML 1.x configuration settings:

Function	Description
Sm_PolicyApi_AddMessageConsumerPluginTo SAML1Scheme()	Adds or updates a message consumer plugin setting to a SAML1.x authentication scheme.
Sm_PolicyApi_AddRedirectURLTo SAML1xScheme() (see page 296)	Adds or updates a redirect URL to a SAML1.x authentication scheme.
Sm_PolicyApi_GetMessageConsumerPluginFrom SAML1Scheme()	Retrieves a message consumer plugin setting from a SAML 1.x authentication scheme.
Sm_PolicyApi_GetRedirectURLFromSAML1Scheme()	Retrieves a redirect URL setting from a SAML 1.x authentication scheme.

SAML 2.0 Attribute Authority Functions

The following functions managed attributes for a SAML Requester:

Function	Description
Sm_PolicyApi_AddAttributeToSAMLScheme() (see page 278)	Adds an attribute to a SAML Requester defined in a SAML 2.0 authentication scheme.
Sm_PolicyApi_GetAllSAMLSchemeAttributes() (see page 392)	Retrieves all attributes defined for a SAML Requester.
Sm_PolicyApi_RemoveAttributeFromSAMLScheme() (see page 492)	Removes an attribute from a SAML Requester defined in a SAML 2.0 authentication scheme.

SAML 2.0 Configuration Functions

The following functions manage SAML 2.0 affiliations, Service Providers, and Identity Providers.

Function	Description
Sm_PolicyApi_AddAttributeToSAMLSP() (see page 279)	Defines a SAML 2.0 attribute for the Service Provider.
Sm_PolicyApi_AddSAMLAffiliation() (see page 303)	Adds a new SAML affiliation object or modifies an existing one.
Sm_PolicyApi_AddSAMLScheme() (see page 304)	Adds a new SAML 2.0 authentication scheme object or modifies an existing one.
Sm_PolicyApi_AddSAMLServiceProvider() (see page 306)	Adds a new SAML 2.0 Service Provider object or modifies an existing one.

Function	Description
Sm_PolicyApi_AddUsersToSAMLServiceProvider() (see page 320)	Associates a user directory entry with SAML 2.0 Service Provider.
Sm_PolicyApi_DeleteSAMLAffiliation() (see page 354)	Deletes the specified SAML affiliation.
Sm_PolicyApi_DeleteSAMLServiceProvider() (see page 355)	Deletes the specified Service Provider.
Sm_PolicyApi_GetAffiliatedSAMLAuthSchemes() (see page 377)	Retrieves all the SAML authentication schemes associated with the specified SAML affiliation.
Sm_PolicyApi_GetAffiliatedSAMLServiceProviders() (see page 378)	Retrieves all the Service Providers associated with the specified SAML affiliation.
Sm_PolicyApi_GetAllSAMLAffiliations() (see page 391)	Retrieves all existing SAML affiliation objects.
Sm_PolicyApi_GetAllSAMLServiceProviders() (see page 393)	Retrieves all the Service Providers in the specified affiliate domain.
Sm_PolicyApi_GetAllSAMLSPAttributes() (see page 395)	Retrieves all the attributes associated with the specified Service Provider.
Sm_PolicyApi_GetSAML (see page 444) Affiliation() (see page 444)	Retrieves the SAML affiliation specified by its OID in the policy store.
Sm_PolicyApi_GetSAMLAffiliationById() (see page 445)	Retrieves the SAML affiliation specified by its unique affiliation identifier (URI).
Sm_PolicyApi_GetSAMLScheme() (see page 446)	Retrieves information about a SAML 2.0 authentication scheme.

Function	Description
Sm_PolicyApi_GetSAMLServiceProvider() (see page 448)	Retrieves the Service Provider specified by its OID in the policy store.
Sm_PolicyApi_GetSAMLServiceProviderById() (see page 449)	Retrieves the Service Provider specified by its unique provider identifier.
Sm_PolicyApi_GetSAMLServiceProviderUsers() (see page 450)	Retrieves the user directory entries associated with the specified Service Provider.
Sm_PolicyApi_RemoveAttributeFromSAMLSP() (see page 493)	Removes the specified SAML attribute from the Service Provider.
Sm_PolicyApi_RemoveUsersFromSAMLServiceProvider() (see page 505)	Removes the specified users from the Service Provider.

SAML 2.0 Indexed Endpoint Functions

The following functions manage indexed endpoints in the Service Provider:

Function	Description
Sm_PolicyApi_AddAssertionConsumerServiceToSAMLSP() (see page 276)	Adds a new indexed endpoint reference (with index, binding, and Assertion Consumer URL) to a Service Provider.
Sm_PolicyApi_GetAllSAMLSPAssertionConsumerService() (see page 394)	Gets a list of all Assertion Consumer Services present in the policy store.

Function	Description
Sm_PolicyApi_RemoveAssertionConsumerService() (see page 490)	Removes an indexed endpoint reference to an Assertion Consumer Service.

User Directory Functions

The following functions manage SiteMinder user directory objects:

Function	Description
Sm_PolicyApi_AddUserDirToDomain() (see page 315)	Associates a directory object with the specified domain.
Sm_PolicyApi_CreateUserDir() (see page 333)	Creates or updates a user directory object.
Sm_PolicyApi_DeleteUserDir() (see page 358)	Deletes a user directory.
Sm_PolicyApi_GetDirectoryContents() (see page 401)	Gets a linked list of user structures for the specified user directory.
Sm_PolicyApi_GetUserContext() (see page 458)	Allows callers of the Policy Management API to access user context information.
Sm_PolicyApi_GetUserDir() (see page 460)	Gets the contents of a user directory by OID.
Sm_PolicyApi_GetUserDirByName() (see page 461)	Gets the contents of a user directory by name.
Sm_PolicyApi_GetUserDirCapabilities() (see page 462)	Gets the capabilities of the user directory.

Function	Description
Sm_PolicyApi_GetUserDirSearchOrder() (see page 463)	Gets the OIDs of the user directory associated with the specified domain.
Sm_PolicyApi_LookupDirectoryEntry() (see page 482)	Finds a user specification in a particular user directory and based on the specified search pattern.
Sm_PolicyApi_RemoveUserDirFromDomain() (see page 501)	Disassociates a user directory from the specified domain.
Sm_PolicyApi_SetUserDirSearchOrder() (see page 518)	Rearranges the search order of the user directories associated with the specified domain.
Sm_PolicyApi_ValidateDirectoryEntry() (see page 521)	Validates a user specification in a given path and user directory.

User and User State Functions

The following functions perform operations on user state and on user entries in a SiteMinder user directory:

Field	Description
Sm_PolicyApi_AddUsersToPolicy() (see page 318)	Adds a user to the specified policy.
Sm_PolicyApi_DisableUser() (see page 361)	Disables a user.
Sm_PolicyApi_EnableUser() (see page 362)	Enables a user.
Sm_PolicyApi_GetDisabledUserState() (see page 402)	Gets the disabled state of a user.

Field	Description
Sm_PolicyApi_GetPasswordMsg() (see page 423)	Gets information about an error that occurred during an attempt to validate a new password.
Sm_PolicyApi_GetPolicyUsers() (see page 430)	Gets a linked list of structures for the users associated with the specified policy and optionally, user directory.
Sm_PolicyApi_GetUserGroups() (see page 464)	Gets the list of groups that a user is member of.
Sm_PolicyApi_RemoveUsersFromPolicy() (see page 503)	Disassociates the user from the specified policy.
Sm_PolicyApi_SetDisabledUserState() (see page 510)	Sets the disabled state of a user.
Sm_PolicyApi_SetPassword() (see page 514)	Changes or validates a user password.
Sm_PolicyApi_SetPath()	Sets the path of a user in a specified policy.

User Password State Functions

The following functions manage SiteMinder password state objects:

Function	Description
Sm_PolicyApi_GetUserPasswordState() (see page 465)	Returns a UserPasswordState object.
Sm_PolicyApi_SetUserPasswordState() (see page 519)	Adds/Updates a UserPasswordState object.

Utility Functions

The following functions provide a variety of services, including memory, cache, and agent encryption key management:

Function	Description
Sm_PolicyApi_FlushRealm() (see page 363)	Flushes the specified realm from the resource cache.
Sm_PolicyApi_FlushUser() (see page 364)	Flushes the specified user from user cache.
Sm_PolicyApi_FreeMemory() (see page 365)	Frees memory allocated by the Policy Management API.
Sm_PolicyApi_FreeMemoryEx() (see page 366)	Frees memory allocated by the Policy Management API. Required for clients starting at version SM_POLICY_API_VERSION_6_0.
Sm_PolicyApi_FreeString() (see page 367)	Frees a single string allocated by the Policy Management API.
Sm_PolicyApi_FreeStringArray() (see page 367)	Frees string arrays allocated by the Policy Management API.
Sm_PolicyApi_ManagementCommand() (see page 485)	Performs user, key, and resource management services.

Variable Functions

The following methods manage variables.

Function	Description
Sm_PolicyApi_AddVariable() (see page 323)	Adds a variable object.
Sm_PolicyApi_DeleteVariable() (see page 359)	Deletes a variable object.
Sm_PolicyApi_GetVariable() (see page 468)	Gets a specified variable by OID.
Sm_PolicyApi_GetVariableByName() (see page 469)	Gets a specified variable by name.
Sm_PolicyApi_GetVariableType() (see page 470)	Gets a specified variable type by OID.
Sm_PolicyApi_GetVariableTypeByName() (see page 471)	Gets a specified variable type by name.

WS-Federation Functions

The following table lists the supported functions for Resource Partners and Account Partners:

Function	Description
Sm_PolicyApi_AddWSFEDResourcePartner() (see page 324)	Creates or updates a WS-Federation Resource Partner object.
Sm_PolicyApi_GetWSFEDResourcePartner() (see page 473)	Gets a WS-Federation Resource Partner object
Sm_PolicyApi_GetAllWSFEDResourcePartners() (see page 397)	Gets all WS-Federation Resource Partner objects for a domain as a linked list
Sm_PolicyApi_DeleteWSFEDResourcePartner() (see page 360)	Deletes a WS-Federation Resource Partner. This will then delete the agent, realm, rule, policy, and policy link objects associated with the Resource Partner object
Sm_PolicyApi_AddUsersToWSFEDResourcePartner() (see page 322)	Associates a user directory entry with a WS-Federation Resource Partner.
Sm_PolicyApi_RemoveUsersFromWSFEDResourcePartner() (see page 507)	Disssociates a user directory entry from a WS-Federation Resource Partner

Function	Description
Sm_PolicyApi_GetUsersFromWSFEDResourcePartner() (see page 467)	Gets the user directory entries associated with a WS-Federation Resource Partner
Sm_PolicyApi_AddWSFEDScheme() (see page 326)	Creates or updates a WS-Federation authentication scheme
Sm_PolicyApi_GetWSFEDScheme() (see page 474)	Gets a WS-Federation authentication scheme.

Function Declarations for the Policy Management API

Function declarations include the syntax and return values for each function in the Policy Management API for reference.

Sm_PolicyApi_AddAdmin()

Creates a new SiteMinder administrator object at a global scope. The administrator's attributes are contained in the *pstructAdmin* structure.

If the administrator object exists and *bUpdate* is true, the item is updated.

Type

Administrator function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAdmin (
   void*                    pSessionHandle,
   Sm_PolicyApi_Admin_t*    pstructAdmin,
   const bool               bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructAdmin	I	A pointer to a completely filled-in administrator structure.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The administrator object was created successfully.
- Sm_PolicyApi_Failure. The administrator object was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create an administrator object.
- Sm_PolicyApi_InvalidOid:
  - The administrator object identifier was not found during an update.
  - The user directory object identifier was not found.
  - The scheme object identifier was not found.

Sm_PolicyApi_AddAdminToAffiliateDomain()

Adds an administrator to an affiliate domain.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAdminToAffiliateDomain (
   void*        pSessionHandle,
   const char*  pszAdminOid,
   const char*  pszAffiliateDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAdminOid	I	A null-terminated string containing the object identifier of an existing administrator.
pszAffiliate DomainOid	I	A null-terminated string containing the object identifier of an existing affiliate domain.

Returns

- Sm_PolicyApi_Success. The administrator was added to the affiliate domain.
- Sm_PolicyApi_Failure. The administrator was not added to the affiliate domain.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete an affiliate domain.
- Sm_PolicyApi_InvalidOid. The affiliate domain OID was not found during an update.
- Sm_PolicyApi_DuplicateEntry. The specified administrator object identifier already exists in the affiliate domain

Sm_PolicyApi_AddAdminToDomain()

Gives the specified administrator permission to administer the specified domain, and associates the administrator object identified by *szAdminOid* with the domain identified by *szDomainOid*.

Type

Administrator function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAdminToDomain (
   void*            pSessionHandle,
   const char*      pszAdminOid,
   const char*      pszDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAdminOid	I	A null-terminated string containing the object identifier of an existing administrator.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain.

Returns

- Sm_PolicyApi_Success. The administrator was added to the domain.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add an administrator to a domain.
- Sm_PolicyApi_InvalidOid. The administrator object identifier or the domain object identifier was not found.
- Sm_PolicyApi_DuplicateEntry. The specified administrator object identifier already exists in the domain.

Sm_PolicyApi_AddAffiliate()

Creates a new or updates an existing affiliate object. This function will also retrieve the PropertyCollection object based on the AffiliateDomain OID.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAffiliate (
   void*                       pSessionHandle,
   Sm_PolicyApi_Affiliate_t*   pstructAffiliate,
   const bool                  bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructAffiliate	I	A pointer to a completely filled-in affiliate structure.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Remarks

This function creates affiliate objects that are based on the artifact profile or the POST profile (see the Sm_PolicyApi_Affiliate_t field SAMLProfile). Creating an affiliate object based on the POST profile requires an API version of at least SM_POLICY_API_VERSION_6_0_2. If an earlier version is involved, the POST profile request is ignored (along with any POST-specific fields in Sm_PolicyApi_Affiliate_t) and an attempt is made to create an affiliate object based on the artifact profile.

Returns

- Sm_PolicyApi_Success. The affiliate was created successfully.
- Sm_PolicyApi_Failure. The affiliate was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create an affiliate.
- Sm_PolicyApi_DomainNotFound. The affiliate domain OID was not found.
- Sm_PolicyApi_DomainNotAffiliate. The domain is not an affiliate domain.

Sm_PolicyApi_AddAffiliateDomain()

Creates a new or updates an existing affiliate domain. Sets *bIsAffiliate* to TRUE.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAffiliateDomain (
   void*                           pSessionHandle,
   Sm_PolicyApi_AffiliateDomain_t  *pstructAffiliateDomain,
   const bool                      bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pstructAffiliate Domain	I	A pointer to a completely filled-in affiliate domain structure.
bUpdate	I	A boolean flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The affiliate domain was created successfully.
- Sm_PolicyApi_Failure. The affiliate domain was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create an affiliate domain.
- Sm_PolicyApi_DomainNotFound. The affiliate domain OID was not found during an update.
- Sm_PolicyApi_DomainNotAffiliate. The domain is not an affiliate domain.

Sm_PolicyApi_AddAgent()

Creates a new SiteMinder agent. The attributes of the agent are contained in the *pstructAgent* structure.

If the agent exists and the *bUpdate* flag is true, the item is updated.

You must specify an agent type with this call. To get the agent type OID for the agent, use Sm_PolicyApi_GetGlobalObjects().

Type

Agent function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAgent (
   void*                   pSessionHandle,
   Sm_PolicyApi_Agent_t*   pstructAgent,
   const bool              bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructAgent	I	A pointer to a completely filled-in agent structure.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The agent was created successfully.
- Sm_PolicyApi_Failure. The agent was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create an agent.
- Sm_PolicyApi_InvalidOid:
  - The agent type OID was not found or is not of the specified agent type.
  - The agent OID was not found. This happens when this function is called with *bUpdate* set to true and the *pszOid* field of Sm_PolicyApi_Agent_t holds the OID of the agent being updated.
- Sm_PolicyApi_NotUnique. An agent with the same name exists.
- Sm_PolicyApi_RadiusIpAddrNotUnique. Another RADIUS IP address exists that is the same.
- Sm_PolicyApi_MissingProperty. One of the required fields is not set.

Sm_PolicyApi_AddAgentConfig()

Creates or modifies an agent configuration object in the policy store.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAgentConfig (
   void*                            pSessionHandle,
   Sm_PolicyApi_AgentConfig_t*      pstructAgentConfig,
   const bool                       bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pstructAgentConfig	I	Address of a structure that defines the agent configuration object.
bUpdate	I	If true, the object is being updated.

Returns

- Sm_PolicyApi_Success. The operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to an agent configuration object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to add or modify an agent configuration object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_AddAgentConfigAssociation()

Adds or modifies a configuration parameter name and corresponding value in a specified agent configuration object.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAgentConfigAssociation (
   void*                              pSessionHandle,
   const char*                        pszAgentConfigOid,
   Sm_PolicyApi_Association_t*        pstructAssociation,
   bool                               bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszAgentConfigOid	I	Unique identifier of the agent configuration object.
pstructAssociation	I	The name/value pair to add or modify in the agent configuration object.
bUpdate	I	If true, the object is being updated.

Returns

- Sm_PolicyApi_Success. The operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to an agent configuration object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to add agent configuration object parameters.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_AddAssertionConsumerServiceToSAMLSP

Adds a new indexed endpoint, which includes an index, binding, and an Assertion Consumer Service URL, to the Service Provider.

Note: An existing indexed endpoint reference cannot be modified.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAssertionConsumerServiceToSAMLSP (
   void*                              pSessionHandle,
   const Sm_PolicyApi_SAMLSPAssertionConsumerService_t*
                pstructSAMLSPAssertionConsumerService,
   const char*                        pszSAMLSPOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructSAMLSPAssertion ConsumerService	I	A pointer to an Assertion Consumer Service structure.
pszSAMLSPOid	I	A pointer to a string containing the OID of the Service Provider.

Returns

- Sm_PolicyApi_Success. The Assertion Consumer Service was added successfully.
- Sm_PolicyApi_Failure - The Assertion Consumer Service was not added successfully.
- Sm_PolicyApi_ACSDuplicateIndex - There is already an Assertion Consumer Service assigned to thei Service Provider with the same index value.
- Sm_PolicyApi_InvalidHandle - There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession - There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege - The administrator does not have the privilege to add an Assertion Consumer Service.

Remarks

The following fields of the Sm_PolicyApi_SAMLSPAssertionConsumerService_t structure are evaluated:

- The value of the index is a unique positive integer, less than 65535,
- The Assertion Consumer URL starts with http:// or https://.
- The binding is either HTTP-Post or HTTP_Artifact.

Sm_PolicyApi_AddAttributeToAffiliate()

Adds a new attribute to an affiliate.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAttributeToAffiliate (
   void*                              pSessionHandle,
   const Sm_PolicyApi_AffiliateAttr_t* pstructAffiliateAttr,
   const char*                        pszAffiliateOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pstructAffiliate Attr	I	A pointer to a completely filled-in affiliate attribute structure.
pszAffiliateOid	I	A null-terminated string containing the object identifier of an existing affiliate.

Returns

- Sm_PolicyApi_Success. The affiliate attribute was created successfully.
- Sm_PolicyApi_Failure - The affiliate attribute was not created successfully.
- Sm_PolicyApi_InvalidHandle - There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession - There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege - The administrator does not have the privilege to create an affiliate attribute.
- Sm_PolicyApi_InvalidOid - The affiliate OID was not found.

Sm_PolicyApi_AddAttributeToSAMLScheme()

Adds an attribute, which can be requested from the configured Attribute Service, to a SAML 2.0 authentication scheme.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAttributeToSAMLScheme(
   void*                                  pHandle,
   const Sm_PolicyApi_Scheme_t*           pstructScheme,
   const Sm_PolicyApi_SAMLRequesterAttr_t*     pAttr
);
```

Parameter	I/O	Description
pHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructScheme	I	A pointer to a completely filled-in structure for a SAML 2.0 Scheme.
pAttr	I	A pointer to the Sm_PolicyApi_SAMLRequesterAttr_t structure containing the attribute to be added.

Returns

- Sm_PolicyApi_Success. The attribute was added successfully.
- Sm_PolicyApi_Failure. The attribute was not added successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add an attribute to a SAML Requester.
- Sm_PolicyApi_SAMLIDP_IncorrectParameters. Supplied SAML provided properties are incomplete or incorrect.
- Sm_PolicyApi_DuplicateAttribute. An attribute already exists with the same Name and NameFormat.

Sm_PolicyApi_AddAttributeToSAMLSP()

Defines a SAML 2.0 attribute for the Service Provider.

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 the Service Provider.

Type

SAML 2.0 Configuration function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAttributeToSAMLSP (
   void*                             pHandle,
   const Sm_PolicyApi_SAMLSPAttr_t*     pstructSAMLSPAttr,
   const char*                       pszSAMLSPOid
);
```

Parameter	I/O	Description
pHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructSAMLSPAttr	I	A pointer to a completely filled-in attribute structure.

Parameter	I/O	Description
pszSAMLSPOid		A null-terminated string containing the object identifier of an existing Service Provider.

Returns

- Sm_PolicyApi_Success. The attribute was added successfully.
- Sm_PolicyApi_Failure. The attribute was not added successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add an attribute to a SAML Service Provider.
- Sm_PolicyApi_DuplicateAttribute. An attribute already exists with the same Name and NameFormat.

Sm_PolicyApi_AddDomain()

Creates a new SiteMinder domain. Attributes of the domain are contained in the *pstructDomain* structure.

If the domain exists and the *bUpdate* flag is true, the item is updated.

Type

Domain function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddDomain (
   void*                    pSessionHandle,
   Sm_PolicyApi_Domain_t*   pstructDomain,
   const bool               bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pstructDomain	I	A pointer to a completely filled-in domain structure.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The domain was created successfully.
- Sm_PolicyApi_Failure. The domain was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a domain.
- Sm_PolicyApi_InvalidOid. The domain OID was not found during an update.

Sm_PolicyApi_AddGlobalPolicy()

Creates a new global policy in the object store. The policy attributes are contained in the *pStructPolicy* structure.

If the policy exists and the *bUpdate* flag is true, the item is updated.

Type

Policy function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddGlobalPolicy (
   void*                      pSessionHandle,
   Sm_PolicyApi_Policy_t*     pStructPolicy,
   const bool                 bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pStructPolicy	I	A pointer to a completely filled policy structure. The structure's domain OID is ignored.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The global policy was created successfully.
- Sm_PolicyApi_Failure. The global policy was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a global policy.
- Sm_PolicyApi_NotImplemented: The client who called this function initialized the API with a version less than SM_POLICY_API_VERSION_6_0.

Sm_PolicyApi_AddGlobalResponse()

Creates a new global response in the object store.

Type

Rule function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddGlobalResponse (
   void*                     pSessionHandle,
   Sm_PolicyApi_Response_t*  pStructResponse,
   const bool                bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pStructResponse	I	A pointer to a completely filled response structure. The structure's domain OID is ignored.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The global response was created successfully.
- Sm_PolicyApi_Failure. The global response was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a global response.
- Sm_PolicyApi_InvalidOid: A global response with a matching OID was not found during an update.
- Sm_PolicyApi_NotImplemented: The client who called this function initialized the API with a version less than SM_POLICY_API_VERSION_6_0.

Sm_PolicyApi_AddGlobalRule()

Creates a new global rule in the object store.

Type

Rule function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddGlobalRule (
   void*                   pSessionHandle,
   Sm_PolicyApi_Rule_t*    pStructRule,
   const bool              bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pStructRule	I	A pointer to a completely filled-in rule structure. The realm OID in the structure is ignored.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The global rule was created successfully.
- Sm_PolicyApi_Failure. The global rule was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a global rule.
- Sm_PolicyApi_InvalidOid:
  - A global rule with a matching OID was not found during an update.
  - An agent or agent group with matching OID was not found.
- Sm_PolicyApi_NotImplemented: The client who called this function initialized the API with a version less than SM_POLICY_API_VERSION_6_0.

Sm_PolicyApi_AddGroup()

Adds a new group object to the Siteminder policy store. The attributes of the group are contained in the *pStructGroup* structure.

Note: Groups of global objects are not supported.

The *pszDomainOid* parameter is required by a rule group or response group. An agent group does not require a domain OID because it is not a domain-based object.

If the group object exists and the *bUpdate* flag is true, the item is updated.

Type

Group function, global scope (agents) or domain scope (responses, rules).

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddGroup (
   void*                    pSessionHandle,
   Sm_PolicyApi_Groups_t    dwGroup,
   const char*              pszDomainOid,
   Sm_PolicyApi_Group_t*    pStructGroup,
   const bool               bUpdate
);
```

Parameter	I/O	Description
pSession Handle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
dwGroup	I	Indicates the type of group to be added.
pszDomainOid	I	A null-terminated string containing the name of an existing domain. Required parameter for rule or response group. (Global rule or response groups are not supported.)
pStructGroup	I	A pointer to a completely filled-in group structure.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The add was successful.
- Sm_PolicyApi_Failure. The add was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add a group.
- Sm_PolicyApi_InvalidOid. The domain OID was not found (for a domain-based group).
- Sm_PolicyApi_BadGroup. The *dwGroup* parameter is not the rule, response, or agent group type.

Sm_PolicyApi_AddHostConfig()

Creates or updates a host configuration object in the policy store.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddHostConfig (
   void*                          pSessionHandle,
   Sm_PolicyApi_HostConfig_t*     pstructHostConfig,
   const bool                     bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructHostConfig	I	Address of structure that defines the host configuration object.
bUpdate	I	If true, the object is being updated.

Returns

- Sm_PolicyApi_Success. The operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to a host configuration object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to create or modify a host configuration object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_AddMessageConsumerPluginToSAML1xScheme()

Adds or updates a message consumer plugin setting to a SAML 1.x authentication scheme.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddMessageConsumerPluginToSAML1xScheme(
   void*    pHandle,
   char*    pszSchemeOID,
   char*    pluginClass,
   char*    pluginParam
);
```

Parameters

phandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszSchemeOid

[in] A pointer to the OID of the authentication scheme that is being updated.

pluginClass

[in] A pointer to the name of the message consumer plugin class to be set.

pluginParam

[in] A pointer to the parameters of the message consumer plugin class to be set.

Returns

- Sm_PolicyApi_Success. The action was completed successfully.
- Sm_PolicyApi_Failure. The action was unsuccessful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_SchemeNotFound. The authentication scheme corresponding to the OID was not found, or was not a SAML 1.x authentication scheme.
- Sm_PolicyApi_InvalidOid. The OID of the authentication scheme is NULL.

Sm_PolicyApi_AddOneTimeUsePropToAffiliate()

Adds or updates the OneTimeUse property for an assertion in a SAML 1.x affiliate.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddOneTimeUsePropToAffiliate(
   void*    pHandle,
   char*    pszAffiliateOID,
   bool     bOneTimeUse

);
```

Parameters

pHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszAffiliateOid

[in] A pointer to the OID of an existing SMAL 1.x affiliate.

bOneTimeUse

[in] A Boolean value that specifies whether an assertion is used only once in this affiliate.

Returns

- Sm_PolicyApi_Success. The action was completed successfully.
- Sm_PolicyApi_Failure. The action was unsuccessful.
- Sm_PolicyApi_NoSession. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator did not have sufficient access privileges.

Sm_PolicyApi_AddPasswordPolicy()

Adds a password policy object.

Note the following about Sm_PolicyApi_PasswordPolicy_t:

- When the password policy applies to an entire directory (*bEntireDir* is set to true), you must set *pszPath* and *pszClass* to an empty string. The Policy Management API returns Sm_PolicyAPI_BadArgument if these strings are not empty.
- When the *pszPasswordServicesRedirect* field is set to an empty string, the Policy Management API sets it to a default URL:

  ```
  /siteminderagent/forms/smpwservices.fcc
  ```

Type

Password policy function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddPasswordPolicy (
   void*                             pSessionHandle,
   Sm_PolicyApi_PasswordPolicy_t*    pstructPasswordPolicy,
   const bool                        bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructPassword Policy	I	The address of a pointer to a Sm_PolicyApi_PasswordPolicy_t structure containing information about password policy.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The create was successful.
- Sm_PolicyApi_Failure. The create was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.

- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a password policy.
- Sm_PolicyApi_InvalidOid:
  - The password policy OID was not found during an update.
  - The user directory OID was not found.
- Sm_PolicyApi_BadArgument. The path and class are empty.

Sm_PolicyApi_AddPolicy()

Creates a new SiteMinder policy. The policy attributes are contained in the *pStructPolicy* structure.

If the policy exists and the *bUpdate* flag is true, the item is updated.

Type

Policy function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddPolicy (
   void*                    pSessionHandle,
   Sm_PolicyApi_Policy_t*   pStructPolicy,
   const bool               bUpdate
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pStructPolicy	I	A pointer to a completely filled-in policy structure.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The policy was created successfully.
- Sm_PolicyApi_Failure. The policy was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a policy.
- Sm_PolicyApi_InvalidOid:
  - The policy OID was not found during an update.
  - The domain OID was not found.

Sm_PolicyApi_AddPolicyLink()

Creates a new SiteMinder policy link for the policy identified by *pszPolicyOid*.

A policy link object binds a policy to a rule and, optionally, a response. The attributes of the new policy link are contained in the *pstructPolicyLink* structure.

If the policy link exists and the *bUpdate* flag is true, the item is updated.

Type

Policy function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddPolicyLink (
   void*                         pSessionHandle,
   const                         pszPolicyOid,
   Sm_PolicyApi_PolicyLink_t*    pstructPolicyLink
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPolicyOid	I	A null-terminated string containing the object identifier of an existing policy.

Parameter	I/O	Description
pstructPolicyLink	I	A pointer to a completely filled-in policy link structure.

Returns

- Sm_PolicyApi_Success. The policy link was created successfully.
- Sm_PolicyApi_Failure:
  - The domain of a rule OID is not the same as the domain of the policy OID.
  - The domain of a response OID is not the same as the domain of the policy OID.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a policy link.
- Sm_PolicyApi_InvalidOid:
  - The policy OID was not found.
  - The rule or rule group OID was not found.
  - The response or response group OID was not found.

Sm_PolicyApi_AddRealm()

Creates a new SiteMinder realm within the domain specified in *pStructRealm*. The *pStructRealm* structure also contains other attributes of the realm, including the agent or agent group that protects the realm.

Sm_PolicyApi_AddRealm() fails if a protecting agent or agent group is not specified.

If the new realm is a top-level realm, set *pszParentRealmOid* (in Sm_PolicyApi_Realm_t) to the domain OID. Otherwise, set *pszParentRealmOid* to the OID of the new realm's parent realm.

If the realm exists and the *bUpdate* flag is true, the existing item is updated.

It is the responsibility of the client application to meet the following conditions in order to add an authorization directory to a realm successfully:

- The directory mapping between the authorization user directory and authentication user directory should exist.

  Note: The Policy Management API checks only to see if the authorization user directory exists. It does not validate if there is a directory mapping for the given authorization user directory.

- The associated authentication user directory should be present in the user directory collection of the domain.

Type

Realm function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddRealm (
   void*                    pSessionHandle,
   Sm_PolicyApi_Realm_t*    pStructRealm,
   const bool               bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pStructRealm	I	A pointer to a completely filled-in structure.

Parameter	I/O	Description
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The realm was created successfully.
- Sm_PolicyApi_Failure. The realm was not created successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a realm.
- Sm_PolicyApi_InvalidOid:
  - The realm OID was not found during an update.
  - The domain OID was not found.
  - The agent OID or agent group OID was not found.
  - The scheme OID was not found.
  - The parent OID could not be found. (The parent OID can be a realm OID or a domain OID.)

Sm_PolicyApi_AddRedirectURLToSAML1xScheme()

Adds or updates a redirect URL setting in a SAML 1.x authentication scheme.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddRedirectURLToSAML1xScheme(
        void*        pSessionHandle,
        const char*  pszSchemeOid,
        int          iTypeURL,
        char*        URL,
        int          redirectMode
);
```

Parameters

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszSchemeOid

[in] A null-terminated string containing the object identifier of the authentication scheme being updated.

iTypeUrl

[in] An integer specifying the type of redirect URL, defined in Sm_PolicyAPI_SAML1_STATUS_REDIRECT_URL_TYPE_t as follows:

- Sm_PolicyAPI_SAML1_STATUS_REDIRECT_URL_USER_NOT_FOUND_TYPE = 0
- Sm_PolicyAPI_SAML1_STATUS_REDIRECT_URL_INVALID_SSO = 1
- Sm_PolicyAPI_SAML1_STATUS_REDIRECT_URL_UNACCEPTABLE_USER_CREDEN TIALS = 2

URL

[in] A pointer to the input redirect URL.

redirectMode

[in] An integer specifying the input redirect mode, which is either 0 for 302 No Data, or 1 for Http-Post.

Return Values

- Sm_PolicyApi_Success. The action was completed successfully.
- Sm_PolicyApi_Failure. The action was unsuccessful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_SchemeNotFound. The authentication scheme corresponding to the OID was not found, or was not a SAML 1.x authentication scheme.

- Sm_PolicyApi_InvalidOid. The OID of the authentication scheme is NULL.

Sm_PolicyApi_AddRegistrationScheme()

Adds a registration scheme.

Type

Registration scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddRegistrationScheme (
   void*                          pSessionHandle,
   Sm_PolicyApi_RegistrationScheme_t*
                                  pstructRegistrationScheme,
   const bool                     bUpdate
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstruct Registration Scheme	I	The address of a pointer to Sm_PolicyApi_RegistrationScheme_t, which contains information about the registration scheme.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The registration scheme was created successfully.
- Sm_PolicyApi_Failure. The registration scheme was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a registration scheme.
- Sm_PolicyApi_InvalidOid:
  - The registration scheme object identifier was not found during an update.
  - The user directory object identifier was not found.

Sm_PolicyApi_AddRegularExpressionToPasswordPolicy()

Adds a regular expression to the referenced password policy. Implemented only if the session's version is set to SM_POLICY_API_VERSION_6_0.

Type

Regular Expression function.

Syntax

```
int SM_EXTERN    Sm_PolicyApi_AddRegularExpressionToPasswordPolicy (
   void*                               pSessionHandle,
   const char*                         pszPasswordPolicyOid,
   Sm_PolicyApi_RegularExpression_t*   pstructRegExpr
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPassword PolicyOid	I	The OID of the password policy to add the regular expression to.
pstructRegExpr	I	A pointer to the regular expression structure to add.

Returns

- Sm_PolicyApi_Success. The regular expression was added successfully.
- Sm_PolicyApi_Failure. The regular expression was not added successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add a regular expression.
- Sm_PolicyApi_InvalidOid: The password policy OID was not found.

Sm_PolicyApi_AddResponse()

Creates a new SiteMinder response. The attributes of the response itself are contained in the *pStructResponse* structure.

If the response exists, and the *bUpdate* flag is true, the item is updated.

Type

Response function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddResponse (
   void*                       pSessionHandle,
   Sm_PolicyApi_Response_t*   pStructResponse,
   const bool                  bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pStructResponse	I	A pointer to a completely filled-in response structure.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The response was created successfully.
- Sm_PolicyApi_Failure. The response was not created successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a response.
- Sm_PolicyApi_InvalidOid:
  - The response OID was not found during an update.
  - The domain OID was not found.
  - The agent type OID was not found.

Sm_PolicyApi_AddResponseAttr()

Creates a new SiteMinder response attribute object within the response identified by *pszResponseOid*. The response attributes are contained in the *pstructResponseAttr* parameter.

Type

Response function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddResponseAttr (
   void*                          pSessionHandle,
   const char*                    pszResponseOid,
   Sm_PolicyApi_ResponseAttr_t*   pstructResponseAttr
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszResponseOid	I	A null-terminated string containing the object identifier of an existing response.

Parameter	I/O	Description
pstructResponseAttr	I	A pointer to a completely filled-in response structure.

Returns

- Sm_PolicyApi_Success. The response attribute was added successfully.
- Sm_PolicyApi_Failure. The response attribute was not added successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add a response attribute.
- Sm_PolicyApi_InvalidOid:

The response OID was not found.

  - The agent type attribute OID was not found.

Sm_PolicyApi_AddRule()

Creates a new SiteMinder rule. The attributes of the rule itself are contained in the structure referenced by *pStructRule*.

If the rule exists and the *bUpdate* flag is true, the item is updated.

Note: A rule is always associated with a realm. Rule names are unique within in a realm but not within a domain.

Type

Rule function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddRule (
   void*                   pSessionHandle,
   Sm_PolicyApi_Rule_t*    pStructRule,
   const bool              bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pStructRule	I	A pointer to a completed rule structure.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The rule was created successfully.
- Sm_PolicyApi_Failure. The rule was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a rule.
- Sm_PolicyApi_InvalidOid:
  - The rule OID was not found during an update.
  - The realm OID was not found.

Sm_PolicyApi_AddSAMLAffiliation()

Adds a new SAML affiliation object or modifies an existing one.

Type

SAML 2.0 Configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddSAMLAffiliation (
   void*                              pSessionHandle,
   Sm_PolicyApi_SAMLAffiliation_t*    pstructAffiliation,
   const bool                         bUpdate,
   char**                             pszErrMsg
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructAffiliation	I	A pointer to a completed SAML affiliation structure.
bUpdate	I	Specifies whether an update operation should be performed on an existing object.
pszErrMsg	O	String containing an error message if the affiliation is not added or updated successfully. The Policy Management API allocates memory for this parameter dynamically. It is the responsibility of the custom application to free it using a call to Sm_PolicyApi_FreeString(). If Sm_PolicyApi_MissingProperty or Sm_PolicyApi_InvalidProp are returned, this field contains the name of the property that is missing or invalid.

Returns

- Sm_PolicyApi_Success. The operation was successful.
- Sm_PolicyApi_Failure. The operation was not successful.
- Sm_PolicyApi_InvalidProp. A specified property is invalid. The property name is returned in *pszErrMsg*.
- Sm_PolicyApi_MissingProperty. A required property was not specified. The property name is returned in *pszErrMsg*.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a SAML affiliation.
- Sm_PolicyApi_InsufficientData. Required properties for configuring the SAML affiliation were missing.

Sm_PolicyApi_AddSAMLScheme()

Adds a new SAML 2.0 authentication scheme object or modifies an existing one. This function also defines metadata properties for the associated Identity Provider. The metadata properties are stored with the authentication scheme.

Type

SAML 2.0 Configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddSAMLScheme

(
   void*                                    pHandle,
   Sm_PolicyApi_Scheme_t*                   pstructScheme,
   Sm_PolicyApi_SAMLProviderProp_t*         pProps,
   const bool                               bUpdate,
   char**                                   pszErrMsg
);
```

Parameter	I/O	Description
pHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pstructScheme	I	A pointer to a completed scheme structure.
pProps	I	SAML 2.0 metadata properties associated with the authentication scheme. If you do not assign a value to a property associated with a default value, the default will be assigned.
bUpdate	I	Specifies whether an update operation should be performed on an existing object.
pszErrMsg	O	String containing an error message if the authentication scheme is not added or updated successfully. The Policy Management API allocates memory for this parameter dynamically. It is the responsibility of the custom application to free it using a call to Sm_PolicyApi_FreeString().. If Sm_PolicyApi_MissingProperty or Sm_PolicyApi_InvalidProp are returned, this field contains the name of the property that is missing or invalid.

Returns

- Sm_PolicyApi_Success. The SAML 2.0 authentication scheme operation was successful.
- Sm_PolicyApi_Failure. The SAML 2.0 authentication operation was not successful
- Sm_PolicyApi_InvalidProp. A specified property is invalid. The property name is returned in *pszErrMsg*.
- Sm_PolicyApi_MissingProperty. A required property was not specified. The property name is returned in *pszErrMsg*.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a SAML 2.0 authentication scheme.
- Sm_PolicyApi_SAMLIDP_IncorrectParameters. The supplied SAML provider properties are incomplete or incorrect.
- Sm_PolicyApi_SAMLIDP_ProviderIdNotUnique. The supplied SAML provider ID is not unique.

Sm_PolicyApi_AddSAMLServiceProvider()

Adds a new SAML 2.0 Service Provider object or modifies an existing one.

If a Service Provider cannot be created, any associated objects created in the policy store during the attempt will be rolled back.

Type

SAML 2.0 Configuration function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddSAMLServiceProvider
(
   void*                          pSessionHandle,
   Sm_PolicyApi_SAMLSP_t*         pstructSAMLSP,
   const bool                     bUpdate,
   char**                         pszErrMsg
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructSAMLSP	I	A pointer to a completed Service Provider structure.
bUpdate	I	Specifies whether an update operation should be performed on an existing object.
pszErrMsg	O	String containing an error message if the Service Provider is not added or updated successfully. The Policy Management API allocates memory for this parameter dynamically. It is the responsibility of the custom application to free it using a call to Sm_PolicyApi_FreeString(). If Sm_PolicyApi_MissingProperty or Sm_PolicyApi_InvalidProp are returned, this field contains the name of the property that is missing or invalid.

Returns

- Sm_PolicyApi_Success. The Service Provider operation was successful.
- Sm_PolicyApi_Failure. The Service Provider operation was not successful.
- Sm_PolicyApi_InvalidProp. A specified property is invalid. The property name is returned in *pszErrMsg*.
- Sm_PolicyApi_MissingProperty. A required property was not specified. The property name is returned in *pszErrMsg*.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a Service Provider.
- Sm_PolicyApi_DomainNotFound. The affiliate domain OID was not found.
- Sm_PolicyApi_DomainNotAffiliate. The domain is not an affiliate domain.
- Sm_PolicyApi_InsufficientSPData. Required properties for configuring the Service Provider were missing.

Sm_PolicyApi_AddScheme()

Creates a new SiteMinder authentication scheme. Attributes of the scheme are contained in the *pstructScheme* structure.

If the scheme exists and the *bUpdate* flag is true, the item is updated.

Type

Authentication scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddScheme (
   void*                   pSessionHandle,
   Sm_PolicyApi_Scheme_t*  pstructScheme,
   const bool              bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructScheme	I	A pointer to a completely filled-in scheme structure.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The scheme was created successfully.
- Sm_PolicyApi_Failure. The scheme was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a scheme.
- Sm_PolicyApi_InvalidOid. The scheme object identifier was not found during an update.

Sm_PolicyAPI_AddTargetConfigToSAML1xScheme

Adds a default target configuration setting in a SAML 1.x authentication scheme.

Syntax

The Sm_PolicyApi_AddTargetConfigToSAML1xScheme function has the following syntax:

```
int SM_EXTERN Sm_PolicyApi_AddTargetConfigToSAML1xScheme(
        void*        pHandle,
        const char*  pszSchemeOid,
        const char*  pszURL,
        int         iQPOverridesTarget
);
```

Parameters

The Sm_PolicyApi_AddTargetConfigToSAML1xScheme function accepts the following parameters:

pHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszSchemeOid

[in] A null-terminated string containing the object identifier of the authentication scheme being updated.

pszURL

[in] A pointer to a valid default target configuration URL.

iQPOverrides Target

[in] An integer specifying the value specified in the 'Query parameter override Default Target' check box.

Valid values for iQPOverridesTarget are:

- 0 specifies that the query parameter does not override the target.
- 1 specifies that the query parameter overrides the target.

Return Values

The Sm_PolicyApi_AddTargetConfigToSAML1xScheme function returns one of the following values:

- Sm_PolicyApi_Success. The action was completed successfully.
- Sm_PolicyApi_Failure. The action was unsuccessful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.

- Sm_PolicyApi_SchemeNotFound. The authentication scheme corresponding to the OID was not found, or was not a SAML 1.x authentication scheme.
- Sm_PolicyApi_InvalidOid. The OID of the authentication scheme is NULL.

Sm_PolicyApi_AddToGroup()

Adds an item to the group identified by *pszGroupOid*.

The item (which may be a group) and the group must exist and must be of the same type.

The *pszDomainOid* parameter is required by a rule group or response group. An agent group does not require a domain OID because it is not a domain-based object.

Type

Group function, global scope (agents) or domain scope (responses, rules).

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddToGroup (
   void*                  pSessionHandle,
   Sm_PolicyApi_Groups_t  dwGroup,
   const char*            pszItemOid,
   const char*            pszGroupOid,
   const char*            pszDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
dwGroup	I	The type of group.
pszItemOid	I	A null-terminated string containing the object identifier of an existing item of the same type as the group.

Parameter	I/O	Description
pszGroupOid	I	A null-terminated string containing the object identifier of a group of the type indicated by *dwGroup*.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain. Required for rule and response objects.

Returns

- Sm_PolicyApi_Success. The add was successful.
- Sm_PolicyApi_Failure. The add was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add a rule, response, or agent OID to its respective group.
- Sm_PolicyApi_InvalidOid:
    - The domain OID was not found (for a domain-based group).
    - The group OID was not found.
    - The rule, response, or agent OID or group OID was not found.
- Sm_PolicyApi_BadGroup. Parameter *dwGroup* is not the rule, response, or agent type.

Sm_PolicyApi_AddTrustedHost()

Creates or modifies a trusted host object in the object store.

Use this function to register the trusted host "offline"-that is, without a connection between the agent and the Policy Server. When you use this function, you must run the SiteMinder smreghost tool to define the shared secret in the host configuration file (default name SmHost.conf). You define the shared secret with the -sh option of the smreghost tool.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddTrustedHost (
     void*                             pSessionHandle,
     Sm_PolicyApi_TrustedHost_t*       pstructTrustedHost,
     bool                              bUpdate,
     bool                              bGenSharedSecret
);
```

Parameter	I/O	Description
pSessionHandle	I	Specifies the identifier of the session.
pstructTrustedHost	I/O	The address of a structure of type Sm_PolicyApi_TrustedHost_t. The structure is filled by a caller prior to a function call.
bUpdate	I	Specifies whether the function was called to update the existing object in the object store.
bGenSharedSecret	I	Indicates whether to generate the shared secret.

Remarks

If *bGenSharedSecret* is true, the function generates a 128-byte value and updates the *pszSecret* field of *pstructTrustedHost*. If *bGenSharedSecret* is false, the value of the shared secret for a new or updated trusted host object is taken from the *pszSecret* field of the *pstructTrustedHost* structure.

If both *bGenSharedSecret* and *bUpdate* are true, the function ignores the value specified in the *pszSecret* field of *pstructTrustedHost*, generates a new value, and updates the object in the object store and in the *pszSecret* field of *pstructTrustedHost*.

If the function generates the shared secret, you must retrieve the generated shared secret in clear text so that you can define it in the -sh option of the smreghost tool. To retrieve the shared secret, call Sm_PolicyApi_GetTrustedHost().

In past releases, agent registration with the Policy Server always used 128-byte random ASCII shared secrets. The new model makes it possible to use a user-defined string value as a shared secret. This is a potential security weakness. The administrator who chooses offline agent host configuration must create a strong shared secret and store it safely. It is strongly recommended that you call the function Sm_PolicyApi_AddTrustedHost() with the bGenSharedSecret parameter set to true. This enforces automatic generation of a hard-to-guess shared secret.

Use of this function is not required to either create a trusted host or to define the host configuration. SiteMinder automatically creates and configures the trusted host during installation, and also when you run smreghost without using the -sh option.

Returns

- Sm_PolicyApi_Success. The trusted host object was created or modified.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to a Trusted Host object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the proper privileges to add or modify a trusted host object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NotUnique. The shared secret already exists.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.
- Sm_PolicyApi_InvalidProp. Invalid shared secret value specified.
- Sm_PolicyApi_GenSharedSecretFailure. Failed to generate a 128-byte shared secret value.

Sm_PolicyApi_AddUserDirToAffiliateDomain()

Adds a user directory to an existing affiliate domain.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddUserDirToAffiliateDomain (
   void*         pSessionHandle,
   const char*   pszUserDirOid,
   const char*   pszAffiliateDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory.
pszAffiliate DomainOid	I	A null-terminated string containing the object identifier of an existing affiliate domain.

Returns

- Sm_PolicyApi_Success. The user directory was added to the affiliate domain successfully.
- Sm_PolicyApi_Failure. The user directory was not added to the affiliate domain.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add a user directory to an affiliate domain.
- Sm_PolicyApi_InvalidOid. The affiliate domain OID or user directory OID was not found during an update.
- Sm_PolicyApi_DuplicateEntry. The user directory exists in the user directory collection of the domain.

Sm_PolicyApi_AddUserDirToDomain()

Associates the directory object identified by *pszUserDirOid* with the domain identified by *pszDomainOid*. The directory object is appended to the end of the search order.

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddUserDirToDomain (
   void*          pSessionHandle,
   const char*    pszUserDirOid,
   const char*    pszDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain.

Returns

- Sm_PolicyApi_Success. The add operation was successful.
- Sm_PolicyApi_Failure. The add operation was not successful.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add a user directory to a domain.
- Sm_PolicyApi_InvalidOid. A user directory or domain OID was not found.
- Sm_PolicyApi_DuplicateEntry. The user directory exists in the user directory collection of the domain.

Sm_PolicyApi_AddUseSecureAuthPropToAffiliate()

Adds or updates the UseSecureAuthURL property to a SAML 1.x affiliate.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddUseSecureAuthPropToAffiliate(
   void*    pHandle,
   char*    pszAffiliateOID,
   bool     bUseSecureAuthURL

);
```

Parameters

pHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszAffiliateOid

[in] A pointer to the OID of an existing SMAL 1.x affiliate.

bUseSecureAuthURL

[in] A Boolean value that specifies whether to use a secure authentication URL for this affiliate.

Returns

- Sm_PolicyApi_Success. The action was completed successfully.
- Sm_PolicyApi_Failure. The action was unsuccessful.
- Sm_PolicyApi_NoSession. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator did not have sufficient access privileges.

Sm_PolicyApi_AddUsersToAffiliate()

Adds a user directory entry to an affiliate.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddUsersToAffiliate (
   void*              pSessionHandle,
   const char*        pszAffiliateOid,
   Sm_PolicyApi_User_t *pStructUsers,
   int                iPolicyFlags
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliateOid	I	A null-terminated string containing the object identifier of an existing affiliate.
pStructUsers	I	Pointer to a Sm_PolicyApi_User_t structure containing information about the user directory.
iPolicyFlags	I	A bit field that indicates whether the policy includes or excludes a user and whether the policy should be applied recursively. The bit definitions are listed in Figure 21 on page 112.

Returns

- Sm_PolicyApi_Success. The user was added was successful.
- Sm_PolicyApi_Failure. The user was not added successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add a user.
- Sm_PolicyApi_InvalidOid: The affiliate OID was not found.
- Sm_PolicyApi_DuplicateEntry. The user is already part of the affiliate.

Sm_PolicyApi_AddUsersToPolicy()

Adds a user directory entry to the policy identified by *pszPolicyOid*. Only one user specification (which may be an aggregate) can be added at a time.

Type

User and user state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddUsersToPolicy (
   void*                  pSessionHandle,
   const char*            pszPolicyOid,
   Sm_PolicyApi_User_t*   pStructUsers,
   int                    iPolicyFlags
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPolicyOid	I	A null-terminated string containing the object identifier of an existing policy to which a user is to be added.
pStructUsers	I	Pointer to a Sm_PolicyApi_User_t structure containing information about the user directory.

Parameter	I/O	Description
iPolicyFlags	I	A bit field that indicates whether the policy includes or excludes users, and whether the policy should be applied recursively. Bit definitions are listed in Policy Flags.

Returns

- Sm_PolicyApi_Success. The add was successful.
- Sm_PolicyApi_Failure. The user directory is not part of the domain user directory collection.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add users to the policy.
- Sm_PolicyApi_InvalidOid:
  - The policy OID was not found.
  - The user directory OID was not found.
- Sm_PolicyApi_DuplicateEntry. The user is already part of the policy.
- Sm_PolicyApi_InConsistentANDBitMask. An existing user policy of this particular user directory under the policy has a different value of Sm_PolicyBehavior_AND_Mask set as against the one to be added.

Sm_PolicyApi_AddUsersToSAMLServiceProvider()

Associates a user directory entry with SAML 2.0 Service Provider.

Type

SAML 2.0 Configuration function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddUsersToSAMLServiceProvider

(
   void*                          pSessionHandle,
   const char*                    pszProviderOid,
   Sm_PolicyApi_User_t*           pStructUsers,
   int                            iPolicyFlags
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszProviderOid	I	A null-terminated string containing the object identifier of an existing SAML Service Provider.
pStructUsers	I	Pointer to a Sm_PolicyApi_User_t structure containing information about the user directory.
iPolicyFlags	I	A bit field that indicates whether: ■ The policy created for the SAML Service Provider includes a user ■ The policy should be applied recursively

Returns

- Sm_PolicyApi_Success. The user was added successfully.
- Sm_PolicyApi_Failure. The user was not added successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add a user.
- Sm_PolicyApi_InvalidOid. The Service Provider OID was not found.
- Sm_PolicyApi_DuplicateEntry. The user is already part of the Service Provider.

Sm_PolicyApi_AddUsersToWSFEDResourcePartner()

Associates a user directory entry with WS-Federation Resource Partner.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddUsersToWSFEDResourcePartner (
        void* pSessionHandle,
        const char * pszProviderOid,
        Sm_PolicyApi_User_t *pStructUsers,
        int   iPolicyFlags
);
```

Parameters

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszProviderOid

[in] A null-terminated string containing the object identifier of an existing WS-Federation Resource Partner.

pStructUsers

[in] A Pointer to a Sm_PolicyApi_User_t structure containing information about the user directory.

iPolicyFlags

[in] A bit field that indicates whether the policy created for WS-Federation Resource Partner includes or excludes a user and whether the policy should be applied recursively.

Return Values

- Sm_PolicyApi_Success. The user was added successfully.
- Sm_PolicyApi_Failure. The user was not added successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add a user.
- Sm_PolicyApi_InvalidOid. The Resource Partner OID was not found.
- Sm_PolicyApi_DuplicateEntry. The user is already part of the Resource Partner.

Sm_PolicyApi_AddVariable()

Adds a variable object.

Type

Variable function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddVariable (
   void*                       pSessionHandle,
   Sm_PolicyApi_Variable_t*    pstructVariable,
   const bool                  bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructVariable	I	A pointer to a Sm_PolicyApi_Variable_t structure containing information about the variable.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The add operation was successful.
- Sm_PolicyApi_Failure. The add operation was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to add a variable.
- Sm_PolicyApi_FeatureNotSupported. The client who called this function initialized the API with a version less than SM_POLICY_API_VERSION_6_0.

Sm_PolicyApi_AddWSFEDResourcePartner()

Creates a new or update an existing WS-Federation Resource Partner object. Validation of properties (values and dependencies on other properties) is performed.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddWSFEDResourcePartner (
        void* pSessionHandle,
        Sm_PolicyApi_WSFEDResourcePartner_t* structServiceProvider,
        const bool bUpdate,
        char **pszErrMsg
);
```

Parameters

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pstructServiceProvider

[in] A pointer to a completely filled-in WS-Federation Resource Partner structure.

bUpdate

[in] A flag to indicate that if an existing object is found, update it.

pszErrMsg

[out] When a call to this function returns a value of Sm_PolicyApi_SAML_UnknownProperty, Sm_PolicyApi_MissingProperty, or Sm_PolicyApi_InvalidProp, this variable contains the name of the property which produced the error. You release the memory allocated for this variable by using a call to Sm_PolicyApi_FreeString().

Return Values

- Sm_PolicyApi_Success. The Resource Partner was created successfully.
- Sm_PolicyApi_Failure. The Resource Partner was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create an affiliate.
- Sm_PolicyApi_DomainNotFound. The affiliate domain OID was not found.
- Sm_PolicyApi_DomainNotAffiliate. The domain is not an affiliate domain.
- Sm_PolicyApi_InsufficientRPData. Required properties for configuring the Resource Partner were missing.

- Sm_PolicyApi_WSFED_UnknownProperty. An unknown property name was provided.
- Sm_PolicyApi_WSFEDRP_AssertionConsumerDefaultMissing. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_AuthenticationURLMissing. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_DomainOidMissing. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_APIDMissing. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_NameMissing. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_NameIdFormatMissing. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_NameIdTypeMissing. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_NameIdStaticMissing. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_NameIdAttrNameMissing. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_NameIdDNSpecMissing. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_ProviderIdMissing. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_ProviderIdNotUnique. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_UnSupportedSAMLVersion. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_WSFEDRP_UnknownProperty. The property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_MissingProperty. A property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_InvalidProp.The value for a provided property is invalid.

Remarks

In the event that a Resource Partner cannot be created, any objects created in the policy store must be rolled back. In addition, the prefix string *wsfed:* will be used for the Name property to differentiate between *affiliate:* and *samlsp:*.

Sm_PolicyApi_AddWSFEDScheme()

Creates a new or updates an existing SiteMinder WSFED authentication scheme (WSFED auth scheme) object. Validation of properties (values and dependencies on other properties) is performed.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddWSFEDScheme(
        void* pSessionHandle,
        Sm_PolicyApi_Scheme_t* pstructScheme,
        Sm_PolicyApi_WSFEDProviderProp_t* pProps,
        const bool bUpdate,
        char **pszErrMsg
);
```

Parameters

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pstructScheme

[in] A pointer to a completely filled-in scheme structure.

pProps

[in] A pointer to a linked list of WSFED provider properties.

bUpdate

[in] A flag to indicate that if an existing object is found, it should be updated.

pszErrMsg

[out] When a call to this function returns a value of Sm_PolicyApi_WSFED_UnknownProperty, Sm_PolicyApi_MissingProperty or Sm_PolicyApi_InvalidProp, this variable contains the name of the property which produced the error. You release the memory allocated for this variable by using a call to SmPolicyApi_FreeString().

Return Values

- Sm_PolicyApi_Success. The WSFED auth scheme was created successfully.
- Sm_PolicyApi_Failure. The WSFED auth scheme was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a WSFED auth scheme.
- Sm_PolicyApi_WSFEDAP_IncorrectParameters. Supplied WSFED Account Partner properties are incomplete or incorrect.
- Sm_PolicyApi_WSFEDAP_ProviderIdNotUnique. Supplied WSFED Account Partner ID is not unique.
- Sm_PolicyApi_WSFED_UnknownProperty. An unknown property name was provided.
- Sm_PolicyApi_MissingProperty. A property which is required, potentially due to a dependency, was not provided.
- Sm_PolicyApi_InvalidProp. The value for a provided property is invalid.

Sm_PolicyApi_ConvertFromLegacyAgent()

Converts a v4.x agent to a v5.x agent.

Calling this function is the equivalent of clearing the Support 4.x agents check box on the Agent Properties dialog box.

Type

Agent function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_ConvertFromLegacyAgent (
   void*                   pSessionHandle,
   Sm_PolicyApi_Agent_t*   pStructAgent
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pStructAgent	I	A pointer to a completely filled-in agent structure.

Returns

- Sm_PolicyApi_Success. The agent was converted successfully.
- Sm_PolicyApi_Failure. The agent conversion failed.
- Sm_PolicyApi_InvalidProp. One or more properties defined in *pStructAgent* are invalid.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to modify an agent.

Sm_PolicyApi_ConvertToLegacyAgent()

Converts a v5.x agent to a v4.x agent.

Calling this function is the equivalent of checking the Support 4.x agents check box on the Agent Properties dialog box.

Type

Agent function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_ConvertToLegacyAgent (
   void*                   pSessionHandle,
   Sm_PolicyApi_Agent_t*   pStructAgent
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pStructAgent	I	A pointer to a completely filled-in agent structure.

Returns

- Sm_PolicyApi_Success. The agent was converted successfully.
- Sm_PolicyApi_Failure. The agent conversion failed.
- Sm_PolicyApi_InvalidProp. One or more properties defined in *pStructAgent* are invalid.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to modify an agent.

Sm_PolicyApi_CreateAuthAzMap()

Creates an authentication and authorization directory mapping object.

Type

Authentication/Authorization map function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_CreateAuthAzMap (
   void*                        pSessionHandle,
   Sm_PolicyApi_AuthAzMap_t*    pAuthAzMap,
   const bool                   bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pAuthAzMap	I	The address of a pointer to information about authentication and authorization mapping.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The directory mapping object was created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a directory mapping object.
- Sm_PolicyApi_InvalidOid:
  - An authentication user directory OID was not found.
  - An authorization user directory OID was not found.
  - A directory-mapping object OID was not found.

Sm_PolicyApi_CreateCertMap()

Creates a certification mapping object.

Type

Certificate map function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_CreateCertMap (
   void*                      pSessionHandle,
   Sm_PolicyApi_CertMap_t*    pCertMap,
   const bool                 bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pCertMap	I	The address of a pointer to Sm_PolicyApi_CertMap_t, which contains information about certificate mapping.
Update	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The certificate mapping object was created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a certificate mapping object.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found or the certificate mapping object OID was not found during an update.
- Sm_PolicyApi_Invalid. The directory type is not valid.

Sm_PolicyApi_CreateODBCQueryScheme()

Creates a new ODBC Query Scheme. An ODBC query scheme is used to create an ODBC directory. The attributes of the user directory are contained in the *pstructODBCQueryScheme* structure.

If the user ODBC query scheme object exists and the *bUpdate* flag is true, the item is updated.

Type

ODBC query scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_CreateODBCQueryScheme (
   void*                             pSessionHandle,
   Sm_PolicyApi_ODBCQueryScheme_t*   pstructODBCQueryScheme,
   const bool                        bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructODBCQueryScheme	I	A pointer to a completely filled-in ODBC query scheme.
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

Returns

- Sm_PolicyApi_Success. The ODBC Query Scheme was created successfully.
- Sm_PolicyApi_Failure. The ODBC Query Scheme was not created successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create an ODBC Query Scheme.
- Sm_PolicyApi_InvalidOid. The ODBC Query Scheme OID was not found. This happens when this function is called with *bUpdate* set to true and the *pszOid* field of Sm_PolicyApi_ODBCQueryScheme_t holds the OID of the ODBC Query Scheme being updated.
- Sm_PolicyApi_NotUnique. An ODBC Query Scheme with the same name exists.
- Sm_PolicyApi_MissingProperty. One of the required fields is not set.

Sm_PolicyApi_CreateUserDir()

Creates a new SiteMinder user directory object at global scope. The attributes of the user directory are contained in the *pstructUserDir* structure.

If the user directory object exists and the *bUpdate* flag is true, the item is updated.

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_CreateUserDir (
   void*                     pSessionHandle,
   Sm_PolicyApi_UserDir_t*   pstructUserDir,
   const bool                bUpdate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructUserDir	I	A pointer to a completely filled-in user directory structure.

Parameter	I/O	Description
bUpdate	I	A flag to indicate that if an existing object is found, it should be updated.

SM_PolicyAPI_UserDir_t Field Usage

The following table shows the SM_PolicyAPI_UserDir_t field that Sm_PolicyApi_CreateUserDir() uses for different types of user directories:

Field	**User Directory Type**			
	ODBC	**LDAP**	**WinNT**	**Custom**
pszOid Object Identifier of the user directory being updated	X	X	X	X
pszName Required field.	X	X	X	X
pszDesc	X	X	X	X
pszNamespace Required field.	ODBC	LDAP	WinNT	Custom
pszServer Required field.	ODBC data source	IP address	NT Domain name	Name of shared library
pszODBCQuerySchemeOid Required field.	X			
pszSearchRoot		X		paramete r string
pszUserLookupStart		X		
pszUserLookupEnd		X		
bRequireCredentials	X	X	X	X
pszUsername Required field if *bRequireCredentials* is true.	X	X	X	X
pszPassword Required field if *bRequireCredentials* is true.	X	X	X	X

Field	**User Directory Type**			
	ODBC	**LDAP**	**WinNT**	**Custom**
nSearchResults		X		Max results
nSearchScope		X		
nSearchTimeout		X		Max time out
bSecureConnection		X		X
pszDisabledAttr	X	X		(Varies)
pszUniversalIDAttr	X	X	X	(Varies)
pszAnonymousId		X		(Varies)
pszPasswordData	X	X		(Varies)
pszPasswordAttribute	X	X		(Varies)
pszEmailAddressAttr				
pszChallengeRespAttr		X		(Varies)

Note: With custom directories, fields indicated by the word *varies* may or may not apply to the user directory object being created.

Returns

- Sm_PolicyApi_Success. The user directory was created successfully.
- Sm_PolicyApi_Failure. The user directory was not created successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to create a user directory.
- Sm_PolicyApi_InvalidOid:
  - A user directory type with the specified OID was not found.
  - A user directory OID cannot be found. This happens when this function is called with bUpdate set to true and *pszOid* holds the OID of the user directory that is being updated.
  - You are creating an ODBC user directory and the OCBC Query Scheme OID was not found.
- Sm_PolicyApi_NotUnique. A user directory of the same name exists.
- Sm_PolicyApi_MissingProperty. One of the required fields is not set.

Sm_PolicyApi_DeleteAdmin()

Deletes the administrator object identified by *pszAdminOid*.

Type

Administrator function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteAdmin (
   void*          pSessionHandle,
   const char*    pszAdminOid
)
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszAdminOid	I	A null-terminated string containing the object identifier of an existing administrator.

Returns

- Sm_PolicyApi_Success. The delete operation was successful.
- Sm_PolicyApi_Failure. The delete operation was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete an administrator object.
- Sm_PolicyApi_InvalidOid. The administrator object identifier was not found.

Sm_PolicyApi_DeleteAffiliate()

Deletes an existing affiliate.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteAffiliate (
   void*          pSessionHandle,
   const char*    pszAffiliateOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliateOid	I	A null-terminated string containing the object identifier of an existing affiliate.

Remarks

This function deletes affiliate objects that are based on the artifact profile or the POST profile (see the Sm_PolicyApi_Affiliate_t field SAMLProfile). Deletion of an affiliate object based on the POST profile requires an API version of at least SM_POLICY_API_VERSION_6_0_2. If an earlier version is involved, and the function specifies an affiliate object based on a POST profile, the request fails.

Returns

- Sm_PolicyApi_Success. The affiliate was deleted successfully.
- Sm_PolicyApi_Failure. The affiliate was not deleted successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete an affiliate.
- Sm_PolicyApi_InvalidOid. The affiliate OID was not found.

Sm_PolicyApi_DeleteAffiliateDomain()

Deletes an existing affiliate domain.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteAffiliateDomain (
   void*          pSessionHandle,
   const char*    pszAffiliateDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliate DomainOid	I	A null-terminated string containing the object identifier of an existing affiliate domain.

Returns

- Sm_PolicyApi_Success. The affiliate domain was deleted successfully.
- Sm_PolicyApi_Failure. The affiliate domain was not deleted successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete an affiliate domain.
- Sm_PolicyApi_InvalidOid. An affiliate with the specified OID was not found during an update.

Sm_PolicyApi_DeleteAgent()

Deletes the agent identified by *pszAgentOid*.

Type

Agent function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteAgent (
   void*          pSessionHandle,
   const char*    pszAgentOid
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAgentOid	I	A null-terminated string containing the object identifier of an existing agent.

Returns

- Sm_PolicyApi_Success. The agent was deleted successfully.
- Sm_PolicyApi_Failure. The agent was not deleted successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete an agent.
- Sm_PolicyApi_InvalidOid. An agent with the specified OID was not found.

Sm_PolicyApi_DeleteAgentConfig()

Deletes an agent configuration object.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteAgentConfig (
   void*            pSessionHandle,
   const char*      pszAgentConfigOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAgentConfigOid	I	Unique identifier of the agent configuration object to delete.

Returns

- Sm_PolicyApi_Success. The delete operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to an agent configuration object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to delete an agent configuration object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_DeleteAuthAzMap()

Deletes an authentication and authorization directory mapping object.

Type

Authentication/Authorization map function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteAuthAzMap (
   void*          pSessionHandle,
   const char*    pszAuthAzMapOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAuthAzMapOid	I	A null-terminated string containing the object identifier of the directory mapping object.

Returns

- Sm_PolicyApi_Success. The delete operation was successful.
- Sm_PolicyApi_Failure. The delete operation was unsuccessful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a directory mapping object.
- Sm_PolicyApi_InvalidOid. The directory-mapping object OID was not found.

Sm_PolicyApi_DeleteCertMap()

Deletes a certificate mapping object.

Type

Certificate map function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteCertMap (
   void*         pSessionHandle,
   const char*   pszCertMapOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszCertMapOid	I	A null-terminated string containing the object identifier of the certificate mapping object.

Returns

- Sm_PolicyApi_Success. The delete operation was successful.
- Sm_PolicyApi_Failure. The delete operation was unsuccessful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a certificate mapping object.
- Sm_PolicyApi_InvalidOid. A certificate mapping object with the specified OID was not found.

Sm_PolicyApi_DeleteDomain()

Deletes the domain identified by *pszDomainOid* as well as the domain's children (rules, responses, realms, and policies).

Type

Domain function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteDomain (
   void*          pSessionHandle,
   const char*    pszDomainOid
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain.

Returns

- Sm_PolicyApi_Success. The domain was deleted successfully.
- Sm_PolicyApi_Failure. The domain was not deleted successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a domain.
- Sm_PolicyApi_InvalidOid. The domain OID was not found.

Sm_PolicyApi_DeleteHostConfig()

Deletes a host configuration object from the policy store.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteHostConfig (
   void*                 pSessionHandle,
   const char*           pszHostConfigOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszHostConfigOid	I	Unique identifier of the host configuration object to delete.

Returns

- Sm_PolicyApi_Success. The delete operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to a host configuration object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to delete a host configuration object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_DeleteGroup()

Deletes the group object identified by *pszGroupOid*.

The *pszDomainOid* parameter is required by a rule group or response group. An agent group does not require a domain OID because it is not a domain-based object.

Type

Group function, global scope (agents) or domain scope (responses, rules).

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteGroup (
   void*                  pSessionHandle,
   Sm_PolicyApi_Groups_t  dwGroup,
   const char*            pszGroupOid,
   const char*            pszDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
dwGroup	I	Indicates the type of the group to be deleted.
pszGroupOid	I	A null-terminated string containing the object identifier of the group and the type indicated by *dwGroup*.

Parameter	I/O	Description
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain. Required parameter for rule or response group.

Returns

- Sm_PolicyApi_Success. The delete was successful.
- Sm_PolicyApi_Failure. The delete was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a rule group, response group, or agent group.
- Sm_PolicyApi_InvalidOid:
  - The group OID was not found.
  - The domain OID was not found (for a domain-based group).
- Sm_PolicyApi_BadGroup. The *dwGroup* parameter is not the rule, response, or agent type.

Sm_PolicyApi_DeleteODBCQueryScheme()

Deletes the ODBC query scheme identified by *pszODBCQuerySchemeOid*.

Type

ODBC query scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteODBCQueryScheme (
   void*          pSessionHandle,
   const char*    pszODBCQuerySchemeOid
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszODBCQuerySchemeOid	I	A null-terminated string containing the object identifier of an existing ODBC Query Scheme.

Returns

- Sm_PolicyApi_Success. The delete operation was successful.
- Sm_PolicyApi_Failure. The delete operation was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete an ODBC Query Scheme.
- Sm_PolicyApi_InvalidOid. The ODBC Query Scheme OID was not found.

Sm_PolicyApi_DeletePasswordPolicy()

Deletes a password policy.

Type

Password policy function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeletePasswordPolicy (
   void*         pSessionHandle,
   const char*   pszPasswordPolicyOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPasswordPolicyOid	I	A null-terminated string containing the object identifier of a password policy.

Returns

- Sm_PolicyApi_Success. The delete operation was successful.
- Sm_PolicyApi_Failure. The delete operation was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a password policy.
- Sm_PolicyApi_InvalidOid. The password policy OID was not found.

Sm_PolicyApi_DeletePolicy()

Deletes the policy identified by *pszPolicyOid*.

Type

Policy function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeletePolicy (
   void*         pSessionHandle,
   const char*   pszPolicyOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPolicyOid	I	A null-terminated string containing the object identifier of an existing policy.

Returns

- Sm_PolicyApi_Success. The policy was deleted successfully.
- Sm_PolicyApi_Failure. The policy was not deleted successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession.l There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a policy.
- Sm_PolicyApi_InvalidOid. The policy OID was not found.

Sm_PolicyApi_DeleteRealm()

Deletes the realm identified by *pszRealOid*.

Note: You cannot delete a realm while it is inked to rules.

Type

Realm function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteRealm (
   void*          pSessionHandle,
   const char*    pszRealmOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszRealmOid	I	A null-terminated string containing the object identifier of an existing realm.

Returns

- Sm_PolicyApi_Success. The realm was deleted successfully.
- Sm_PolicyApi_Failure. The realm was not deleted successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a realm.
- Sm_PolicyApi_InvalidOid. The realm OID was not found.

Sm_PolicyApi_DeleteRegistrationScheme()

Deletes a registration scheme.

Type

Registration scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteRegistrationScheme (
   void*         pSessionHandle,
   const char*   pszRegistrationSchemeOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszRegistrationOid	I	A null-terminated string containing the object identifier of the registration scheme.

Returns

- Sm_PolicyApi_Success. The registration scheme was deleted successfully.
- Sm_PolicyApi_Failure. The registration scheme was not deleted successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a registration scheme.
- Sm_PolicyApi_InvalidOid. The registration scheme OID was not found.

Sm_PolicyApi_DeleteResponse()

Deletes the response identified by *pszResponseOid* and any related response attributes.

Type

Response function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteResponse (
   void*          pSessionHandle,
   const char*    pszResponseOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszResponseOid	I	A null-terminated string containing the object identifier of an existing response.

Returns

- Sm_PolicyApi_Success. The response was deleted successfully.
- Sm_PolicyApi_Failure. The response was not deleted successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a response.
- Sm_PolicyApi_InvalidOid. The response OID was not found.

Sm_PolicyApi_DeleteRule()

Deletes the rule identified by *pszRuleOid*.

Type

Rule function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteRule (
   void*          pSessionHandle,
   const char*    pszRuleOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszRuleOid	I	A null-terminated string containing the object identifier of an existing rule.

Returns

- Sm_PolicyApi_Success. The rule was deleted successfully.
- Sm_PolicyApi_Failure. The rule was not deleted successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a rule.
- Sm_PolicyApi_InvalidOid. The rule OID was not found.

Sm_PolicyApi_DeleteSAMLAffiliation()

Deletes the SAML affiliation identified by *pszAffiliationOid*.

Type

SAML 2.0 Configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteSAMLAffiliation
(
   void*            pSessionHandle,
   const char*      pszAffiliationOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliationOid	I	A null-terminated string containing the object identifier of an existing SAML affiliation.

Returns

- Sm_PolicyApi_Success. The SAML affiliation was deleted successfully.
- Sm_PolicyApi_Failure. The SAML affiliation was not deleted successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a SAML affiliation.
- Sm_PolicyApi_InvalidOID. The SAML affiliation's object identifier was not found.

Sm_PolicyApi_DeleteSAMLServiceProvider()

Deletes the Service Provider identified by *pszProviderOid*.

Type

SAML 2.0 Configuration function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteSAMLServiceProvider(
void*            pSessionHandle,
const char*      pszProviderOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszProviderOid	I	A null-terminated string containing the object identifier of an existing Service Provider.

Returns

- Sm_PolicyApi_Success. The Service Provider was deleted successfully.
- Sm_PolicyApi_Failure. The Service Provider was not deleted successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a Service Provider.
- Sm_PolicyApi_InvalidOID. The Service Provider's object identifier was not found.

Sm_PolicyApi_DeleteScheme()

Deletes the authentication scheme identified by *pszSchemeOid*.

Type

Authentication scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteScheme (
   void*         pSessionHandle,
   const char*   pszSchemeOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszSchemeOid	I	A null-terminated string containing the object identifier of an existing authentication scheme.

Returns

- Sm_PolicyApi_Success. The scheme was deleted successfully.
- Sm_PolicyApi_Failure. The scheme was not deleted successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a scheme.
- Sm_PolicyApi_InvalidOid. The scheme object identifier was not found.

Sm_PolicyApi_DeleteTrustedHost()

Deletes an existing trusted host object from the policy store.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteTrustedHost (
   void*            pSessionHandle,
   const char*      pszTrustedHostOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszTrustedHostOid	I	Unique identifier of the trusted host object to delete.

Returns

- Sm_PolicyApi_Success. The delete operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to a trusted host object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to delete a trusted host object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_DeleteUserDir()

Deletes the user directory identified by *pszUserDirOid*.

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteUserDir (
   void*          pSessionHandle,
   const char*    pszUserDirOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory.

Returns

- Sm_PolicyApi_Success. The user directory was deleted successfully.
- Sm_PolicyApi_Failure. The user directory was not deleted successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a user directory.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.

Sm_PolicyApi_DeleteVariable()

Deletes a variable object.

Type

Variable function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteVariable (
   void*         pSessionHandle,
   const char*   pszVariableOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszVariableOid	I	A null-terminated string containing the object identifier of an existing variable.

Returns

- Sm_PolicyApi_Success. The delete operation was successful.
- Sm_PolicyApi_Failure. The delete operation was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to delete a variable.
- Sm_PolicyApi_InvalidOid. The variable object identifier was not found.
- Sm_PolicyApi_FeatureNotSupported. The client who called this function initialized the API with a version less than SM_POLICY_API_VERSION_6_0.

Sm_PolicyApi_DeleteWSFEDResourcePartner()

Deletes an existing WS-Federation Resource Partner object.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DeleteWSFEDResourcePartner (
        void* pSessionHandle,
        const char * pszProviderOid,
);
```

Parameters

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszProviderOid

[in] A null-terminated string containing the object identifier of an existing WS-Federation Resource Partner.

Return Values

- Sm_PolicyApi_Success. The Resource Partner was retrieved successfully.
- Sm_PolicyApi_Failure. The Resource Partner was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve an affiliate.
- Sm_PolicyApi_InvalidOID. The Resource Partner OID was not found.

Sm_PolicyApi_DisableUser()

Disables a user for the reason Sm_Api_Disabled_AdminDisabled. It does not change other concurrent disabling reasons.

To make this function work, the attribute for tracking disabled users must be set in the user directory (the *pszDisabledAttr* field of Sm_PolicyApi_UserDir_t). You can also set the attribute using the Policy Server UI.

Type

User and user state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_DisableUser (
   void*        pSessionHandle,
   const char*  pszUserDirOid,
   const char*  pszUserDN,
   char**       pszErrMsg
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of the user directory where the user may be found.
pszUserDN	I	The distinguished name of the user to be disabled.
pszErrMsg	O	String containing an error message if the user is not disabled successfully. You release the memory allocated for this variable by calling Sm_PolicyApi_FreeString().

Returns

- Sm_PolicyApi_Success. The user was disabled successfully.
- Sm_PolicyApi_Failure. The user was not disabled successfully or memory could not be allocated to the error message string.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to disable a user.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.

Sm_PolicyApi_EnableUser()

Enables a user by clearing all the disabled bits. It does not clear the qualifier bit Sm_Api_Disabled_PWMustChange.

Type

User and user state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_EnableUser (
   void*           pSessionHandle,
   const char*     pszUserDirOid,
   const char*     pszUserDN,
   char**          pszErrMsg
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of the user directory where the user may be found.
pszUserDN	I	The distinguished name of the user to be enabled.

Parameter	I/O	Description
pszErrMsg	O	String containing an error message if the user is not enabled successfully. You release the memory allocated for this variable by calling Sm_PolicyApi_FreeString().

Returns

- Sm_PolicyApi_Success. The user was enabled successfully.
- Sm_PolicyApi_Failure. The user was not enabled successfully or memory could not be allocated to the error message string.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to enable a user.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.

Sm_PolicyApi_FlushRealm()

Flushes the specified realm identified by *pszRealmOid* from a resource cache.

Type

Utility function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_FlushRealm (
   void*          pSessionHandle,
   const char*    pszRealmOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszRealmOid	I	A null-terminated string containing the object identifier of a realm.

Returns

- Sm_PolicyApi_Success. The flush operation was successful.
- Sm_PolicyApi_Failure. The flush operation was not successful.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to flush a realm.
- Sm_PolicyApi_InvalidOid. The realm OID was not found.

Sm_PolicyApi_FlushUser()

Flushes a user from a User Cache.

Type

Utility function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_FlushUser (
   void*         pSessionHandle,
   const char*   pszUserDirOid,
   const char*   pszUserDN,
   char**        pszErrMsg
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory.
pszUserDN	I	A null-terminated string containing the name of a user in an existing user directory who is to be flushed from the user cache.
pszErrMsg	O	Error message returned by the Policy Management API. You release the memory allocated for this variable by using a call to Sm_PolicyApi_FreeString().

Returns

- Sm_PolicyApi_Success. The flush operation was successful.
- Sm_PolicyApi_Failure. The flush operation was not successful or memory could not be allocated to the error message string.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to flush a user.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.

Sm_PolicyApi_FreeMemory()

Note: This function is deprecated for clients starting at SM_POLICY_API_VERSION_6_0. Instead, use Sm_PolicyApi_FreeMemoryEx().

Call Sm_PolicyApi_FreeMemory() to free memory that was allocated by the Policy Management API. This includes the Sm_PolicyApi_Server_t structures allocated as part of the Sm_PolicyApi_HostConfig_t structure.

Type

Utility function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_FreeMemory (void* pMem);
```

Parameter	I/O	description
pMem	I	A void pointer to memory that was allocated by the Policy Management API**.**

Returns

Sm_PolicyApi_Success or Sm_PolicyApi_Failure.

Sm_PolicyApi_FreeMemoryEx()

Call Sm_PolicyApi_FreeMemoryEx() to free memory that was allocated by the Policy Management API. This includes the Sm_PolicyApi_Server_t structures allocated as part of the Sm_PolicyApi_HostConfig_t structure.

Note: Clients starting at version SM_POLICY_API_VERSION_6_0 must use this function instead of Sm_PolicyApi_FreeMemory().

Type

Utility function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_FreeMemoryEx (
   void* pInitHandle
   void* pMem
);
```

Parameter	**I/O**	**description**
pInitHandle	I	A void pointer that points to memory that was allocated by the Policy Management API**.**
pMem	I	A void pointer that points to memory that was allocated by the Policy Management API**.**

Returns

- Sm_PolicyApi_Success. The operation was successful.
- Sm_PolicyApi_Failure. The operation failed.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.

Sm_PolicyApi_FreeString()

Frees a single string that was allocated by the Policy Management API.

Type

Utility function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_FreeString(
   char* pszString
);
```

Parameter	I/O	Description
pszString	I	A pointer to a null-terminated string.

Returns

This function always returns Sm_PolicyApi_Success.

Sm_PolicyApi_FreeStringArray()

Frees string arrays that were allocated by the Policy Management API.

Type

Utility function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_FreeStringArray (
   char* pszStringArray[]
);
```

Parameter	I/O	Description
pszStringArray	I	A pointer to an array of pointers that point to null-terminated strings.

Returns

This function always returns Sm_PolicyApi_Success.

Sm_PolicyApi_GetAdmin()

Gets the contents of the administrator object identified by *szAdminOid*.

The results of this function are returned in a structure referenced by *ppstructAdmin*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Administrator function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAdmin (
   void*                   pSessionHandle,
   const char*             pszAdminOid,
   Sm_PolicyApi_Admin_t**  ppstructAdmin
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAdminOid	I	A null-terminated string containing the object identifier of an existing administrator.
ppstructAdmin	O	The address of a pointer to an administrator structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_Admin_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an administrator object.
- Sm_PolicyApi_InvalidOid. The administrator object ID was not found.

Sm_PolicyApi_GetAdminByName()

Gets the contents of the administrator object identified by *szAdminName*.

The results of this function are returned in a structure referenced by *ppstructAdmin*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Administrator function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAdminByName (
   void*                  pSessionHandle,
   const char*            pszAdminName,
   Sm_PolicyApi_Admin_t**   ppstructAdmin
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAdminName	I	A null-terminated string containing the name of an existing administrator.
ppstructAdmin	O	The address of a pointer to an administrator structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_Admin_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an administrator object.
- Sm_PolicyApi_NotFound. The administrator name was not found.

Sm_PolicyApi_GetAffiliate()

Gets an affiliate.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAffiliate (
   void*                         pSessionHandle,
   const char*                   pszAffiliateOid,
   Sm_PolicyApi_Affiliate_t**    ppstructAffiliate
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliateOid	I	A null-terminated string containing the object identifier of an existing affiliate.
ppstruct Affiliate	O	The address of a pointer to an affiliate structure.

Remarks

This function retrieves affiliate objects that are based on the artifact profile or the POST profile (see the Sm_PolicyApi_Affiliate_t field SAMLProfile). Retrieval of an affiliate object based on the POST profile requires an API version of at least SM_POLICY_API_VERSION_6_0_2. If an earlier version is involved, and the function specifies an affiliate object based on a POST profile, the request fails.

Returns

- Sm_PolicyApi_Success. The affiliate was retrieved successfully.
- Sm_PolicyApi_Failure. The affiliate was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve an affiliate.
- Sm_PolicyApi_InvalidOID. The affiliate OID was not found.

Sm_PolicyApi_GetAffiliateByName()

Gets an affiliate.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAffiliateByName (
   void*                        pSessionHandle,
   const char*                  pszAffiliateOid,
   const char*                  pszAffiliateName,
   Sm_PolicyApi_Affiliate_t**   ppstructAffiliate
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliateOid	I	A null-terminated string containing the object identifier of an existing affiliate.
pszAffiliateName	I	A null-terminated string containing the name of an existing affiliate.
ppstruct Affiliate	O	The address of a pointer to an affiliate structure.

Remarks

This function retrieves affiliate objects that are based on the artifact profile or the POST profile (see the Sm_PolicyApi_Affiliate_t field SAMLProfile). Retrieval of an affiliate object based on the POST profile requires an API version of at least SM_POLICY_API_VERSION_6_0_2. If an earlier version is involved, and the function specifies an affiliate object based on a POST profile, the request fails.

Returns

- Sm_PolicyApi_Success. The affiliate was retrieved successfully.
- Sm_PolicyApi_Failure. The affiliate was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve an affiliate.
- Sm_PolicyApi_InvalidOID. The affiliate OID was not found.
- Sm_PolicyApi_NotFound. The affiliate name was not found.

Sm_PolicyApi_GetAffiliateDomain()

Gets an affiliate domain.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAffiliateDomain (
   void*                             pSessionHandle,
   const char*                       pszAffiliateDomainOid,
   Sm_PolicyApi_AffiliateDomain_t**  ppstructAffiliateDomain
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszAffiliate DomainOid	I	A null-terminated string containing the object identifier of an existing affiliate domain.
ppstruct AffiliateDomain	O	The address of a pointer to an affiliate domain structure.

Returns

- Sm_PolicyApi_Success. The affiliate domain was retrieved successfully.
- Sm_PolicyApi_Failure. The affiliate domain was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an affiliate domain.
- Sm_PolicyApi_DomainNotFound. The affiliate domain OID was not found.
- Sm_PolicyApi_DomainNotAffiliate. The domain is not an affiliate domain.

Sm_PolicyApi_GetAffiliateDomainByName()

Gets an affiliate domain.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAffiliateDomainByName (
   void*                            pSessionHandle,
   const char*                      pszAffiliateDomainName,
   Sm_PolicyApi_AffiliateDomain_t** ppstructAffiliateDomain
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszAffiliate DomainName	I	A null-terminated string containing the name of an existing affiliate domain.
ppstruct AffiliateDomain	O	The address of a pointer to an affiliate domain structure.

Returns

- Sm_PolicyApi_Success. The affiliate domain was retrieved successfully.
- Sm_PolicyApi_Failure. The affiliate domain was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an affiliate domain.
- Sm_PolicyApi_NotFound. The affiliate domain name was not found.
- Sm_PolicyApi_DomainNotAffiliate. The domain is not an affiliate domain.

Sm_PolicyApi_GetAffiliateDomainObjects()

Gets affiliate domain objects.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAffiliateDomainObjects (
   void*                         pSessionHandle,
   const char*                   pszAffiliateDomainOid,
   const Sm_PolicyApi_Objects_t  nObjectId,
   Sm_PolicyApi_Oid_t**          ppstructObject
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszAffiliate DomainOid	I	A null-terminated string containing the object identifier of an existing affiliate domain.
nObjectId	I	The type of domain object to retrieve. Valid types are for affiliate, admin, and user directory objects only.
ppstructObject	O	The address of a pointer to a Sm_PolicyApi_Oid_t structure

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. The get operation was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get affiliate domain objects.
- Sm_PolicyApi_InvalidOID. The affiliate domain OID was not found.

Sm_PolicyApi_GetAffiliateDomainUserDirSearchOrder()

Gets the user directory search order for an affiliate domain.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAffiliateDomainUserDirSearchOrder (
   void*        pSessionHandle,
   const char*  pszAffiliateDomainOid,
   char**       pszArray[]
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliate DomainOid	I	A null-terminated string containing the object identifier of an existing affiliate domain.
pszArray	O	The returned array of user directory OIDs of the requested objects.

Returns

- Sm_PolicyApi_Success. The function successfully returned the user directory search order.
- Sm_PolicyApi_Failure. The function did not successfully return the user directory search order.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get user directory search order in an affiliate domain.
- Sm_PolicyApi_InvalidOID. The affiliate domain OID was not found.

Sm_PolicyApi_GetAffiliatedSAMLAuthSchemes()

Retrieves all the SAML authentication schemes associated with the specified SAML affiliation.

Type

SAML 2.0 Configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAffiliatedSAMLAuthSchemes
(
   void*                        pSessionHandle,
   const char*                  pszAffiliationOid,
   Sm_PolicyApi_Scheme_t**      ppstructSchemes
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliationOid	I	A null-terminated string containing the object identifier of an existing SAML affiliation.
ppstructSchemes	O	The address of a pointer to SAML authentication scheme structures.

Returns

- Sm_PolicyApi_Success. The SAML authentication schemes were retrieved successfully.
- Sm_PolicyApi_Failure. The SAML authentication schemes were not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve SAML authentication schemes.
- Sm_PolicyApi_InvalidOID. The SAML affiliation object identifier was not found.

Sm_PolicyApi_GetAffiliatedSAMLServiceProviders()

Retrieves all the Service Providers associated with the specified SAML affiliation.

Type

SAML 2.0 Configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAffiliatedSAMLServiceProviders
(
   void*                        pSessionHandle,
   const char*                  pszAffiliationOid,
   Sm_PolicyApi_SAMLSP_t**      ppstructSAMLSPs
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliationOid	I	A null-terminated string containing the object identifier of an existing SAML affiliation.
ppstructSAMLSPs	O	The address of a pointer to Service Provider structures.

Returns

- Sm_PolicyApi_Success. Service Providers were retrieved successfully.
- Sm_PolicyApi_Failure. Service Providers were not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve Service Providers.
- Sm_PolicyApi_InvalidOID. The SAML affiliation object identifier was not found.

Sm_PolicyApi_GetAffiliateUsers()

Gets the user directory entries for an affiliate.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAffiliateUsers (
   void*              pSessionHandle,
   const char*        pszAffiliateOid,
   const char*        pszUserDirOid,
   Sm_PolicyApi_User_t **ppStructUsers
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliateOid	I	A null-terminated string containing the object identifier of an existing affiliate.
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory.
ppStructUsers	O	The address of a pointer to a linked list of user structures.

Returns

- Sm_PolicyApi_Success. The user was retrieved successfully.
- Sm_PolicyApi_Failure. The user was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve users.
- Sm_PolicyApi_InvalidOID. The affiliate OID or user directory OID was not found.

Sm_PolicyApi_GetAgent()

Gets the contents of the agent identified by *pszAgentOid*. The results of this function are returned in a structure referenced by *ppstructAgent*.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Agent function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAgent (
   void*                    pSessionHandle,
   const char*              pszAgentOid,
   Sm_PolicyApi_Agent_t**   ppstructAgent
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAgentOid	I	A null-terminated string containing the object identifier of an existing agent.
ppstructAgent	O	The address of a pointer to an agent structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Agent_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an agent.
- Sm_PolicyApi_InvalidOid. An agent with the specified OID not found.

Sm_PolicyApi_GetAgentByName()

Gets the contents of the agent identified by *szAgentName*. The results of this function are returned in a structure referenced by *ppstructAgent*.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Agent function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAgentByName (
   void*                    pSessionHandle,
   const char*              pszAgentName,
   Sm_PolicyApi_Agent_t**   ppstructAgent
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAgentName	I	A null-terminated string containing the name of an existing agent.
ppstructAgent	O	The address of a pointer to an agent structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Agent_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an agent.
- Sm_PolicyApi_NotFound. An agent with the specified name not found.

Sm_PolicyApi_GetAgentConfig()

Retrieves an existing agent configuration object.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAgentConfig (
   void*                            pSessionHandle,
   const char*                      pszAgentConfigOid,
   Sm_PolicyApi_AgentConfig_t**     ppstructAgentConfig
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAgentConfigOid	I	Unique identifier of the agent configuration object to retrieve.
ppstructAgentConfig	O	Address of a pointer to a structure that defines the agent configuration object. The function allocates the structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to an agent configuration object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to retrieve an agent configuration object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.

Sm_PolicyApi_GetAgentConfigByName()

Retrieves an existing agent configuration object by *pszAgentConfigName*.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAgentConfigByName (

   void*                            pSessionHandle,
   const char*                      pszAgentConfigName,
   Sm_PolicyApi_AgentConfig_t**     ppstructAgentConfig
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAgentConfigName	I	Unique name of the agent configuration object to retrieve.
ppstructAgentConfig	O	Address of a pointer to a structure that defines the agent configuration object. The function allocates the structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_NotFound. The unique name does not correspond to an agent configuration object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to retrieve an agent configuration object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_GetAgentConfigAssociations()

Retrieves a list of configuration parameters for an agent configuration object.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAgentConfigAssociations (
   void*                             pSessionHandle,
   const char*                       pszAgentConfigOid,
   Sm_PolicyApi_Association_t**      ppstructAssociations
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAgentConfigOid	I	Unique identifier of the agent configuration object.
ppstructAssociations	O	A list of name/value pairs representing the configuration parameters for the agent configuration object.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to an agent configuration object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to retrieve configuration parameters for an agent configuration object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_GetAgentType()

Gets the contents of the agent type object identified by *pszAgentTypeOid*. The results of this function will be returned in a structure referenced by *ppstructAgentType*.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Agent function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAgentType (
   void*                         pSessionHandle,
   const char*                   pszAgentTypeOid,
   Sm_PolicyApi_AgentType_t**    ppstructAgentType
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAgentTypeOid	I	A null-terminated string containing the object identifier of an existing agent type.
ppstructAgentType	O	The address of a pointer to an agent type structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_AgentType_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an agent type.
- Sm_PolicyApi_InvalidOid. An agent type with the specified OID was not found.

Sm_PolicyApi_GetAgentTypeByName()

Gets the contents of the agent type object identified by *pszAgentTypeName*. The results of this function will be returned in a structure referenced by *ppstructAgentType*.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Agent function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAgentTypeByName (
   void*                         pSessionHandle,
   const char*                   pszAgentTypeName,
   Sm_PolicyApi_AgentType_t**    ppstructAgentType
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAgentTypeName	I	A null-terminated string containing the name of an existing agent type.
ppstructAgentType	O	The address of a pointer to an agent type structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_AgentType_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an agent type.
- Sm_PolicyApi_NotFound.An agent type with the specified name was not found.

Sm_PolicyApi_GetAgentTypeAttr()

Returns one or all agent attributes:

- If *pszOid* is of type Sm_PolicyApi_AgentType_Prop, the function returns all agent attributes.
- If *pszOid* is of type Sm_PolicyApi_AgentTypeAttr_Prop, the function returns an agent type attribute object.

Type

Agent function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAgentTypeAttr (
   void*                             pSessionHandle,
   const char*                       pszOid,
   Sm_PolicyApi_AgentTypeAttr_t**    ppstructAgentTypeAttr
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszOid	I	A null-terminated string containing the object identifier of an existing agent type or agent type attribute.
ppstructAgentTypeAttr	O	The address of a pointer to an agent type attribute structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_AgentTypeAttr_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an agent type or an agent type attribute.
- Sm_PolicyApi_InvalidOid. An agent type or an agent type attribute with the specified OID was not found.

Sm_PolicyApi_GetAgentTypeAttrByName()

Returns an agent type attribute object by name.

Type

Agent function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAgentTypeAttrByName (
   void*                             pSessionHandle,
   const char*                       szAgentTypeName,
   Sm_PolicyApi_AgentTypeAttr_t**    ppstructAgentTypeAttr
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
szAgentTypeName	I	A null-terminated string containing the name of an existing agent type attribute.
ppstructAgentTypeAttr	O	The address of a pointer to an agent type attribute structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_AgentTypeAttr_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an agent type or an agent type attribute.
- Sm_PolicyApi_NotFound. An agent type attribute with the specified name was not found.

Sm_PolicyApi_GetAllAffiliateAttributes()

Gets all the attributes for an affiliate.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAllAffiliateAttributes (
   void*                            pSessionHandle,
   const char*                      pszAffiliateOid,
   Sm_PolicyApi_AffiliateAttr_t**   ppstructAffiliateAttr
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliateOid	I	A null-terminated string containing the object identifier of an existing affiliate.
ppstruct AffiliateAttr	O	The address of a pointer to an affiliate attribute structure.

Returns

- Sm_PolicyApi_Success. The affiliate attribute was retrieved successfully.
- Sm_PolicyApi_Failure. The affiliate attribute was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve affiliate attributes.
- Sm_PolicyApi_InvalidOID. The affiliate OID was not found.

Sm_PolicyApi_GetAllAffiliates()

Gets all affiliates in the specified affiliate domain.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAllAffiliates (
   void*                          pSessionHandle,
   const char*                    pszAffiliateDomainOid,
   Sm_PolicyApi_Affiliate_t**     ppstructAffiliates
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliate DomainOid	I	A null-terminated string containing the object identifier of an existing affiliate domain.
ppstruct Affiliates	O	The address of a pointer to an affiliate structure.

Returns

- Sm_PolicyApi_Success. The affiliates were retrieved successfully.
- Sm_PolicyApi_Failure. The affiliates were not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve affiliates.
- Sm_PolicyApi_InvalidOID. The affiliate domain OID was not found.

Sm_PolicyApi_GetAllSAMLAffiliations()

Retrieves all existing SAML affiliation objects.

Type

SAML 2.0 Configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAllSAMLAffiliations
(
   void*                              pSessionHandle,
   Sm_PolicyApi_SAMLAffiliation_t**   ppstructAffiliations
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
ppstruct Affiliations	O	The address of a pointer to SAML affiliation structures.

Returns

- Sm_PolicyApi_Success. The SAML affiliations were retrieved successfully.
- Sm_PolicyApi_Failure. The SAML affiliations were not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve SAML affiliations.

Sm_PolicyApi_GetAllSAMLSchemeAttributes()

Returns a linked list of all attributes defined for a SAML Requester.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddAttributeToSAMLScheme(
   void*                                pHandle,
   const Sm_PolicyApi_Scheme_t*         pstructScheme,
   const Sm_PolicyApi_SAMLRequesterAttr_t*     pAttr
);
```

Parameter	I/O	Description
pHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructScheme	I	A pointer to a completely filled-in scheme structure.
pAttr	O	A pointer that will be assigned to the first Sm_PolicyApi_SAMLRequesterAttr_t structure in the returned list of attributes.

Returns

- Sm_PolicyApi_Success. The SAML Requester attributes were returned successfully.
- Sm_PolicyApi_Failure. The SAML Requester attributes were not returned successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to obtain SAML Requester attributes.
- Sm_PolicyApi_SAMLIDP_IncorrectParameters. Supplied SAML provided properties are incomplete or incorrect.
- Sm_PolicyApi_DuplicateAttribute. An attribute already exists with the same Name and NameFormat.

Sm_PolicyApi_GetAllSAMLServiceProviders()

Retrieves all the Service Providers in the specified affiliate domain.

Type

SAML 2.0 Configuration function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAllSAMLServiceProviders
(
   void*                              pSessionHandle,
   const char*                        pszAffiliateDomainOid,
   Sm_PolicyApi_SAMLSP_t**            ppstructSAMLSPs
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliate DomainOid	I	A null-terminated string containing the object identifier of an existing affiliate domain.

Parameter	I/O	Description
ppstructSAMLSPs	O	The address of a pointer to Service Provider structures.

Returns

- Sm_PolicyApi_Success. Service Providers were retrieved successfully.
- Sm_PolicyApi_Failure. Service Providers were not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior
- to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve Service Providers.
- Sm_PolicyApi_InvalidOID. The affiliate domain object identifier was not found.

Sm_PolicyApi_GetAllSAMLSPAssertionConsumerService()

Retrieves a list of all the Assertion Consumer Services currently in the policy store.

Note: The list of Assertion Consumer Service structures that is returned by this function must be freed using the Sm_PolicyApi_FreeMemory function.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAllSAMLSPAssertionConsumerService(
   void* pSessionHandle,
   const Sm_PolicyApi_SAMLSPAssertionConsumerService_t**
               ppstructSAMLSPAssertionConsumerService,
   const char* pszSAMLSPOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
ppstructSAMLSPAssertionConsumerServiceAttr	I	A pointer to an array of Assertion Consumer Service structures.
pszSAMLSPOid	I	A pointer to a string containing the OID of the Service Provider.

Returns

- Sm_PolicyApi_Success. The array of Assertion Consumer Services was retrieved successfully.
- Sm_PolicyApi_Failure - The array of Assertion Consumer Services was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle - There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession - There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege - The administrator does not have the privilege to remove Attributes from a SAML Service Provider.

Sm_PolicyApi_GetAllSAMLSPAttributes()

Retrieves all the attributes associated with the specified Service Provider.

Type

SAML 2.0 Configuration function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAllSAMLSPAttributes
(
   void*                              pHandle,
   const char*                        pszSAMLSPOid,
   Sm_PolicyApi_SAMLSPAttr_t**        ppstructSAMLSPAttr
);
```

Parameter	I/O	Description
pHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszSAMLSPOid	I	A null-terminated string containing the object identifier of an existing Service Provider.
ppstructSAML SPAttr	O	The address of a pointer to attribute structures.

Returns

- Sm_PolicyApi_Success. The attributes were retrieved successfully.
- Sm_PolicyApi_Failure. The attributes were not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve attributes from a SAML Service Provider.

Sm_PolicyApi_GetAllWSFEDResourcePartners()

Retrieves all existing Resource Partner objects.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAllWSFEDResourcePartners (
        void* pSessionHandle,
        const char * pszAffiliateDomainOid,
        Sm_PolicyApi_WSFEDResourcePartner_t** ppResourcePartners
);
```

Parameters

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszAffiliateDomainOid

[in] A null-terminated string containing the object identifier of an existing affiliate domain.

ppResourcePartners

[out] The address of the pointer to WS-Federation Resource Partner structures.

Return Values

- Sm_PolicyApi_Success. The Resource Partner was retrieved successfully.
- Sm_PolicyApi_Failure. The Resource Partner was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve an affiliate.
- Sm_PolicyApi_InvalidOID. The affiliate domain OID was not found.

Sm_PolicyApi_GetAuthAzMap()

Gets the contents of an authentication and authorization directory mapping object.

Type

Authentication/Authorization map function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetAuthAzMap (
   void*                         pSessionHandle,
   const char*                   pszAuthAzMapOid,
   Sm_PolicyApi_AuthAzMap_t**    ppAuthAzMap
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAuthAzMapOid	I	A null-terminated string containing the object identifier of the directory mapping object.
pszAuthAzMap	O	The address of a pointer to an Sm_PolicyApi_AuthAzMap_t structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_AuthAzMap_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a directory mapping object.
- Sm_PolicyApi_InvalidOid. The directory-mapping OID was not found.

Sm_PolicyApi_GetCertMap()

Retrieves a certificate mapping object.

Type

Certificate map function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetCertMap (
   void*                        pSessionHandle,
   const char*                  pszCertMapOid,
   Sm_PolicyApi_CertMap_t**     ppCertMap
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszCertMapOid	I	A null-terminated string containing the object identifier of the certificate mapping object.
ppCertMap	O	The address of a pointer to an Sm_PolicyApi_CertMap_t structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_CertMap_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a certificate mapping object.
- Sm_PolicyApi_InvalidOid. A certificate mapping object matching the specified OID was not found.

Sm_PolicyApi_GetChildren()

Builds the hierarchical realm and rule tree.

This function retrieves a list of OIDs. The OIDs are of type realms, or realms and rules. If the function is called with a domain OID, it retrieves a list of top-level realm OIDs. If the function is called with a realm OID, it retrieves a list of realm and rule OIDs under that realm. The *iObjectId* field in Sm_PolicyApi_Oid_t specifies the type of OID.

If there are no children for the domain or realm OID, the function returns an empty list.

Type

Realm function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetChildren (
   void*                  pSessionHandle,
   const char*            pszOid,
   Sm_PolicyApi_Oid_t**   ppStructObject
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszOid	I	A null-terminated string containing the object identifier of a domain or a realm.
ppStructObject	O	The address of a pointer to a Sm_PolicyApi_Oid_t structure.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_Oid_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a list of realms and rules.
- Sm_PolicyApi_InvalidOid. The domain or realm OID was not found.
- Sm_PolicyApi_NoChildren. The domain has no realms or the realm has no realms or rules.

Sm_PolicyApi_GetDirectoryContents()

Retrieves a linked list of user structures (referenced by *ppStructUsers*) for a particular user directory.

The granularity of the response to this function is governed by the following registry entry:

```
HKLM\SOFTWARE\Wow6423Node\Netegrity\SiteMinder\CurrentVersion\Ds\ClassFilters
```

Free the memory allocated for the returned structures by calling Sm_PolicyApi_FreeMemoryEx().

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetDirectoryContents (
   void*                    pSessionHandle,
   const char*              pszUserDirOid,
   Sm_PolicyApi_User_t**    ppStructUserSpec
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory.
ppStructUserSpec	O	The address of a pointer to a user structure.

Returns

- Sm_PolicyApi_Success. The retrieve operation was successful.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.

Sm_PolicyApi_GetDisabledUserState()

Retrieves the disabled state of a user.

To make this function work successfully, you must specify a directory attribute to track disabled users. This attribute is specified in the disabled flag of the user directory. The disabled reasons are enumerated in Sm_Api_DisabledReason_t, which is defined in SmApi.h.

Type

User and user state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetDisabledUserState (
   void*                        pSessionHandle,
   const char*                  pszUserDirOid,
   const char*                  pszUserDN,
   Sm_Api_DisabledReason_t*     nDisabledReason,
   char**                       pszErrMsg
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory where the user may be found.
pszUserDN	I	A null-terminated string containing the distinguished name of a user whose disabled state is to be changed.
nDisabledReason	O	Reason for disabling or enabling a user. Reasons are enumerated in Sm_Api_DisabledReason_t, which is defined in SmApi.h.
pszErrMsg	O	The error message is held in this string if the retrieval was not successful. You release the memory allocated for this variable by calling Sm_PolicyApi_FreeString().

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure:
  - The disable state was not retrieved.
  - Memory could not be allocated to *pszErrMsg*.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get disabled user state.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.

Sm_PolicyApi_GetDomain()

Gets the contents of the domain identified by *pszDomainOid*. The results of this function are returned in a structure referenced by *ppstructDomain*.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Domain function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetDomain (
   void*                      pSessionHandle,
   const char*                pszDomainOid,
   Sm_PolicyApi_Domain_t**    ppstructDomain
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain.
ppstructDomain	O	The address of a pointer to a domain structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_Domain_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a domain.
- Sm_PolicyApi_InvalidOid. The domain OID was not found.

Sm_PolicyApi_GetDomainByName()

Gets the contents of the domain identified by *pszDomainName*. The results of this function are returned in a structure referenced by *ppstructDomain*.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Domain function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetDomainByName (
   void*                      pSessionHandle,
   const char*                pszDomainName,
   Sm_PolicyApi_Domain_t**    ppstructDomain
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszDomainName	I	A null-terminated string containing the name of an existing domain.
ppstructDomain	O	The address of a pointer to a domain structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_Domain_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a domain.
- Sm_PolicyApi_NotFound. The domain name was not found.

Sm_PolicyApi_GetDomainObjects()

Retrieves the OIDs of domain objects for a given object type within the domain identified by *pszDomainOid*. The returned values are contained in the Sm_PolicyApi_Oid_t structure.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Domain function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetDomainObjects (
   void*                          pSessionHandle,
   const char*                    pszDomainOid,
   const Sm_PolicyApi_Objects_t  nObjectId,
   Sm_PolicyApi_Oid_t**           ppstructObject
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain.
nObjectId	I	The type of domain object to retrieve, as enumerated in Sm_PolicyApi_Objects_t.
ppstructObject	O	The address of a pointer to a Sm_PolicyApi_Oid_t structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_Oid_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get domain-based objects.
- Sm_PolicyApi_InvalidOid. The domain OID was not found.

Sm_PolicyApi_GetGlobalObjects()

Retrieves the object identifiers of global objects of a specified type. Beginning at SiteMinder v6.0, this function will accept rule, policy, and response properties as global objects, and will return global rules, policies, and responses.

The returned values are contained in structure Sm_PolicyApi_Oid_t. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Note: In releases prior to SiteMinder v4.5, the functionality provided by Sm_PolicyApi_GetGlobalObjects() was provided by Sm_PolicyApi_GetGlobalObjectNames().

Type

General object function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetGlobalObjects (
   void*                         pSessionHandle,
   const Sm_PolicyApi_Objects_t   nObjectId,
   Sm_PolicyApi_Oid_t**          ppstructObject
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
nObjectId	I	The type of global object to retrieve. Object types are enumerated in Sm_PolicyApi_Objects_t.
ppstructObject	O	The address of a pointer to a Sm_PolicyApi_Oid_t structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Oid_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get global objects.
- Sm_PolicyApi_Invalid. A non-global object identifier was specified.

Sm_PolicyApi_GetGlobalPolicyByName()

Gets a specified global policy.

The results of this function are returned in a structure referenced by *ppstructPolicy*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Administrator function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetGlobalPolicyByName (
   void*                    pHandle,
   const char*              szPolicyName,
   Sm_PolicyApi_Policy_t**  ppstructPolicy
);
```

Parameter	**I/O**	**Description**
pHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
szPolicyName	I	Unique name that corresponds to a global policy.
ppstructPolicy	O	The address of a pointer to a policy structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Policy_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a policy.
- Sm_PolicyApi_NotFound. The policy name was not found.

Sm_PolicyApi_GetGlobalResponseByName()

Gets the specified global response.

The results of this function are returned in a structure referenced by *ppstructResponse*.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Administrator function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetGlobalResponseByName (
   void*                      pHandle,
   const char*                szResponseName,
   Sm_PolicyApi_Response_t**   ppstructReponse
);
```

Parameter	I/O	Description
pHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
szResponseName	I	Unique name that corresponds to a global response.
ppstructResponse	O	The address of a pointer to a response structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Response_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a scheme.
- Sm_PolicyApi_NotFound. The global response name was not found.

Sm_PolicyApi_GetGlobalRuleByName()

Gets the specified global rule.

The results of this function are returned in a structure referenced by *ppstructRule*.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Administrator function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetGlobalRuleByName (
   void*                    pHandle,
   const char*              szRuleName,
   Sm_PolicyApi_Rule_t**   ppstructRule
);
```

Parameter	I/O	Description
pHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
szRuleName	I	Unique name that corresponds to a global rule.
ppstructRule	O	The address of a pointer to a response structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Rule_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a scheme.
- Sm_PolicyApi_NotFound. The global rule name was not found.

Sm_PolicyApi_GetGroup()

Gets the contents of the group object identified by pszGroupOid. The results of this function are returned in a structure referenced by *ppStructGroup*.

The *pszDomainOid* parameter is required by a rule or response group. An agent group does not require a domain OID because it is not a domain-based object.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Group function, global scope (agents) or domain scope (responses, rules).

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetGroup (
   void*                   pSessionHandle,
   Sm_PolicyApi_Groups_t   dwGroup,
   const char*             pszGroupOid,
   const char*             pszDomainOid,
   Sm_PolicyApi_Group_t**  ppStructGroup
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
dwGroup	I	The type of group to be retrieved.

Parameter	I/O	Description
pszGroupOid	I	A null-terminated string containing the object identifier of the group object being retrieved.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain. Required parameter for rule or response groups.
ppStructGroup	O	The address of a pointer to a group structure.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. The get was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get the contents of a group.
- Sm_PolicyApi_InvalidOid:
  - The group OID was not found.
  - The domain OID was not found (for a domain-based group).
- Sm_PolicyApi_BadGroup. The *dwGroup* parameter is not the rule, response, or agent type.

Sm_PolicyApi_GetGroupByName()

Gets the contents of the group object identified by *pszGroupName*. The results of this function are returned in a structure referenced by *ppStructGroup*.

The *pszDomainOid* parameter is required by a rule or response group. An agent group does not require a domain OID because it is not a domain-based object.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Group function, global scope (agents) or domain scope (responses, rules).

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetGroupByName (
   void*                    pSessionHandle,
   Sm_PolicyApi_Groups_t    dwGroup,
   const char*              pszGroupName,
   const char*              pszDomainOid,
   Sm_PolicyApi_Group_t**   ppStructGroup
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
dwGroup	I	The type of group to be retrieved.
pszGroupOid	I	A null-terminated string containing the object identifier of the group object being retrieved.
pszDomainName	I	A null-terminated string containing the name of an existing domain. Required parameter for rule or response groups.
ppStructGroup	O	The address of a pointer to a group structure.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. The get was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get the contents of a group.
- Sm_PolicyApi_InvalidOid. The domain OID was not found (for a domain-based group).
- SmPolicyApi_Notfound. The group name was not found.
- Sm_PolicyApi_BadGroup. The *dwGroup* parameter is not the rule, response, or agent type.

Sm_PolicyApi_GetGroupOids()

Retrieves the object identifiers contained within a group object. The results of this function are returned in a structure referenced by *ppStructObjects*.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

pszDomainOid is required by a rule group or response group. An agent group does not require a domain OID because it is not a domain-based object.

Note: At releases prior to SiteMinder v4.5, the functionality provided by Sm_PolicyApi_GetGroupOids() was provided by Sm_PolicyApi_GetGroupNames().

Type

Group function, global scope (agents) or domain scope (responses, rules).

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetGroupOids (
   void*                   pSessionHandle,
   Sm_PolicyApi_Groups_t   dwGroup,
   const char*             pszGroupOid,
   const char*             pszDomainOid,
   Sm_PolicyApi_Oid_t**    ppStructObjects
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
dwGroup	I	Indicates the type of the group from which to retrieve object identifiers.
pszGroupOid	I	A null-terminated string containing the object identifier of a group of the type indicated by *dwGroup*.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain. Required parameter for rule or response group.
ppStructObjects	O	A pointer to the address of an object structure.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. The get was not successful or memory could not be allocated to Sm_PolicyApi_Oid_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get group OIDs.
- Sm_PolicyApi_InvalidOid:
  - The group OID was not found.
  - The domain OID was not found (for a domain-based group).
- Sm_PolicyApi_BadGroup. Parameter *dwGroup* is not the rule, response, or agent type.

Sm_PolicyApi_GetHostConfig()

Retrieves an existing host configuration object.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetHostConfig (
   void*                              pSessionHandle,
   const char*                        pszHostConfigOid,
   Sm_PolicyApi_HostConfig_t**        ppstructHostConfig
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszHostConfigOid	I	Unique identifier of the host configuration object to retrieve.

Parameter	I/O	Description
ppstructHostConfig	O	Address of a pointer to a structure that defines the host configuration object. The function allocates the structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to a host configuration object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to get a host configuration object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_GetHostConfigByName()

Retrieves an existing host configuration object.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetHostConfigByName (
   void*                              pSessionHandle,
   const char*                        pszHostConfigName,
   Sm_PolicyApi_HostConfig_t**        ppstructHostConfig
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszHostConfigName	I	Unique name of the host configuration object to retrieve.

Parameter	I/O	Description
ppstructHostConfig	O	Address of a pointer to a structure that defines the host configuration object. The function allocates the structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_NotFound. The unique name does not correspond to a host configuration object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to get a host configuration object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyAPI_GetMessageConsumerPluginFromSAML1xScheme()

Gets a message consumer plugin setting from a SAML 1.x authentication scheme.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetMessageConsumerPluginFromSAML1xScheme(
   void*    pHandle,
   char*    pszSchemeOID,
   char**   pluginClass,
   char**   pluginParam
);
```

Parameters

phandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszSchemeOid

[in] A pointer to the OID of the authentication scheme that is being updated.

pluginClass

[out] A pointer to the name of the plugin class to be read in from the authentication scheme,

pluginParam

[out] A pointer to the parameters of the plugin class to be read in from the authentication scheme.

Returns

- Sm_PolicyApi_Success. The action was completed successfully.
- Sm_PolicyApi_Failure. The action was unsuccessful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_SchemeNotFound. The authentication scheme corresponding to the OID was not found, or was not a SAML 1.x authentication scheme.
- Sm_PolicyApi_InvalidOid. The OID of the authentication scheme is NULL.

Sm_PolicyApi_GetODBCQueryScheme()

Gets the contents of the ODBC Query Scheme object identified by *pszODBCQuerySchemeOid*.

The results of this function are returned in a structure referenced by *pstructODBCQueryScheme*. Free the memory allocated for this structure calling Sm_PolicyApi_FreeMemoryEx().

Type

ODBC query scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetODBCQueryScheme (
   void*                              pSessionHandle,
   const char*                        pszODBCQuerySchemeOid,
   Sm_PolicyApi_ODBCQueryScheme_t**   pstructODBCQueryScheme
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszODBCQuerySchemeOid	I	A null-terminated string containing the object identifier of an existing ODBC Query Scheme.
pstructODBCQueryScheme	O	The address of a pointer to a ODBC query scheme.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_ODBCQueryScheme_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an ODBC Query Scheme.
- Sm_PolicyApi_InvalidOid. The ODBC Query Scheme OID was not found.

Sm_PolicyApi_GetODBCQuerySchemeByName()

Gets the contents of the ODBC Query Scheme object identified by *pszODBCQuerySchemeName*.

The results of this function are returned in a structure referenced by *pstructODBCQueryScheme*. Free the memory allocated for this structure calling Sm_PolicyApi_FreeMemoryEx().

Type

ODBC query scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetODBCQuerySchemeByName (
   void*                              pSessionHandle,
   const char*                        pszODBCQuerySchemeName,
   Sm_PolicyApi_ODBCQueryScheme_t**   pstructODBCQueryScheme
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszODBCQuerySchemeName	I	A null-terminated string containing the name of an existing ODBC Query Scheme.
pstructODBCQueryScheme	O	The address of a pointer to a ODBC query scheme.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_ODBCQueryScheme_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get an ODBC Query Scheme.
- Sm_PolicyApi_NotFound. The ODBC Query Scheme name was not found.

Sm_PolicyApi_GetOneTimeUsePropFromAffiliate()

Retrieves the value of the OneTimeUse property for an assertion in a SAML 1.x affiliate.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetOneTimeUsePropFromAffiliate(
   void*    pHandle,
   char*    pszAffiliateOID,
   bool     &bOneTimeUse

);
```

Parameters

pHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszAffiliateOid

[in] A pointer to the OID of an existing SMAL 1.x affiliate.

bOneTimeUse

[in] A Boolean value that specifies whether an assertion is used only once in this affiliate.

Returns

- Sm_PolicyApi_Success. The action was completed successfully.
- Sm_PolicyApi_Failure. The action was unsuccessful.
- Sm_PolicyApi_NoSession. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator did not have sufficient access privileges.

Sm_PolicyApi_GetPasswordMsg()

Gets information about an error that occurred during an attempt to validate a new password.

Call this function when Sm_PolicyApi_SetPassword() returns the error code Sm_PolicyApi_InvalidPasswordSyntax.

Type

User and user state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetPasswordMsg (
   void*                               pSessionHandle,
   const char*                         pszPasswordMsg,
   unsigned int*                       nVersion,
   unsigned int*                       nMsgId,
   unsigned int*                       nArgs,
   Sm_PolicyApi_PasswordMsgField_t**   ppStructMsgField
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPasswordMsg	I	Encoded error message returned in the *pszErrMsg* parameter of the function Sm_PolicyApi_SetPassword().
nVersion	O	The version of the SiteMinder password services.
nMsgId	O	The password services message identifier retrieved for the encoded error message. Message identifiers are enumerated in Sm_PolicyApi_PasswordMsgId_t.

Parameter	I/O	Description
nArgs	O	The number of fields in the Sm_PolicyApi_PasswordMsgField_t structure referenced by *ppStructMsgField*.
ppStructMsgField	O	The address of a pointer to an Sm_PolicyApi_PasswordMsgField_t structure containing the password error message information.

Remarks

SiteMinder password services errors contain a unique message identifier. A message identifier (enumerated in Sm_PolicyApi_PasswordMsgId_t) is returned in the *nMsgId* parameter of the function Sm_PolicyApi_GetPasswordMsg().

Further, each message identifier is associated with additional information about the error. This additional information is referenced by a field identifier (enumerated in Sm_PolicyApi_PasswordMsgFieldId_t).

For example, suppose an administrator sets the minimum length of a password to seven characters. If a user tries to validate a new password containing just six characters, the following events can occur:

1. As a result of the failed password validation attempt, Sm_PolicyApi_SetPassword() returns Sm_PolicyApi_InvalidPasswordSyntax.
2. The Policy Management application responds by calling Sm_PolicyApi_GetPasswordMsg(), passing the encoded error message returned from Sm_PolicyApi_SetPassword() in the *pszPasswordMsg* parameter.
3. When Sm_PolicyApi_GetPasswordMsg() returns:
   - *nMsgId* contains Sm_PolicyApi_PasswordMsgId_PasswordShort
   - *ppStructMsgField* points to an Sm_PolicyApi_PasswordMsgField_t structure containing the following additional information about the password error:

Field	Value and Meaning
iStructId	Sm_PolicyApi_PasswordMsgField_ID. The error information relates to a password policy.
nId	Sm_PolicyApi_PasswordMsgField_Id_Min. The error violates a minimum character requirement for the password (the minimum password length).

Field	Value and Meaning
nType	Sm_PolicyApi_FieldType_Int. The error description is an integer, so *nValue* is filled and *pszMsg* is not.
pszMsg	"". Not applicable to this error.
nValue	7. The minimum password length. The length of the requested password was less than this value.

Sm_PolicyApi_GetPasswordPolicy()

Gets the contents of a password policy object.

Type

Password policy function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetPasswordPolicy (
   void*                          pSessionHandle,
   const char*                    pszPasswordPolicyOid,
   Sm_PolicyApi_PasswordPolicy_t** ppstructPasswordPolicy
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPassword PolicyOid	I	A null-terminated string containing the object identifier of the password policy.
ppstruct PasswordPolicy	O	The address of a pointer to Sm_PolicyApi_PasswordPolicy_t.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_PasswordPolicy_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a password policy.
- Sm_PolicyApi_InvalidOid. The password policy OID was not found.

Sm_PolicyApi_GetPasswordPolicyByName()

Gets the contents of a password policy object.

Type

Password policy function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetPasswordPolicyByName (
   void*                            pSessionHandle,
   const char*                      pszPasswordPolicyName,
   Sm_PolicyApi_PasswordPolicy_t** ppstructPasswordPolicy
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPassword PolicyName	I	A null-terminated string containing the name of the password policy.
ppstruct PasswordPolicy	O	The address of a pointer to Sm_PolicyApi_PasswordPolicy_t.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_PasswordPolicy_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a password policy.
- Sm_PolicyApi_NotFound. The password policy name was not found.

Sm_PolicyApi_GetPolicy()

Gets the contents of the policy identified by *pszPolicyOid*.

The results of this function are returned in a structure referenced by *ppstructPolicy*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Policy function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetPolicy (
   void*                    pSessionHandle,
   const char*              pszPolicyOid,
   Sm_PolicyApi_Policy_t**  ppstructPolicy
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPolicyOid	I	A null-terminated string containing the object identifier of an existing policy.
ppstructPolicy	O	The address of a pointer to a policy structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Policy_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a policy.
- Sm_PolicyApi_InvalidOid. The policy OID was not found.

Sm_PolicyApi_GetPolicyByName()

Gets the contents of the policy identified by *pszPolicyName* and the corresponding *pszDomainOid* of the Domain in which the policy exists.

The results of this function are returned in a structure referenced by *ppstructPolicy*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Policy function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetPolicyByName (
   void*                     pSessionHandle,
   const char*               szDomainOid,
   const char*               pszPolicyName,
   Sm_PolicyApi_Policy_t**   ppstructPolicy
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
szDomainOid	I	A null-terminated string containing the object identifier of an existing domain.
pszPolicyName	I	A null-terminated string containing the name of an existing policy.

Parameter	I/O	Description
ppstructPolicy	O	The address of a pointer to a policy structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Policy_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a policy.
- Sm_PolicyApi_InvalidOid. The Domain OID was not found.
- Sm_PolicyApi_NotFound. The policy name was not found.

Sm_PolicyApi_GetPolicyLinks()

Returns a linked list of all of the policy links that are associated with the policy identified by *pszPolicyOid*.

The linked list returned is referenced by the *ppstructPolicyLink* structure. Free the memory allocated to these structures by calling Sm_PolicyApi_FreeMemoryEx().

Type

Policy function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetPolicyLinks (
   void*                          pSessionHandle,
   const char*                    pszPolicyOid,
   Sm_PolicyApi_PolicyLink_t**   ppstructPolicyLink
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszPolicyOid	I	A null-terminated string containing the object identifier of an existing policy.
ppstructPolicyLink	O	The address of a policy link structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory is not allocated to Sm_PolicyApi_PolicyLink_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a list of policy links.
- Sm_PolicyApi_InvalidOid. The policy OID was not found.

Sm_PolicyApi_GetPolicyUsers()

Returns a linked list of Sm_PolicyApi_User_t structures corresponding to the users who are associated with the policy identified by *pszPolicyOid* and who optionally belong to the user directory identified by *pszUserDirOid*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

User and user state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetPolicyUsers (
   void*                    pSessionHandle,
   const char*              pszPolicyOid,
   const char*              pszUserDirOid,
   Sm_PolicyApi_User_t**    ppStructUsers
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszPolicyOid	I	A null-terminated string containing the object identifier of an existing policy.
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory. This parameter is optional. If it is empty, the function returns all the policy users under the specified policy. If it is specified, the function returns policy users who belong to this directory under the specified policy.
ppStructUsers	O	The address of a pointer to a linked list of user structures.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_User_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get users from the policy.
- Sm_PolicyApi_InvalidOid:
    - The policy OID was not found.
    - The user directory OID was not found.

Sm_PolicyApi_GetRealm()

Gets the contents of the realm identified by *pszRealmOid*.

The results of this function are returned in a structure referenced by *ppstructRealm*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Realm function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetRealm (
   void*                    pSessionHandle,
   const char*              pszRealmOid,
   Sm_PolicyApi_Realm_t**   ppstructRealm
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszRealmOid	I	A null-terminated string containing the object identifier of an existing realm.
ppstructRealm	O	The address of a pointer to a realm structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_Realm_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a realm.
- Sm_PolicyApi_InvalidOid. The realm OID was not found.

Sm_PolicyApi_GetRealmByName()

Gets the contents of the realm identified by *pszRealmName*, and the corresponding *pszDomainOrParentRealmOid* of the Domain in which the Realm exists, or the Oid of the Parent Realm in the case of a child Realm.

The results of this function are returned in a structure referenced by *ppstructRealm*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Realm function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetRealmByName (
   void*                    pSessionHandle,
   const char*              pszRealmOid,
   Sm_PolicyApi_Realm_t**   ppstructRealm
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
szDomainOr ParentRealm Oid	I	A null-terminated string containing the object identifier of an existing Domain or Realm
pszRealmName	I	A null-terminated string containing the name of an existing realm.
ppstructRealm	O	The address of a pointer to a realm structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_Realm_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a realm.
- Sm_PolicyApi_InvalidOid. The Domain or parent Realm OID was not found.
- Sm_PolicyApi_NotFound The Realm name was not found.

Sm_PolicyAPI_GetRedirectURLFromSAML1xScheme()

Retrieves a redirect URL setting from a SAML 1.x authentication scheme.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetRedirectURLFromSAML1xScheme(
        void*        pSessionHandle,
        const char*  pszSchemeOid,
        int          iTypeURL,
        char**       URL,
        int          &redirectMode
);
```

Parameters

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszSchemeOid

[in] A null-terminated string containing the object identifier of the authentication scheme being updated.

iTypeUrl

[in] An integer specifying the type of redirect URL, defined in Sm_PolicyAPI_SAML1_STATUS_REDIRECT_URL_TYPE_t as follows:

- Sm_PolicyAPI_SAML1_STATUS_REDIRECT_URL_USER_NOT_FOUND_TYPE = 0
- Sm_PolicyAPI_SAML1_STATUS_REDIRECT_URL_INVALID_SSO = 1
- Sm_PolicyAPI_SAML1_STATUS_REDIRECT_URL_UNACCEPTABLE_USER_CREDENTIALS = 2

URL

[out] A pointer to the redirect URL from the authentication scheme

redirectMode

[out] An integer specifying the input redirect mode, which is either 0 for 302 No Data, or 1 for Http-Post.

Return Values

- Sm_PolicyApi_Success. The action was completed successfully.
- Sm_PolicyApi_Failure. The action was unsuccessful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_SchemeNotFound. The authentication scheme corresponding to the OID was not found, or was not a SAML 1.x authentication scheme.

- Sm_PolicyApi_InvalidOid. The OID of the authentication scheme is NULL.

Sm_PolicyApi_GetRegistrationScheme()

Gets a registration scheme.

Type

Registration scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetRegistrationScheme (
   void*                          pSessionHandle,
   const char*                    pszRegistrationSchemeOid,
   Sm_PolicyApi_RegistrationScheme_t**
                                  ppstructRegistrationScheme
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszRegistration SchemeOid	I	A null-terminated string containing the object identifier of the registration scheme.
ppstruct Registration Scheme	O	The address of a pointer to Sm_PolicyApi_RegistrationScheme_t.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_RegistrationScheme_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a registration scheme.
- Sm_PolicyApi_InvalidOid. A registration scheme OID was not found.

Sm_PolicyApi_GetRegistrationSchemeByName()

Gets a registration scheme.

Type

Registration scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetRegistrationSchemeByName (
   void*                          pSessionHandle,
   const char*                    pszRegistrationSchemeName,
   Sm_PolicyApi_RegistrationScheme_t**
                                  ppstructRegistrationScheme
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszRegistration SchemeName	I	A null-terminated string containing the name of the registration scheme.
ppstruct Registration Scheme	O	The address of a pointer to Sm_PolicyApi_RegistrationScheme_t.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_RegistrationScheme_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a registration scheme.
- Sm_PolicyApi_NotFound. A registration scheme name was not found.

Sm_PolicyApi_GetRegularExpressions()

Gets a list of regular expressions belonging to the referenced password policy. Implemented only if the session's version is set to SM_POLICY_API_VERSION_6_0.

Type

Regular Expression function.

Syntax

```
int SM_EXTERN    Sm_PolicyApi_GetRegularExpressions (
   void*                               pSessionHandle,
   const char*                         pszPasswordPolicyOid,
   Sm_PolicyApi_RegularExpression_t**  ppstructRegExpr
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPassword PolicyOid	I	The OID of the password policy.
ppstructRegExpr	O	Pointer to a linked list of regular expressions belonging to the referenced password policy.

Returns

- Sm_PolicyApi_Success. The regular expression list was retrieved successfully.
- Sm_PolicyApi_Failure. The regular expression list was not retrieved successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get regular expressions.
- Sm_PolicyApi_InvalidOid: The password policy OID was not found.

Sm_PolicyApi_GetResponse()

Gets the contents of the response identified by *pszResponseOid*.

The results of this function are returned in a structure referenced by *ppstructResponse*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Response function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetResponse (
   void*                     pSessionHandle,
   const char*               pszResponseOid,
   Sm_PolicyApi_Response_t**  ppstructResponse
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszResponseOid	I	A null-terminated string containing the object identifier of an existing response.
ppstructResponse	O	The address of a pointer to a response structure.

Returns

- Sm_PolicyApi_Success. The get operation was completed successfully.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Response_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a response.
- Sm_PolicyApi_InvalidOid. The response OID was not found.

Sm_PolicyApi_GetResponseByName()

Gets the contents of the response identified by *pszResponseName* and the corresponding *pszDomainOid* of the Domain in which the response exists.

The results of this function are returned in a structure referenced by *ppstructResponse*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Response function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetResponseByName (
   void*                      pSessionHandle,
   const char*                szDomainOid,
   const char*                pszResponseName,
   Sm_PolicyApi_Response_t**  ppstructResponse
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
szDomainOid		A null-terminated string containing the object identifier of an existing Domain.
pszResponseName	I	A null-terminated string containing the name of an existing response.
ppstructResponse	O	The address of a pointer to a response structure.

Returns

- Sm_PolicyApi_Success. The get operation was completed successfully.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Response_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a response.
- Sm_PolicyApi_InvalidOid. The Domain OID was not found.
- Sm_PolicyApi_NotFound. The response name was not found.

Sm_PolicyApi_GetResponseAttrs()

Retrieves a linked list of the response attributes that are associated with the response identified by *pszResponseOid*.

The linked list that is returned is referenced by the *ppstructResponseAttr* structure. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Response function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetResponseAttrs (
   void*                          pSessionHandle,
   const char*                    pszResponseOid,
   Sm_PolicyApi_ResponseAttr_t**   ppstructResponseAttr
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszResponseOid	I	A null-terminated string containing the object identifier of an existing response.

Parameter	I/O	Description
ppstructResponseAttr	O	A pointer to a response attribute structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_ResponseAttr_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a list of response attributes.
- Sm_PolicyApi_InvalidOid. The response OID was not found.

Sm_PolicyApi_GetRule()

Gets the contents of the rule identified by *pszRuleOid*.

The results are returned in a structure referenced by *ppstructRule*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Rule function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetRule (
   void*                    pSessionHandle,
   const char*              pszRuleOid,
   Sm_PolicyApi_Rule_t**    ppstructRule
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszRuleOid	I	A null-terminated string containing the object identifier of an existing rule.

Parameter	I/O	Description
ppstructRule	O	The address of a pointer to a Sm_PolicyApi_Rule_t structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Rule_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a rule.
- Sm_PolicyApi_InvalidOid. The rule OID was not found.

Sm_PolicyApi_GetRuleByName()

Gets the contents of the rule identified by *pszRuleName*.

The results are returned in a structure referenced by *ppstructRule*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Rule function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetRuleByName (
   void*                     pSessionHandle,
   const char*               szRealmOid,
   const char*               pszRuleName,
   Sm_PolicyApi_Rule_t**     ppstructRule
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
szRealmOid	I	A null-terminated string containing the object identifier of an existing Realm.

Parameter	I/O	Description
pszRuleName	I	A null-terminated string containing the name of an existing rule.
ppstructRule	O	The address of a pointer to a Sm_PolicyApi_Rule_t structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Rule_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a rule.
- Sm_PolicyApi_InvalidOid. The Realm OID was not found.
- Sm_PolicyApi_NotFound. The rule name was not found.

Sm_PolicyApi_GetSAMLAffiliation()

Retrieves the SAML affiliation specified by its object identifier in the policy store.

Type

SAML 2.0 Configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetSAMLAffiliation
(
   void*                             pSessionHandle,
   const char*                       pszAffiliationOid,
   Sm_PolicyApi_SAMLAffiliation_t**  ppstructAffiliation
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszAffiliationOid	I	A null-terminated string containing the policy store object identifier of an existing SAML affiliation.
ppstruct Affiliation	O	The address of a pointer to a SAML affiliation structure.

Returns

- Sm_PolicyApi_Success. The SAML affiliation was retrieved successfully.
- Sm_PolicyApi_Failure. The SAML affiliation was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve a SAML affiliation.
- Sm_PolicyApi_InvalidOID. The SAML affiliation OID was not found.

Sm_PolicyApi_GetSAMLAffiliationById()

Retrieves the SAML affiliation specified by its unique affiliation identifier (URI).

Type

SAML 2.0 Configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetSAMLAffiliationById
(
   void*                                pSessionHandle,
   const char*                          pszAffiliationID,
   Sm_PolicyApi_SAMLAffiliation_t**     ppstructAffiliation
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszAffiliationID	I	A null-terminated string containing the unique affiliation identifier of an existing SAML affiliation. The affiliation identifier is specified in SAML_KEY_AFFILIATION_ID.
ppstruct Affiliation	O	The address of a pointer to a SAML affiliation structure.

Returns

- Sm_PolicyApi_Success. The SAML affiliation was retrieved successfully.
- Sm_PolicyApi_Failure. The SAML affiliation was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve a SAML affiliation.

Sm_PolicyApi_GetSAMLScheme()

Retrieves information about a SAML 2.0 authentication scheme and the metadata properties of the associated Identity Provider.

Type

SAML 2.0 Configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetSAMLScheme
(
   void*                               pHandle,
   const char*                         pszSchemeOid,
   Sm_PolicyApi_Scheme_t**             ppstructScheme,
   Sm_PolicyApi_SAMLProviderProp_t**   ppProps
);
```

Parameter	I/O	Description
pHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

Parameter	I/O	Description
pszSchemeOid	I	A null-terminated string containing the object identifier of an existing SAML authentication scheme.
ppstructScheme	O	The address of a pointer to a SAML authentication scheme structure.
ppProps	O	The address of a pointer to a SAML 2.0 metadata properties structure. For information about these properties, see SAML 2.0 Authentication Scheme Properties.

Returns

- Sm_PolicyApi_Success. The authentication scheme was retrieved successfully.
- Sm_PolicyApi_Failure. The authentication scheme was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior
- to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve an authentication scheme.
- Sm_PolicyApi_InvalidOID. The authentication scheme OID was not found.

Sm_PolicyApi_GetSAMLServiceProvider()

Retrieves the Service Provider specified by its object identifier in the policy store.

Type

SAML 2.0 Configuration function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetSAMLServiceProvider
(
   void*                              pSessionHandle,
   const char*                        pszProviderOid,
   Sm_PolicyApi_SAMLSP_t**            pstructSAMLSP
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszProviderOid	I	A null-terminated string containing the policy store object identifier of an existing Service Provider.
ppstructSAMLSP	O	The address of a pointer to a Service Provider structure.

Returns

- Sm_PolicyApi_Success. The Service Provider was retrieved successfully.
- Sm_PolicyApi_Failure. The Service Provider was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior
- to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve a Service Provider.
- Sm_PolicyApi_InvalidOID. The Service Provider OID was not found.
- Sm_PolicyApi_DomainNotAffiliate. The domain is not an affiliate domain.

Sm_PolicyApi_GetSAMLServiceProviderById()

Retrieves the Service Provider specified by its unique provider identifier.

Type

SAML 2.0 Configuration function, domain scope

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetSAMLServiceProvider
(
   void*                            pSessionHandle,
   const char*                      pszProviderId,
   Sm_PolicyApi_SAMLSP_t**          pstructSAMLSP
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszProviderId	I	A null-terminated string containing the unique provider identifier of an existing Service Provider. The provider identifier is specified in SAML_KEY_SPID.
ppstructSAMLSP	O	The address of a pointer to a Service Provider structure.

Returns

- Sm_PolicyApi_Success. The Service Provider was retrieved successfully.
- Sm_PolicyApi_Failure. The Service Provider was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior
- to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve a Service Provider.
- Sm_PolicyApi_InvalidOID. The Service Provider OID was not found.
- Sm_PolicyApi_DomainNotAffiliate. The domain is not an affiliate domain.

Sm_PolicyApi_GetSAMLServiceProviderUsers()

Retrieves the user directory entries associated with the specified Service Provider.

Type

SAML 2.0 Configuration function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetSAMLServiceProviderUsers
(
   void*                         pSessionHandle,
   const char*                   pszProviderOid,
   const char*                   pszUserDirOid,
   Sm_PolicyApi_User_t**         ppStructUsers
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszProviderOid	I	A null-terminated string containing the policy store object identifier of an existing Service Provider.
pszUserDirOid	I	A null-terminated string containing the policy store object identifier of an existing user.
ppStructUsers	O	The address of a pointer to user structures.

Returns

- Sm_PolicyApi_Success. The users were retrieved successfully.
- Sm_PolicyApi_Failure. The users were not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve users.
- Sm_PolicyApi_InvalidOid. The affiliate OID was not found.

Sm_PolicyApi_GetScheme()

Gets the contents of the authentication scheme identified by *pszSchemeOid*. The results of this function are returned in a structure referenced by *ppstructScheme*.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Authentication scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetScheme (
   void*                     pSessionHandle,
   const char*               pszSchemeOid,
   Sm_PolicyApi_Scheme_t**   ppstructScheme
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszSchemeOid	I	A null-terminated string containing the object identifier of an existing scheme.
ppstructScheme	O	The address of a pointer to a scheme structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Scheme_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a scheme.
- Sm_PolicyApi_InvalidOid. The scheme OID was not found.

Sm_PolicyApi_GetSchemeByName()

Gets the contents of the authentication scheme identified by *pszSchemeName*. The results of this function are returned in a structure referenced by *ppstructScheme*.

Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Authentication scheme function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetSchemeByName (
   void*                    pSessionHandle,
   const char*              pszSchemeName,
   Sm_PolicyApi_Scheme_t**  ppstructScheme
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszSchemeName	I	A null-terminated string containing the name of an existing scheme.
ppstructScheme	O	The address of a pointer to a scheme structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Scheme_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a scheme.
- Sm_PolicyApi_NotFound. The scheme name was not found.

Sm_PolicyApi_GetSharedSecretPolicy()

Gets the current SharedSecretPolicy.

Returns the current shared secret policy object. There will always be exactly one such object, so it is not necessary to specify its OID.

Type

Agent configuration.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetSharedSecretPolicy (
   void*                                pSessionHandle,
   Sm_PolicyApi_SharedSecretPolicy_t**  ppstructSecretPolicy
);
```

Parameter	I/O	Description
pSessionHandle	I	The current Policy API session handle.
ppstructSecret Policy	I/O	Address of pointer to shared secret policy structure

Returns

- Sm_PolicyApi_Success. The operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidHandle. The session handle was invalid.
- Sm_PolicyApi_NoSession. The user is not logged in.
- Sm_PolicyApi_NoPrivilege. User does not have rights to manage objects.

Sm_PolicyApi_GetTargetConfigFromSAML1xScheme

Retrieves a target configuration setting from a SAML 1.x authentication scheme.

Syntax

The Sm_PolicyApi_GetTargetConfigFromSAML1xScheme function has the following syntax:

```
int SM_EXTERN Sm_PolicyApi_GetTargetConfigFromSAML1xScheme(
        void*        pHandle,
        const char*  pszSchemeOid,
        char**       pszDefaultTarget,
        int*         iQPOverridesTarget
);
```

Parameters

The Sm_PolicyApi_GetTargetConfigFromSAML1xScheme function accepts the following parameters:

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszSchemeOid

[in] A null-terminated string containing the object identifier of the authentication scheme being updated.

pszDefaultTarget

[out] Specifies the target configuration. The calling program must free up this memory by calling the Sm_PolicyApi_FreeMemory() function.

iQPOverridesTarget

[out] Specifies the value of the 'Query parameter override Default Target' check box. The calling program is responsible for passing allocated memory.

Return Values

The Sm_PolicyApi_GetTargetConfigFromSAML1xScheme function returns one of the following values:

- Sm_PolicyApi_Success. The action was completed successfully.
- Sm_PolicyApi_Failure. The action was unsuccessful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_SchemeNotFound. The authentication scheme corresponding to the OID was not found, or was not a SAML 1.x authentication scheme.

- Sm_PolicyApi_InvalidOid. The OID of the authentication scheme is NULL.

Sm_PolicyApi_GetTrustedHost()

Retrieves an existing trusted host object.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetTrustedHost (
    void*                           pSessionHandle,
    const char*                     pszTrustedHostOid,
    Sm_PolicyApi_TrustedHost_t**    ppstructTrustedHost
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszTrustedHostOid	I	Unique identifier of the trusted host object to retrieve.
ppstructTrustedHost	O	Address of a pointer to a structure that defines the trusted host object. The function allocates the structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to a trusted host object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to get a trusted host object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_GetTrustedHostByName()

Retrieves an existing trusted host object.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetTrustedHostByName (
   void*                           pSessionHandle,
   const char*                     pszTrustedHostName,
   Sm_PolicyApi_TrustedHost_t**    ppstructTrustedHost
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszTrustedHostName	I	Unique name of the trusted host object to retrieve.
ppstructTrustedHost	O	Address of a pointer to a structure that defines the trusted host object. The function allocates the structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_NotFound. The unique name does not correspond to a trusted host object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to get a trusted host object.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_GetUseSecureAuthPropFromAffiliate()

Retrieves the value of the UseSecureAuthURL property from a SAML 1.x affiliate.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetUseSecureAuthPropFromAffiliate(
   void*    pHandle,
   char*    pszAffiliateOID,
   bool     &bUseSecureAuthURL

);
```

Parameters

pHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszAffiliateOid

[in] A pointer to the OID of an existing SMAL 1.x affiliate.

bUseSecureAuthURL

[in] A Boolean value that specifies whether to use a secure authentication URL for this affiliate.

Returns

- Sm_PolicyApi_Success. The action was completed successfully.
- Sm_PolicyApi_Failure. The action was unsuccessful.
- Sm_PolicyApi_NoSession. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator did not have sufficient access privileges.

Sm_PolicyApi_GetUserContext()

Allows callers of the Policy Management API to access user context information.

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetUserContext (
   void                          *pSessionHandle,
   const char                    *lpszUserDirOid,
   const char                    *lpszUserPath,
   const char                    *lpszSessionID,
   Sm_PolicyApi_UserContext_t    **ppPolicyApiUserContext);
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
lpszUserDirOid	I	A null-terminated string containing the object identifier of the user directory for the user specified in *lpszUserPath*.
lpszUserPath	I	The distinguished name of the user.
lpszSessionID	I	A unique identifier of the session. After a successful login, the session ID is returned in the *lpszSessionId* field of the structure Sm_AgentApi_Session_t. If the session ID is not known, assign an empty string ("") to this parameter.
ppPolicyApi UserContext	O	The user context information that SiteMinder passes to the function.

Remarks

This function allows an application to access information about a user without having to connect to the underlying user directory. To retrieve the user information, the application calls the functions in the Sm_Api_UserContext_t structure, which is returned in ppPolicyApiUserContext.For example:

- Calling fGetProp or fSetProp to get or set a user attribute.
- Calling fAuthenticate to verify a user's password.
- Calling fGetProp to search the directory and examine the results to verify that the correct user has been found.
- Calling fGetProp to retrieve an attribute for a user who is not the logged-in user.

Important! After calling Sm_GetUserContext(), release the allocated memory by calling Sm_PolicyApi_FreeMemoryEx(). Otherwise, you can possibly see some Policy Server performance degradation.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. The get operation was not successful.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_Invalid. The specified user or directory was not valid.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a user directory.

Sm_PolicyApi_GetUserDir()

Gets the contents of the user directory identified by *pszUserDirOid*.

The results of this function are returned in a structure referenced by *ppstructUserDir*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetUserDir (
   void*                    pSessionHandle,
   const char*              pszUserDirOid,
   Sm_PolicyApi_UserDir_t**   ppstructUserDir
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory.
ppstructUserDir	O	Address of a pointer to a user directory structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_UserDir_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a user directory.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.

Sm_PolicyApi_GetUserDirByName()

Gets the contents of the user directory identified by *pszUserDirName*

The results of this function are returned in a structure referenced by *ppstructUserDir*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetUserDirByName (
   void*                    pSessionHandle,
   const char*              pszUserDirName,
   Sm_PolicyApi_UserDir_t**   ppstructUserDir
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirName	I	A null-terminated string containing the name of an existing user directory.
ppstructUserDir	O	Address of a pointer to a user directory structure.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated for Sm_PolicyApi_UserDir_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get a user directory.
- Sm_PolicyApi_NotFound. The user directory name was not found.

Sm_PolicyApi_GetUserDirCapabilities()

Retrieves the user directory capabilities.

The user directory capabilities are enumerated in Sm_DirectoryCapability_t in SmApi.h.

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetUserDirCapabilities (
   void*            pSessionHandle,
   const char*      pszUserDirOid,
   unsigned long*   pCapabilities
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of the user directory where the lookup has to be performed.
pCapabilities	O	The address of a pointer to an unsigned long that will hold information about directory capability. Directory capabilities are enumerated in Sm_DirectoryCapability_t, which is defined in the header file SmApi.h. See Figure 12 on page 103.

Returns

- Sm_PolicyApi_Success. The retrieve was successful.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve user directory capability.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.

Sm_PolicyApi_GetUserDirSearchOrder()

Retrieves the OIDs of the user directory objects that are associated with the domain identified by *pszDomainOid*.

The retrieved list of OIDs is stored in the *pszArray* string array in the order in that they are searched by SiteMinder. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeStringArray().

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetUserDirSearchOrder (
   void*         pSessionHandle,
   const char*   pszDomainOid,
   char**        pszArray[]
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain.
pszArray	O	The returned array of user directory OIDs of the requested objects.

Returns

- Sm_PolicyApi_Success. The function successfully returned the user directory search order.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get user directory search order in a domain.
- Sm_PolicyApi_InvalidOid. The domain OID was not found.

Sm_PolicyApi_GetUserGroups()

Gets the list of groups that a user is member of.

Type

User and user state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetUserGroups (
   void *          pSessionHandle,
   const char *    pszUserDirOid,
   const char *    pszUserDN,
   const bool      bRecursive,
   char**          pszGroups[]
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	The user's object identifier.
pszUserDN	I	The user's distinguished name.
bRecursive	I	Value indicating whether to search just one level or all levels. If you specify True, all levels are searched.
pszGroups	O	Array that will contain the groups that the user belongs to. Free the memory allocated for the array by calling Sm_PolicyApi_FreeStringArray().

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. The get operation was not successful.
- Sm_PolicyApi_Invalid. There was no valid directory for the specified user OID and DN.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_InvalidOid. The directory OID was not found (for a directory-based group).
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to get the groups.

Sm_PolicyApi_GetUserPasswordState()

Returns Sm_Api_UserPasswordState_t. The memory for the Sm_Api_UserPasswordState_t object should be allocated by the calling function.

Type

Password state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetUserPasswordState (
   void       *pSessionHandle,
   const char *pszUserDirOid,
   const char *pszUserDN,
   Sm_PolicyApi_UserPasswordState_t *pPasswordState
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	Unique object identifier that corresponds to a particular User Directory.
pszUserDN	I	Specifies the DN of the user within the user directory.
pPasswordState	O	The user password state object.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_BadArgument. The input password state pointer is NULL.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to User Directory object.
- Sm_PolicyApi_UserDirNotFound. Failed to retrieve User Directory object from policy store.
- Sm_PolicyApi_ErrorUserDir. Invalid User Directory object or wrong user DN.
- Sm_PolicyApi_NoPrivilege. The caller does not have the proper privileges.

Sm_PolicyApi_GetUsersFromWSFEDResourcePartner()

Gets the user directory entries associated with a WS-Federation Resource Partner.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetUsersFromWSFEDResourcePartner (
        void* pSessionHandle,
        const char * pszProviderOid,
        const char * pszUserPolicyOid,
       Sm_PolicyApi_User_t ** ppStructUsers
);
```

Parameters

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszProviderOid

[in] A null-terminated string containing the object identifier of an existing WS-Federation Resource Partner.

pszUserPolicyOid

[in] A null-terminated string containing the object identifier of an existing policy user.

ppStructUsers

[out] The address of a pointer to a linked list of user structures.

Return Values

- Sm_PolicyApi_Success. The user directory was added successfully.
- Sm_PolicyApi_Failure. The user directory was not added successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve users.
- Sm_PolicyApi_InvalidOid. The affiliate OID was not found.

Sm_PolicyApi_GetVariable()

Gets a specified variable.

The results of this function are returned in a structure referenced by *ppstructVariable*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Variable function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetVariable (
   void*                      pSessionHandle,
   const char*                pszVariableOid,
   Sm_PolicyApi_Variable_t**  ppstructVariable
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszVariableOid	I	Unique object identifier that corresponds to a variable.
ppstructVariable	O	The address of a pointer to a variable structure.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Variable_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to obtain a variable.
- Sm_PolicyApi_InvalidOid. The variable object identifier was not found.
- Sm_PolicyApi_FeatureNotSupported. The client who called this function initialized the API with a version less than SM_POLICY_API_VERSION_6_0.

Sm_PolicyApi_GetVariableByName()

Gets a specified variable.

The results of this function are returned in a structure referenced by *ppstructVariable*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Variable function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetVariableByName (
   void*                      pSessionHandle,
   const char*                pszVariableName,
   Sm_PolicyApi_Variable_t**  ppstructVariable
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing Domain.
pszVariableName	I	Unique name that corresponds to a variable.
ppstructVariable	O	The address of a pointer to a variable structure.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Variable_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to obtain a variable.
- Sm_PolicyApi_InvalidOid. The Domain OID was not found.
- Sm_PolicyApi_NotFound. The variable name was not found.
- Sm_PolicyApi_FeatureNotSupported. The client who called this function initialized the API with a version less than SM_POLICY_API_VERSION_6_0.

Sm_PolicyApi_GetVariableType()

Gets a specified variable type.

The results of this function are returned in a structure referenced by *ppstructVariable*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Variable function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetVariableType (
   void*                            pSessionHandle,
   const char*                      pszVariableTypeOid,
   Sm_PolicyApi_VariableType_t**    ppstructVariableType
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszVariableOid	I	Unique object identifier that corresponds to a variable type.
ppstructVariable	O	The address of a pointer to a variable structure.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Variable_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to obtain the variable type.
- Sm_PolicyApi_InvalidOid. The variable type object identifier was not found.
- Sm_PolicyApi_FeatureNotSupported. The client who called this function initialized the API with a version less than SM_POLICY_API_VERSION_6_0.

Sm_PolicyApi_GetVariableTypeByName()

Gets a specified variable type.

The results of this function are returned in a structure referenced by *ppstructVariable*. Free the memory allocated for this structure by calling Sm_PolicyApi_FreeMemoryEx().

Type

Variable function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetVariableTypeByName (
   void*                           pSessionHandle,
   const char*                     pszVariableTypeName,
   Sm_PolicyApi_VariableType_t**   ppstructVariableType
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszVariableName	I	Unique name that corresponds to a variable type.
ppstructVariable	O	The address of a pointer to a variable structure.

Returns

- Sm_PolicyApi_Success. The get was successful.
- Sm_PolicyApi_Failure. Memory could not be allocated to Sm_PolicyApi_Variable_t.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to obtain the variable type.
- Sm_PolicyApi_NotFound The variable type object identifier was not found.
- Sm_PolicyApi_FeatureNotSupported. The client who called this function initialized the API with a version less than SM_POLICY_API_VERSION_6_0.

Sm_PolicyApi_GetWSFEDResourcePartner()

Gets an existing Resource Partner object.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetWSFEDResourcePartner (
        void* pSessionHandle,
        const char * pszProviderOid;
        Sm_PolicyApi_WSFEDResourcePartner_t** pstructServiceProvider
);
```

Parameters

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszProviderOid

[in] A null-terminated string containing the object identifier of an existing WS-Federation Resource Partner.

pstructServiceProvider

[out] The address of the pointer to WS-Federation Resource Partner structure.

Return Values

- Sm_PolicyApi_Success. The Resource Partner was retrieved successfully.
- Sm_PolicyApi_Failure. The Resource Partner was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve an affiliate.
- Sm_PolicyApi_InvalidOID. The Resource Partner OID was not found.
- Sm_PolicyApi_DomainNotAffiliate. The domain is not an affiliate domain

Sm_PolicyApi_GetWSFEDScheme()

Gets an existing WSFED authentication scheme object.

Syntax

```
int SM_EXTERN Sm_PolicyApi_GetWSFEDScheme (
        void* pSessionHandle,
        const char * pszProviderOid,
        Sm_PolicyApi_Scheme_t** ppstructScheme,
        Sm_PolicyApi_WSFEDProviderProp_t** ppProps
);
```

Parameters

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszProviderOid

[in] A null-terminated string containing the object identifier of an existing WSFED auth scheme.

ppstructScheme

[out] The address of the pointer to SiteMinder auth scheme structure.

ppProps

[out] The address of the pointer to linked list WSFED provider properties.

Return Values

- Sm_PolicyApi_Success. The authentication scheme was retrieved successfully.
- Sm_PolicyApi_Failure. The authentication scheme was not retrieved successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to retrieve an affiliate.
- Sm_PolicyApi_InvalidOID. The provider OID was not found.

Sm_PolicyApi_Init()

Initializes the connection to the SiteMinder policy store and establishes the *init handle*.

You can specify an initialization flag that will affect the API's behavior.

This function must be called once per API client session. It must be the first function called in the session.

Type

Required function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_Init (
   void**                        ppInitHandle,
   const Sm_PolicyApi_InitFlags_t   nInitFlag
);
```

Parameter	**I/O**	**Description**
ppInitHandle	O	A pointer to an internal Policy Management API data structure that contains client session information. This init handle is returned on successful initialization and is used as an input parameter to every call to Sm_PolicyApi_Login() and Sm_PolicyApi_Release().
nInitFlag	I	Value affecting the behavior of the API.

Returns

- Sm_PolicyApi_Success. The initialization of the Policy Store connection was successful.
- Sm_PolicyApi_Failure. The initialization of the Policy Store connection was unsuccessful or memory could not be allocated.

Sm_PolicyApi_InitEx()

Initializes a connection to the SiteMinder policy store and establishes the init handle based on a supplied version.

Type

Required function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_InitEx (
   void**                           ppInitHandle,
   const Sm_PolicyApi_InitFlags_t   nInitFlag,
   const unsigned                   version
)
```

Parameter	I/O	Description
ppInitHandle	O	A pointer to an internal Policy Management API data structure that contains client session information.
nInitFlag	I	Value affecting the behavior of the API.
version	I	Version of the Policy Management API to initialize.

Returns

- Sm_PolicyApi_Success. The initialization of the Policy Store connection was successful.
- Sm_PolicyApi_Failure. The initialization of the Policy Store connection was unsuccessful or memory could not be allocated.

Sm_PolicyApi_IsGroup()

Determines whether an item is a group.

Type

Group function, global scope (agents) or domain scope (responses, rules).

Syntax

```
int SM_EXTERN Sm_PolicyApi_IsGroup (
   void*                   pSessionHandle,
   Sm_PolicyApi_Groups_t   dwGroup,
   const char*             pszOid,
   const char*             pszDomainOid,
   bool*                   bIsGroup
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
dwGroup	I	Indicates the type of the object.
pszOid	I	A null-terminated string containing the object identifier of the item to check.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain.
bIsGroup	O	A pointer to a boolean value. true if the *pszOid* object is a group.

Returns

- Sm_PolicyApi_Success. The OID is a group.
- Sm_PolicyApi_Failure. The OID is not a group.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to find out if a rule, response, or agent OID is a group.
- Sm_PolicyApi_InvalidOid:
    - The domain OID was not found (for a domain-based group).
    - The group OID was not found.
    - The rule, response, or agent OID or group OID was not found.
- Sm_PolicyApi_BadGroup. Parameter *dwGroup* is not the rule, response, or agent type.

Sm_PolicyApi_Login()

After initialization, a successful call to Sm_PolicyApi_Login() is a prerequisite to making any further function calls. This function checks the administrator's login credentials (username and password). If the API detects an uninitialized or improperly initialized init handle, an error is generated. If the administrator is authenticated, the function initializes internal data structures and resources. Once the administrator is logged in, the Policy Server initializes a session handle, which is used as an input parameter to all the Policy Management API functions.

Internally, the session handle contains data structures and context information required for the operation of the Policy Management API, including the client session data (from the init handle) and the administrator session data. The data structures and context information are transparent to the caller.

You can call Sm_PolicyApi_Login() to initialize a session handle *without* checking the administrator's credentials or without specifying any administrator. To log in under either of these circumstances, set the parameter *nCheckCreds* to false. See the description of the *nCheckCreds* parameter for more information.

This function can be called more than once during the client session and depends on the successful initialization of the Policy Store connection.

Type

Required function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_Login (
   void*           pInitHandle,
   void**          ppSessionHandle,
   int             nCheckCreds,
   const char*     pszUserName,
   const char*     pszPassword,
   const char*     pszClientIP,
   char**          pszUserMsg,
   char**          pszErrMsg
);
```

Parameter	I/O	Description
pInitHandle	I	A pointer to an internal Policy Management API data structure. This is the init handle returned by Sm_PolicyApi_Init().

Parameter	I/O	Description
ppSessionHandle	O	A pointer to an internal Policy Management API data structure (a different instance of the structure from *pInitHandle*). The structure contains administrator session data. The session handle is transparent to the caller.
nCheckCreds	I	Flag indicating whether to check the credentials of the administrator, as follows: ■ If *nCheckCreds* is false and an empty string ("") is passed in *pszUserName*, no credential checking is done, and the caller is granted all administrator rights. ■ If *nCheckCreds* is false and a user name is passed in *pszUserName*, no credential checking is done. However, the passed name must be that of a valid SiteMinder administrator. The caller is granted rights defined for the administrator. ■ If *nCheckCreds* is true, both the user name and the password are verified and should match a valid SiteMinder administrator. The caller is granted rights defined for the administrator.
pszUserName	I	User Name of the Policy Management API administrator. If you pass in an empty string ("") and set *nCheckCreds* to false, no administrator name and password are required. The caller is granted all administrator rights.
pszPassword	I	Password of the Policy Management API administrator.
pszClientIP	I	IP address of the machine the administrator is logging from.
pszUserMsg	O	User message returned by the Policy Management API. You release the memory allocated for this variable by calling Sm_PolicyApi_FreeString().
pszErrMsg	O	Error message returned by the Policy Management API. You release the memory allocated for this variable by calling Sm_PolicyApi_FreeString().

Returns

- Sm_PolicyApi_Success. The login was successful.
- Sm_PolicyApi_Failure. Memory cannot be allocated.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have privileges to some or all the domains.
- Sm_PolicyApi_ErrorLogin. The authentication failed.

Example

See the Sm_PolicyApi_Login() call in the example application smpolicyapiexample.cpp.

Sm_PolicyApi_Logout()

Logs out an administrator session.

Type

Required function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_Logout (
   void* pSessionHandle
);
```

Parameter	I/O	Description
pSessionHandle	I	The session handle that was returned by Sm_PolicyApi_Login() after successful login of the administrator.

Returns

- Sm_PolicyApi_Success. The logout was successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.

Sm_PolicyApi_LookupDirectoryEntry()

Looks up the user specification in a user directory.

pszSearchPattern holds the search pattern for the lookup. User directory searches vary for each type of user directory namespaces.

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_LookupDirectoryEntry (
   void*                   pSessionHandle,
   const char*             pszUserDirOid,
   const char*             pszSearchPattern,
   Sm_PolicyApi_User_t**   ppStructUserSpec
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of the user directory where lookup is to be performed.
pszSearchPattern	I	A null-terminated string containing the search pattern for the lookup. Information about the search expression grammar for different namespace types appears in the section below.
ppStructUserSpec	O	The address of a pointer to a user structure.

Search Expression Grammar for an LDAP Namespace

The search in an LDAP user directory can be based on an attribute-value pair or on an LDAP search expression.

- For an attribute-value pair, the format of *pszSearchPattern* is:

  ```
  <attribute-name>=<value>
  ```

  For example if <*attribute-name*> is disabled and <*value*> is 0, the LDAP search filter is:

  Base:'o=airius.com',

  Filter:

  ```
  '(&
        (|(objectclass=organizationalPerson)
        (objectclass=inetOrgPerson)
        (objectclass=organization)
        (objectclass=organizationalUnit)
        (objectclass=groupOfNames)
        (objectclass=groupOfUniqueNames)
        (objectclass=group)
        )
        (disabled=0)
  )'
  ```

- If the search uses an LDAP search expression, *pszSearchPattern* will hold the LDAP search expression. For example, if the search expression is 'uid=user1111', the LDAP search filter is:

  Base:'o=airius.com',

  Filter:

  ```
  '(&
        (|(objectclass=organizationalPerson)
        (objectclass=inetOrgPerson)
        (objectclass=organization)
        (objectclass=organizationalUnit)
        (objectclass=groupOfNames)
        (objectclass=groupOfUniqueNames)
        (objectclass=group)
        )
        (uid=user1111)
  )'
  ```

Search Expression Grammar for ODBC, WinNT and Custom Namespaces

You can search in an ODBC user directory for users, groups, or both. The search is based on attribute-value pairs.

The format of *pszSearchPattern* is:

```
[ <class>= ] <value>
```

In the format example:

- <*class*> is an empty-string or user or group. An empty-string implies user and group.
- <*value*> is a wildcard string.

Returns

- Sm_PolicyApi_Success. The lookup was successful.
- Sm_PolicyApi_Failure. The user specification lookup failed or memory could not be allocated to Sm_PolicyApi_User_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.
- Sm_PolicyApi_UserDirNotValid. There is no connection to the user directory or the user directory provider has not been loaded.

Sm_PolicyApi_ManagementCommand()

Performs user, key, and resource management activities.

The Policy Management API supports the types of management commands that are enumerated in Sm_PolicyApi_ManagementCommands_t.

Type

Utility function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_ManagementCommand (
  void*                              pSessionHandle,
  Sm_PolicyApi_ManagementCommand_t* pstructManagementCommand
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructManagementCommand	I	The address of a pointer to a management command structure.

Returns

- Sm_PolicyApi_Success. The management command was issued successfully.
- Sm_PolicyApi_Failure. The management command was not issued successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to call Sm_PolicyApi_ManagementCommand().
- Sm_PolicyApi_Invalid. An invalid management command was specified.

Sm_PolicyApi_Release()

Disconnects from the policy store and releases memory and resources held by the API.

This function must be the last function called by the API client session. This function must be called once per client session.

Note: Failure to call this function will result in a memory leak.

Type

Required function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_Release(void* pInitHandle);
```

Parameter	I/O	Description
pInitHandle	I	The init handle that was returned by Sm_PolicyApi_Init() after successful initialization of the client session.

Returns

This function always returns Sm_PolicyApi_Success.

Sm_PolicyApi_RemoveAdminFromAffiliateDomain()

Removes an administrator from an affiliate domain.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveAdminFromAffiliateDomain (
   void*         pSessionHandle,
   const char*   pszAdminOid,
   const char*   pszAffiliateDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAdminOid	I	A null-terminated string containing the object identifier of an existing administrator.
pszAffiliate DomainOid	I	A null-terminated string containing the object identifier of an existing affiliate domain.

Returns

- Sm_PolicyApi_Success. The administrator was removed from the affiliate domain.
- Sm_PolicyApi_Failure. The administrator was not removed from the affiliate domain.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove an administrator from an affiliate domain.
- Sm_PolicyApi_InvalidOID. The affiliate domain OID was not found during an update.
- Sm_PolicyApi_NotFound. The administrator object identifier could not be found in the affiliate domain collection.

Sm_PolicyApi_RemoveAdminFromDomain()

Disassociates the administrator object identified by *pszAdminOid* from the domain identified by *szDomainOid*.

Type

Administrator function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveAdminFromDomain (
   void*          pSessionHandle,
   const char*    pszAdminOid,
   const char*    pszDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAdminOid	I	A null-terminated string containing the object identifier of an existing administrator.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain.

Returns

- Sm_PolicyApi_Success. The administrator was removed successfully from a domain.
- Sm_PolicyApi_Failure. The administrator was not removed successfully from a domain.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove an administrator from a domain.
- Sm_PolicyApi_InvalidOid. The administrator object identifier or the domain object identifier was not found.
- Sm_PolicyApi_NotFound, if
  - There are no administrator object identifiers in the domain collection.
  - The administrator object identifier could not be found in the domain collection.

Sm_PolicyApi_RemoveAgentConfigAssociation()

Removes a configuration parameter name/value pair from the specified agent configuration object.

Type

Agent configuration function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveAgentConfigAssociation (
   void*                  pSessionHandle,
   const char*            pszAssociationOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAssociationOid	I	Unique identifier of the name/value pair to remove.

Returns

- Sm_PolicyApi_Success. The operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidOid. The unique ID does not correspond to an agent configuration object.
- Sm_PolicyApi_NoPrivilege. The caller does not have the privilege to remove agent configuration object parameters.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.

Sm_PolicyApi_RemoveAssertionConsumerServiceFromSAMLSP()

Removes an existing indexed endpoint reference to an Assertion Consumer Service from the policy store. The index, binding type, and Assertion Consumer Service URL must match an existing Assertion Consumer Service.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveAssertionConsumerServiceToSAMLSP (
   void*                              pSessionHandle,
   const Sm_PolicyApi_SAMLSPAssertionConsumerService_t*
                pstructSAMLSPAssertionConsumerService,
   const char*                        pszSAMLSPOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructSAMLAssertionConsumerService	I	A pointer to an Assertion Consumer Service structure.
pszSAMLSPOid	I	A pointer to a string containing the OID of the Service Provider.

Returns

- Sm_PolicyApi_Success. The Assertion Consumer Service was removed successfully.
- Sm_PolicyApi_Failure - The Assertion Consumer Service was not removed successfully.
- Sm_PolicyApi_ACSIndexedEndpointNotFound - There is no Assertion Consumer Service that matches the Assertion Consumer Service to be removed..
- Sm_PolicyApi_InvalidHandle - There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession - There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege - The administrator does not have the privilege to remove Attributes from a SAML Service Provider.

Sm_PolicyApi_RemoveAttributeFromAffiliate()

Removes an attribute from an affiliate.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveAttributeFromAffiliate (
   void*                             pSessionHandle,
   const Sm_PolicyApi_AffiliateAttr_t* pstructAffiliateAttr,
   const char*                       pszAffiliateOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructAffiliate Attr	I	A pointer to an affiliate attribute structure.
pszAffiliateOid	I	A null-terminated string containing the object identifier of an existing affiliate.

Returns

- Sm_PolicyApi_Success. The affiliate attribute was removed successfully.
- Sm_PolicyApi_Failure. The affiliate attribute was not removed successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove an affiliate attribute.
- Sm_PolicyApi_InvalidOID. The affiliate OID was not found.

Sm_PolicyApi_RemoveAttributeFromSAMLScheme()

Removes a configured attribute from a SAML authentication scheme.

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveAttributeFromSAMLScheme(
   void*                                pHandle,
   const Sm_PolicyApi_Scheme_t*         pstructScheme,
   const Sm_PolicyApi_SAMLRequesterAttr_t*     pAttr
);
```

Parameter	I/O	Description
pHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructScheme	I	A pointer to a completely filled-in scheme structure.
pAttr	I	A pointer to the Sm_PolicyApi_SAMLRequesterAttr_t structure containng the attribute to be removed.

Returns

- Sm_PolicyApi_Success. The attribute was removed successfully.
- Sm_PolicyApi_Failure. The attribute was not removed successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove an attribute to a SAML Requester.
- Sm_PolicyApi_SAMLIDP_IncorrectParameters. Supplied SAML provided properties are incomplete or incorrect.
- Sm_PolicyApi_NoSuchAttribute. An attribute with a matching Name and NameFormat does not exist.

Sm_PolicyApi_RemoveAttributeFromSAMLSP()

Removes the specified SAML attribute from the Service Provider.

Type

SAML 2.0 Configuration function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveAttributeFromSAMLSP
(
   void*                              pHandle,
   const Sm_PolicyApi_SAMLSPAttr_t*   pstructSAMLSPAttr,
   const char*                        pszSAMLSPOid
);
```

Parameter	I/O	Description
pHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pstructSAMLSPAttr	I	A pointer to a SAML attribute structure

Parameter	I/O	Description
pszSAMLSPOid	I	The Service Provider's object identifier in the policy store.

Returns

- Sm_PolicyApi_Success. The attribute was removed successfully.
- Sm_PolicyApi_Failure. The attribute was not removed successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove an SAML attribute from a Service Provider.
- Sm_PolicyApi_NoSuchAttribute. No attribute exists with the Name and NameFormat provided.

Sm_PolicyApi_RemoveFromGroup()

Removes an item from a group.

The item and the group must exist and must be of the same type, and the item must be contained in the group.

The *pszDomainOid* parameter is required by rule and response groups. An agent group does not require a domain OID because it is not a domain-based object.

Type

Group function, global scope (agents) or domain scope (responses, rules).

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveFromGroup (
   void*                   pSessionHandle,
   Sm_PolicyApi_Groups_t   dwGroup,
   const char*             pszItemOid,
   const char*             pszGroupOid,
   const char*             pszDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
dwGroup	I	Indicates the type of the group.
pszItemOid	I	A null-terminated string containing the object identifier of an existing item; must be the same type as the group.
pszGroupOid	I	A null-terminated string containing the object identifier of the group.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain. Required by rule and response groups.

Returns

- Sm_PolicyApi_Success. The remove was successful.
- Sm_PolicyApi_Failure. The remove was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove a rule, response, or agent OID from its respective group.
- Sm_PolicyApi_InvalidOid:
  - The domain OID was not found (for a domain-based group).
  - The group OID was not found.
  - The rule, response, or agent OID or group OID was not found.
- Sm_PolicyApi_BadGroup. Parameter *dwGroup* is not the rule, response, or agent type.

Sm_PolicyApi_RemovePolicyLinkFromPolicy()

By removing the policy link identified by *pszPolicyLinkOid* from the policy identified by *pszPolicyOid*, this function effectively removes the rule from the policy.

A policy link object binds a policy to a rule and, optionally, a response.

Note: In releases prior to SiteMinder v4.5, the functionality provided by Sm_PolicyApi_RemovePolicyLinkFromPolicy() was provided by Sm_PolicyApi_RemoveRuleFromPolicy().

Type

Policy function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemovePolicyLinkFromPolicy (
   void*         pSessionHandle,
   const char*   pszPolicyLinkOid,
   const char*   pszPolicyOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPolicyLinkOid	I	A null-terminated string containing the object identifier of an existing policy link under *pszPolicyOid*.
pszPolicyOid	I	A null-terminated string containing the object identifier of an existing policy.

Returns

- Sm_PolicyApi_Success. The policy link was removed successfully.
- Sm_PolicyApi_Failure. The policy link was not removed successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove a policy link.
- Sm_PolicyApi_InvalidOid:
  - The policy OID was not found.
  - The policy link OID was not found.

Sm_PolicyApi_RemoveRegularExpressionFromPasswordPolicy()

Removes a Regular Expression from the referenced password policy. Implemented only if the session's version is set to SM_POLICY_API_VERSION_6_0.

Type

Regular Expression function.

Syntax

```
int SM_EXTERN    Sm_PolicyApi_RemoveRegularExpressionFromPasswordPolicy(
   void*        pSessionHandle,
   const char*  pszRegularExpressionOid,
   const char*  pszPasswordPolicyOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszRegular ExpressionOid	I	The OID of the regular expression to be removed.
pszPassword PolicyOid	I	The OID of the password policy to remove the expression from.

Returns

- Sm_PolicyApi_Success. The regular expression was removed successfully.
- Sm_PolicyApi_Failure. The regular expression was not removed successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove regular expressions.
- Sm_PolicyApi_InvalidOid: The password policy OID was not found.

Sm_PolicyApi_RemoveResponseAttr()

Disassociates the response attribute defined by the *pstructResponseAttr* structure from the response identified by *pszResponseAttrOid*. The attribute name and attribute value must match in order for the remove to occur.

Type

Response function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveResponseAttr (
   void*          pSessionHandle,
   const char*    pszResponseAttrOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszResponseAttrOid	I	A null-terminated string containing the object identifier of an existing response attribute.

Returns

- Sm_PolicyApi_Success. The remove operation was successful.
- Sm_PolicyApi_Failure. The remove operation was not successful.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove a response attribute.
- Sm_PolicyApi_InvalidOid. The response attribute OID was not found.

Sm_PolicyApi_RemoveUserDirFromAffiliateDomain()

Removes a user directory from an existing affiliate domain.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveUserDirFromAffiliateDomain(
   void*        pSessionHandle,
   const char*  pszUserDirOid,
   const char*  pszAffiliateDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory.
pszAffiliate DomainOid	I	A null-terminated string containing the object identifier of an existing affiliate domain.

Returns

- Sm_PolicyApi_Success. The user directory was removed from the affiliate domain successfully.
- Sm_PolicyApi_NotFound. The domain does not have the user directory.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove a user directory from an affiliate domain.
- Sm_PolicyApi_InvalidOID. The user directory OID or affiliate domain OID was not found during an update.

Sm_PolicyApi_RemoveUserDirFromDomain()

Disassociates the user directory identified by *pszUserDirOid* from the domain identified by *pszDomainOid*.

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveUserDirFromDomain (
   void*          pSessionHandle,
   const char*    pszUserDirOid,
   const char*    pszDomainOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain.

Returns

- Sm_PolicyApi_Success. The remove operation was successful.
- Sm_PolicyApi_NotFound. The domain does not have any user directories.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove a user directory from a domain.
- Sm_PolicyApi_InvalidOid. A user directory or domain OID was not found.

Sm_PolicyApi_RemoveUsersFromAffiliate()

Removes a user directory entry from an affiliate.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveUsersFromAffiliate (
   void*        pSessionHandle,
   const char*  pszAffiliateOid,
   const char*  pszUserPolicyOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliateOid	I	A null-terminated string containing the object identifier of an existing affiliate.
pszUserPolicyOid	I	A null-terminated string containing the object identifier of an existing policy user.

Returns

- Sm_PolicyApi_Success. The user was removed successfully.
- Sm_PolicyApi_Failure. The user was not removed successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove a user.
- Sm_PolicyApi_InvalidOID. The affiliate OID or user policy OID was not found.

Sm_PolicyApi_RemoveUsersFromPolicy()

Disassociates the user identified by *pszUserPolicyOid* from the policy identified by *pszPolicyOid*. Only one user specification (which may be an aggregate) can be removed at a time.

Type

User and user state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveUsersFromPolicy (
   void*          pSessionHandle,
   const char*    pszPolicyOid,
   const char*    pszUserPolicyOid
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPolicyOid	I	A null-terminated string containing the object identifier of an existing policy from which a user is to be removed.
pszUserPolicyOid	I	A null-terminated string containing the object identifier of an existing policy user.

Remarks

This function is successful only when the pointer to Sm_PolicyApi_User_t is obtained with the Sm_PolicyApi_GetPolicyUsers() function. If the user is retrieved with Sm_PolicyApi_LookupDirectoryEntry(), Sm_PolicyApi_GetDirectoryContents(), or Sm_PolicyApi_ValidateDirectoryEntry(), *pszUserPolicyOid* will be invalid and the call will fail.

Returns

- Sm_PolicyApi_Success. The remove operation was successful.
- Sm_PolicyApi_Failure. The remove operation was not successful.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove a user from the policy.
- Sm_PolicyApi_InvalidOid:
  - The policy OID was not found.
  - The user policy OID was not found.

Sm_PolicyApi_RemoveUsersFromSAMLServiceProvider()

Removes the specified users from the Service Provider.

Type

SAML 2.0 Configuration function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_AddUsersToSAMLServiceProvider
(
   void*                              pSessionHandle,
   const char*                        pszProviderOid,
   Sm_PolicyApi_User_t*               pStructUsers,
   int                                iPolicyFlags
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszProviderOid	I	A null-terminated string containing the Service Provider's object identifier.
pStructUsers	I	The users to remove from the Service Provider.
iPolicyFlags	I	A bit field that indicates whether: ■ The policy created for the SAML Service Provider includes a user ■ The policy should be applied recursively

Returns

- Sm_PolicyApi_Success. The users were removed successfully.
- Sm_PolicyApi_Failure. The users were not removed successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove a user.
- Sm_PolicyApi_InvalidOid. The Service Provider OID was not found.

Sm_PolicyApi_RemoveUsersFromWSFEDResourcePartner()

Dissociates a user directory entry from WS-Federation Resource Partner.

Syntax

```
int SM_EXTERN Sm_PolicyApi_RemoveUsersFromWSFEDResourcePartner (
        void* pSessionHandle,
        const char * pszProviderOid,
        const char * pszUserPolicyOid
);
```

Parameters

pSessionHandle

[in] A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.

pszProviderOid

[in] A null-terminated string containing the object identifier of an existing WS-Federation Resource Partner.

pszUserPolicyOid

[in] A null-terminated string containing the object identifier of an existing policy user.

Return Values

- Sm_PolicyApi_Success. The user directory was removed a successfully.
- Sm_PolicyApi_Failure. The user directory was not removed successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to remove a user.
- Sm_PolicyApi_InvalidOid. The Resource Partner OID was not found.

Sm_PolicyApi_RenameObject()

Renames a domain or global object.

This function requires the Object Identifier (OID) of the object to be renamed. You can retrieve the object identifier by performing the Get operation on the object.

Type

General object function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_RenameObject (
   void*          pSessionHandle,
   const char*    pszOid,
   const char*    pszNewName
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszOid	I	Object Identifier of the object to be renamed.
pszNewName	I	New name for the object.

Returns

- Sm_PolicyApi_Success. The object was renamed successfully.
- Sm_PolicyApi_Failure. The object was not renamed successfully.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to rename an object.
- Sm_PolicyApi_InvalidOid. The OID was not found.

Sm_PolicyApi_SetAffiliateDomainUserDirSearchOrder()

Sets the user directory search order for an affiliate domain.

Type

Federation function

Syntax

```
int SM_EXTERN Sm_PolicyApi_SetAffiliateDomainUserDirSearchOrder (
   void*         pSessionHandle,
   const char*   pszAffiliateDomainOid,
   char**        pszArray[]
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszAffiliate DomainOid	I	A null-terminated string containing the object identifier of an existing affiliate domain.
pszArray	I	An array of user directory OIDs, in the desired order.

Returns

- Sm_PolicyApi_Success. The function successfully set the user directory search order.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to set user directory search order in an affiliate domain.
- Sm_PolicyApi_InvalidOID. The affiliate OID was not found.
- Sm_PolicyApi_DomainNotAffiliate. The domain is not an affiliate domain.
- Sm_PolicyApi_WrongNumberOfElements. The number of user directories in the affiliate domain collection is different from the number of elements in the array.

Sm_PolicyApi_SetDisabledUserState()

Sets the disabled state of a user. You can also enable a user with this function.

To make this function work, the attribute for tracking disabled users must be set in the user directory (the *pszDisabledAttr* field of structure Sm_PolicyApi_UserDir_t). You can also set the attribute using the Policy Server User Interface.

Type

User and user state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_SetDisabledUserState (
   void*                             pSessionHandle,
   const char*                       pszUserDirOid,
   const char*                       pszUserDN,
   const Sm_Api_DisabledReason_t     nDisabledReason,
   char**                            pszErrMsg
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of an existing user directory.
pszUserDN	I	The distinguished name of the user whose disabled state is to be changed.
nDisabledReason	I	Reason for disabling or enabling a user. The reasons are enumerated in Sm_Api_DisabledReason_t, which is defined in SmApi.h. It is the responsibility of the caller to set the correct state. Multiple reasons can exist concurrently. When a user is enabled, all the flags in the disabled mask should be cleared.

Parameter	I/O	Description
pszErrMsg	O	The error message is held in the string if the operation was not successful. You release the memory allocated for this variable by calling Sm_PolicyApi_FreeString().

Returns

- Sm_PolicyApi_Success. The disable user state was set successfully.
- Sm_PolicyApi_Failure:
  - User state was not disabled.
  - Memory could not be allocated to *pszErrMsg*.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to set disabled user state.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.

Examples

- To disable a user for password expiration:

```
nDisabledReason = Sm_Api_Disabled_DisabledMask &
      Sm_Api_Disabled_PWExpired;
```

- To disable a user for administrative reasons:

```
enum Sm_Api_DisabledReason_t nDisabledReason;


iRes =  Sm_PolicyApi_GetDisabledUserState (pSessionHandle,
                                           pszUserDirOid,
                                           pszUserDN,
                                           &nDisabledReason,
                                           &pszGetErrMsg);

if (iRes != Sm_PolicyApi_Success)
{
    cout << "Error: " << pszGetErrMsg << endl;
}

// Set admin disabled reason bit.
nDisabledReason=(Sm_Api_DisabledReason_t) (nDisabledReason |
      Sm_Api_Disabled_AdminDisabled);

// Set Disable user state
iRes = Sm_PolicyApi_SetDisabledUserState(pSessionHandle,
                                         pszUserDirOid,
                                         pszUserDN,
                                         nDisabledReason,
                                         &pszSetErrMsg);
```

- To enable a user and clear all disable reason bits:

```
enum Sm_Api_DisabledReason_t nDisabledReason;
iRes = Sm_PolicyApi_GetDisabledUserState(pSessionHandle,
                                         pszUserDirOid,
                                         pszUserDN,
                                         &nDisabledReason,
                                         &pszGetErrMsg);

if (iRes != Sm_PolicyApi_Success)
{
   if (pszGetErrMsg)
   {
      cout << "Error: " << pszGetErrMsg << endl;
   }
}
// Clear all the disable reason bits.
nDisabledReason=(Sm_Api_DisabledReason_t)
      (nDisabledReason & (~Sm_Api_Disabled_DisabledMask));

// Set Disable user state to enable
iRes = Sm_PolicyApi_SetDisabledUserState(pSessionHandle,
                                         pszUserDirOid,
                                         pszUserDN,
                                         nDisabledReason,
                                         &pszSetErrMsg);
```

Sm_PolicyApi_SetPassword()

Changes the password of a user account. It can also be used to validate a new password or an old password without changing the password.

To validate a new password, you must set SiteMinder Password services for the directory, and you must identify a password attribute in the SiteMinder user directory.

Type

User and user state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_SetPassword (
   void*          pSessionHandle,
   const char*    pszUserDirOid,
   const char*    pszUserDN,
   const char*    pszNewPassword,
   const char*    pszOldPassword,
   bool           bChangePassword,
   bool           bValidateNewPassword,
   bool           bValidateOldPassword,
   char**         pszErrMsg
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of the user directory where the user may be found.
pszUserDN	I	The distinguished name of the user whose password is to be changed and/or whose new or old password is to be validated.
pszNewPassword	I	New user password to validate or change.
pszOldPassword	I	Old user password to validate or change.

Parameter	I/O	Description
bChangePassword	I	If true, the password is changed to the new password and is recorded in the user's password history. If an error occurs, the function returns an error code.
bValidateNewPassword	I	If true, the new password is checked to make sure it satisfies all password policy requirements. If the new password is in violation of any password policies, *pszErrMsg* is set and the function returns an error code.
bValidateOldPassword	I	If true, the old password is used to authenticate the user. If authentication fails, the function returns an error code.
pszErrMsg	O	String containing an error message if the user password change or validation was not successful. You release the memory allocated for this variable by calling Sm_PolicyApi_FreeString().

Returns

- Sm_PolicyApi_Success. The change or validation was successful.
- Sm_PolicyApi_Failure. The change or validation was not successful or memory could not be allocated to the error message string.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_InvalidPasswordSyntax. Returned when both of these conditions exist:
  - *bValidateNewPassword* is true.
  - The new password did not satisfy the password policy requirements set for the specified directory.

  For information about the error, call Sm_PolicyApi_GetPasswordMsg() and pass to it the encoded error message (*pszErrMsg*) returned from Sm_PolicyApi_SetPassword().
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to set or validate user passwords.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.

Sm_PolicyApi_SetResponseInPolicyLink()

Alters the policy link described by the *ppstructPolicyLink* structure.

This function sets a response or a response group to a rule or rule group. It can also be used to remove a response or response group from a policy link. To remove a response or response group from a policy link, set the *pszResponseOid* in Sm_PolicyApi_PolicyLink_t to an empty string.

Type

Response function, domain scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_SetResponseInPolicyLink (
   void*                        pSessionHandle,
   const char*                  pszPolicyOid,
   Sm_PolicyApi_PolicyLink_t*   ppstructPolicyLink
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszPolicyOid	I	A null-terminated string containing the object identifier of an existing policy.
ppstructPolicyLink	I	The address of a policy link structure.

Returns

- Sm_PolicyApi_Success. The response was successfully set or removed in a policy link.
- Sm_PolicyApi_Failure. The response cannot be set or removed in a policy link.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to set a policy link.
- Sm_PolicyApi_InvalidOid:
  - The policy OID was not found.
  - The rule or rule group OID was not found.
  - The response or response group OID was not found.
- Sm_PolicyApi_DoesNotExist. There are no policy links in the policy collection.

Sm_PolicyApi_SetSharedSecretPolicy()

Sets the current SharedSecretPolicy. There will always be exactly one such object, so it is not necessary to provide the bUpdate boolean flag.

Type

Agent configuration.

Syntax

```
int SM_EXTERN Sm_PolicyApi_SetSharedSecretPolicy (
   void*                               pSessionHandle,
   Sm_PolicyApi_SharedSecretPolicy_t*  ppstructSecretPolicy
);
```

Parameter	I/O	Description
pSessionHandle	I	The current Policy API session handle.
ppstructSecret Policy	I/O	The shared secret policy structure.

Returns

- Sm_PolicyApi_Success. The operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidHandle. The session handle was invalid.
- Sm_PolicyApi_NoSession. The user is not logged in.
- Sm_PolicyApi_NoPrivilege. User does not have rights to manage objects.

Sm_PolicyApi_SetUserDirSearchOrder()

Rearranges the search order of the user directory objects associated with the domain identified by *pszDomainOid*.

The ordered list of names is specified in the *pszArray* string array. The user directories in this array must match in OID and number (but not order) the list of user directory names that were retrieved by a call to Sm_PolicyApi_GetUserDirSearchOrder().

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_SetUserDirSearchOrder (
   void*          pSessionHandle,
   const char*    pszDomainOid,
   char**         pszArray[]
);
```

Parameter	**I/O**	**Description**
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszDomainOid	I	A null-terminated string containing the object identifier of an existing domain.
pszArray	I	An array of user directory OIDs, in the desired order.

Returns

- Sm_PolicyApi_Success. The user directory search order was set successfully.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_NoPrivilege. The administrator does not have the privilege to set user directory search order in a domain.
- Sm_PolicyApi_InvalidOid. The domain OID was not found.
- Sm_PolicyApi_WrongNumberOfElements. The number of user directories in the domain collection is different from the number of elements in the array.

Sm_PolicyApi_SetUserPasswordState()

Add or update a UserPasswordState object.

If there is no PasswordState associated with the user, a new PasswordState will be created. Otherwise, the UserPasswordState will be updated.

Type

Password state function.

Syntax

```
int SM_EXTERN Sm_PolicyApi_SetUserPasswordState (
   void                             *pSessionHandle,
   const char                       *pszUserDirOid,
   const char                       *pszUserDN,
   Sm_PolicyApi_UserPasswordState_t *pPasswordState
   bool bEmptyHistory
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	Unique object identifier that corresponds to a particular User Directory.
pszUserDN	I	Specifies the distinguished name of the user within the user directory.

Parameter	I/O	Description
pPasswordState	I/O	The user password state object. If this parameter changes the user directory setting for the last time the password was changed, and the password is reset outside of SiteMinder, the password policy preventing password reuse may not work as expected. If this parameter is set to NULL, the function returns Sm_PolicyApi_Failure.
bEmptyHistory	I	Specifies whether this function should clear the existing password history. If this parameter is set to true, the field *tLastPWChange* field of structure Sm_PolicyApi_UserPasswordState_t is implicitly reset to zero.

Returns

- Sm_PolicyApi_Success. The get operation was successful.
- Sm_PolicyApi_Failure. Generalized failure.
- Sm_PolicyApi_InvalidHandle. The session pointer is not valid.
- Sm_PolicyApi_BadArgument. The input password state pointer is NULL.
- Sm_PolicyApi_NoSession. The API user is not properly logged in.
- Sm_PolicyApi_InvalidOid. The unique id does not correspond to User Directory object.
- Sm_PolicyApi_UserDirNotFound. Failed to retrieve User Directory object from policy store.
- Sm_PolicyApi_ErrorUserDir. Invalid User Directory object or wrong user DN.
- Sm_PolicyApi_NoPrivilege. The caller does not have the proper privileges.

Sm_PolicyApi_ValidateDirectoryEntry()

Validates a user specification in a user directory.

Type

User directory function, global scope.

Syntax

```
int SM_EXTERN Sm_PolicyApi_ValidateDirectoryEntry (
   void*                  pSessionHandle,
   const char*            pszUserDirOid,
   const char*            pszPath,
   Sm_PolicyApi_User_t**  ppUserEntry
);
```

Parameter	I/O	Description
pSessionHandle	I	A pointer to an internal Policy Management API data structure. The structure holds information about the administrator session and the client session.
pszUserDirOid	I	A null-terminated string containing the object identifier of the user directory.
pszPath	I	A null-terminated string containing the path of a user.
ppUserEntry	O	The address of a pointer to a user structure.

Returns

- Sm_PolicyApi_Success. The validation was successful.
- Sm_PolicyApi_Failure:
  - The policy resolution of the user path in *pszPath* is not of type Sm_PolicyResolution_User.
  - There is no connection to the user directory or the user directory provider has not been loaded.
  - Memory could not be allocated to Sm_PolicyApi_User_t.
- Sm_PolicyApi_NoSession. There is no valid administrator session.
- Sm_PolicyApi_InvalidHandle. There was no valid initialization prior to this call.
- Sm_PolicyApi_InvalidOid. The user directory OID was not found.

Authentication Scheme Configuration

When you configure an authentication scheme programmatically, you provide information that would otherwise be provided through the Authentication Scheme Properties dialog of the Policy Server UI. You provide this information through the fields in the structure Sm_PolicyApi_Scheme_t.

Note: The following categories of information can be used for different purposes in different authentication schemes. For example, with the TeleID authentication scheme, the shared secret is used to supply the encryption seed.

- 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 C Policy Management API was deprecated in SDK 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 to save the user's credentials.

- 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 True (1), password policies are disabled.

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

Anonymous Template

Use this table when configuring an authentication scheme based on the scheme type Anonymous. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

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

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_Anonymous The scheme type Anonymous.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel*=0 Set to 0. Not applicable to this scheme type.
Library	*pszLib*="smauthanon" The default library for this scheme type.
Parameter	*pszParam=param* A string containing the guest DN. Policies associated with the guest DN must apply to anonymous users.
Shared secret	*pszSecret*="" Set to an empty string. Not applicable to this scheme.

Information Type	Value Assignment and Meaning
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (1)-ignore password checking.

Basic Over SSL Template

Use this table when configuring an authentication scheme based on the scheme type Basic over SSL. The structure fields referenced in the table
are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_BasicOverSSL The scheme type Basic over SSL.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 10.
Library	*pszLib*="smauthcert" The default library for this scheme type.

Information Type	Value Assignment and Meaning
Parameter	*pszParam=param* 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	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0) for this scheme.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck=flag* Set to true (1) to ignore password checking, or false (0) to check passwords. Default is 0.

Basic Template

Use this table when configuring an authentication scheme based on the scheme type Basic. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_Basic The scheme type Basic.
Description	*pszDesc=description* The description of the authentication scheme.

Information Type	Value Assignment and Meaning
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthdir" The default library for this scheme type.
Parameter	*pszParam*="" Set to an empty string. Not applicable to this scheme.
Shared secret	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=1 Set to true (1)-scheme can be used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=1 Set to true (1)-scheme can be used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck=flag* Set to true (1) to ignore password checking, or false (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 Authentication API. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_Custom The scheme type Custom.

Information Type	Value Assignment and Meaning
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 0 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib=customLibName* The name of the custom shared library you created using the Authentication API.
Parameter	*pszParam=param* 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.
Shared secret	*pszSecret=secret* The shared secret, if any, that your custom authentication scheme uses for encryption of credentials.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin=flag* 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?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck=flag* Set to true (1) to ignore password checking, or false (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. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	**Value Assignment and Meaning**
Scheme type	*nType*=Sm_Api_SchemeType_HTMLForm The scheme type HTML Form.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthhtml" The default library for this scheme type.
Parameter	*pszParam=param* 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	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.

Information Type	Value Assignment and Meaning
Save credentials?	*bAllowSaveCreds=flag* Set to true (1) to indicate that user credentials should be saved, or false (0) to indicate that user credentials should not be saved. Default is 0.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck=flag* Set to true (1) to ignore password checking, or false (0) to check passwords. Default is 0.

Impersonation Template

Use this table when configuring an authentication scheme based on the scheme type Impersonation. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_Impersonation The scheme type Impersonation.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthimpersonate" The default library for this scheme type.

Information Type	Value Assignment and Meaning
Parameter	*pszParam=param* 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/imp.fcc
Shared secret	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (1)-ignore password checking.

RADIUS CHAP/PAP Template

Use this table when configuring an authentication scheme based on the scheme type RADIUS CHAP/PAP. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	**Value Assignment and Meaning**
Scheme type	*nType*=Sm_Api_SchemeType_RadiusChapPap The scheme type RADIUS CHAP/PAP.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthchap" The default library for this scheme type.
Parameter	*pszParam=param* A string containing the name of a user directory attribute. This attribute is used as the clear text password for authentication.
Shared secret	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=1 Set to true (1)-scheme can be used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck=flag* Set to true (1) to ignore password checking, or false (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. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_RadiusServer The scheme type RADIUS Server.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthradius" The default library for this scheme type.
Parameter	*pszParam=param* 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	*pszSecret=secret* The user attribute that the RADIUS Server will use as the clear text password.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=1 Set to true (1)-scheme can be used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=1 Set to true (1)-scheme can be used with RADIUS agents.

Information Type	**Value Assignment and Meaning**
Ignore password check?	*bIgnorePwCheck=flag* Set to true (1) to ignore password checking, or false (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 HTML Form. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	**Value Assignment and Meaning**
Scheme type	*nType*=Sm_Api_SchemeType_SafeWordHTMLForm The scheme type SafeWord HTML Form.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 10.
Library	*pszLib*="smauthenigmahtml" The default library for this scheme type.
Parameter	*pszParam=param* 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	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=1 Set to true (1)-scheme can be used to authenticate administrators.

Information Type	Value Assignment and Meaning
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=1 Set to true (1)-scheme can be used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (1)-ignore password checking.

SafeWord Template

Use this table when configuring an authentication scheme based on the scheme type SafeWord. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_SafeWordServer The scheme type SafeWord.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 10.
Library	*pszLib*="smauthenigma" The default library for this scheme type.
Parameter	*pszParam*="" Set to an empty string. Not applicable to this scheme.
Shared secret	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=1 Set to true (1)-scheme can be used to authenticate administrators.

Information Type	Value Assignment and Meaning
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=1 Set to true (1)-scheme can be used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (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.

The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_SAMLArtifact The scheme type SAML Artifact.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthsaml" The default library for this scheme type.
Parameter	*pszParam=param* 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:

Information Type	Value Assignment and Meaning
	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 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 portal and the affiliate. This value is compared with the audience value specified in the SAML assertion. ■ Issuer. The SAML issuer specified in the assertion. ■ 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.

Information Type	Value Assignment and Meaning
	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	*pszSecret=secret* The password for the affiliate site.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (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.

The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_SAMLPOST The scheme type SAML POST.
Description	*pszDesc=description* The description of the authentication scheme.

Information Type	Value Assignment and Meaning
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthsaml" The default library for this scheme type.
Parameter	*pszParam=param* 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 portal 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	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (1)-ignore password checking.

SAML 2.0 Template

Use this table when configuring a SAML authentication scheme based on the SAML 2.0 scheme type. A Service Provider uses this authentication scheme to transparently validate a user based on the information in a SAML 2.0 assertion. This transparent validation allows functionality such as single sign-on and single logout.

When you configure a SAML 2.0 authentication scheme, you also define metadata properties for the associated Identity Provider-that is, the Identity Provider that supplies the assertion to the Service Provider.

The properties of the Identity Provider are stored with the authentication scheme object as a separate set of properties. As a result, two structures are used to configure a SAML 2.0 authentication scheme:

- The structure fields referenced in the following table are in Sm_PolicyApi_Scheme_t.
- The metadata properties for the associated Identity Provider are defined through Sm_PolicyApi_SAMLProviderProp_t.

Information Type	**Value Assignment and Meaning**
Scheme type	*nType*=Sm_Api_SchemeType_SAML2 The scheme type SAML 2.0.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthsaml" The default library for this scheme type.
Parameter	*pszParam*="" Set to an empty string. SiteMinder assigns a parameter value. The parameter is a reference to the SAML 2.0 metadata properties for the associated Identity Provider. The properties are defined through Sm_PolicyApi_SAMLProviderProp_t.
Shared secret	*pszSecret*="" Set to an empty string. Not applicable to this scheme.

Information Type	Value Assignment and Meaning
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (1)-ignore password checking.

More Information:

Custom Agents and Single Sign-On (see page 43)

SecurID HTML Form Template

Use this table when configuring an authentication scheme based on the scheme type SecurID HTML Form. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_ACEServerHTMLForm The scheme type SecurID HTML Form.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 15.
Library	*pszLib*="smauthacehtml" The default library for this scheme type.

Information Type	Value Assignment and Meaning
Parameter	*pszParam=param* 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 SecurID authentication with forms support. It also specifies whether an SSL connection is required. Format: *attr*;https://*server*/*target* **Note:** The "s" in "https" is optional, depending on whether you want an SSL connection. The following example uses the default for processing SecurID authentication with forms support: ace_id;https://my.server.com/ siteminderagent/forms/smpwservices.fcc .
Shared secret	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (1)-ignore password checking.

SecurID Template

Use this table when configuring an authentication scheme based on the scheme type SecurID. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_ACEServer The scheme type SecurID.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 15.
Library	*pszLib*="smauthace" The default library for this scheme type.
Parameter	*pszParam=param* A string containing the attribute in the authentication user directory that contains the ACE Server user ID.
Shared secret	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=1 Set to true (1)-scheme can be used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=1 Set to true (1)-scheme can be used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (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 smauthetsso authentication scheme, which is based on the Custom scheme type. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	**Value Assignment and Meaning**
Scheme type	*nType*=Sm_Api_SchemeType_Custom Uses the Custom scheme type
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 0 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthetsso" The name of the library of this authentication scheme.

Information Type	Value Assignment and Meaning
Parameter	*pszParam=param* 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 eTrust 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 eTrust Policy Server on the <*eTPS_Host*> so that SiteMinder can request validation of eTrust SSO cookies. Examples: pszParam="cookie; SMPS_sso; myserver.myco.com" pszParam="cookieorforms; /siteminderagent/forms/login.fcc; SMPS_sso; myserver.myco.com"
Shared secret	*pszSecret=secret* The password of the eTrust Policy Server administrator named in the Paramter field.
Is template?	*bIsTemplate*=0 Set to false (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?	*bIsUsedbyAdmin=flag* 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?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck=flag* Set to true (1) to ignore password checking, or false (0) to check passwords. Default is 0.

TeleID Template

Use this table when configuring an authentication scheme based on the scheme type TeleID. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_Encotone The scheme type TeleID.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 15.
Library	*pszLib*="smauthencotone" The default library for this scheme type.
Parameter	*pszParam*="" Set to an empty string. Not applicable to this scheme.
Shared secret	*pszSecret=seed* The encryption seed. SiteMinder uses this value as an encryption seed for initializing hardware tokens.

Information Type	Value Assignment and Meaning
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=1 Set to true (1)-scheme can be used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=1 Set to true (1)-scheme can be used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (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.

The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_NTLM The scheme type Windows Authentication (NTLM).
Description	*pszDesc=description* The description of the authentication scheme.

Information Type	Value Assignment and Meaning
Protection level	*nLevel*=*value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthntlm" The default library for this scheme type.
Parameter	*pszParam*=*param* The value of *pszParam* 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, *pszParam* 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	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (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?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck=flag* 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

Use this table when configuring a WSFED authentication scheme based on the WSFED scheme type. A Resource Partner uses this authentication scheme to transparently validate a user based on the information in a SAML 1.0 assertion. This transparent validation allows functionality such as single sign-on and single logout.

When you configure a WSFED authentication scheme, you also define metadata properties for the associated Account Partner, that is, the Account Partner that supplies the assertion to the Resource Partner.

The properties of the Account Partner are stored with the authentication scheme object as a separate set of properties. As a result, two structures are used to configure a WSFED authentication scheme:

- The structure fields referenced in the following table are in Sm_PolicyApi_Scheme_t.
- The metadata properties for the associated Account Partner are defined through Sm_PolicyApi_WSFEDProviderProp_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_WSFED The scheme type WSFED.

Information Type	**Value Assignment and Meaning**
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthsaml" The default library for this scheme type.
Parameter	*pszParam*="" Set to an empty string. SiteMinder assigns a parameter value. The parameter is a reference to the WSFED metadata properties for the associated Account Partner. The properties are defined through Sm_PolicyApi_WSFEDProviderProp_t.
Shared secret	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (1)-ignore password checking.

More Information:

WS-Federation (see page 124)

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. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*= Sm_Api_SchemeType_X509ClientCertAndBasic The scheme type X.509 Client Certificate and Basic.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 15.
Library	*pszLib*="smauthcert" The default library for this scheme type.
Parameter	*pszParam=param* 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	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.

Information Type	Value Assignment and Meaning
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck=flag* Set to true (1) to ignore password checking, or false (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. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*= Sm_Api_SchemeType_X509ClientCertAndForm The scheme type X.509 Client Certificate and HTML Form.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 15.
Library	*pszLib*="smauthcert" The default library for this scheme type.
Parameter	*pszParam=param* 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	*pszSecret*="" Set to an empty string. Not applicable to this scheme.

Information Type	Value Assignment and Meaning
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to 0-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to 0 to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to 0-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck=flag* 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. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*= Sm_Api_SchemeType_X509ClientCertOrBasic The scheme type X.509 Client Certificate or Basic.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthcert" The default library for this scheme type.

Information Type	Value Assignment and Meaning
Parameter	*pszParam=param* 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
Shared secret	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck=flag* Set to true (1) to ignore password checking, or false (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. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*= Sm_Api_SchemeType_X509ClientCertOrForm The scheme type X.509 Client Certificate or HTML Form.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthcertorform" The default library for this scheme type.
Parameter	C*pszParam=param* 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

Information Type	Value Assignment and Meaning
Shared secret	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to 0-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to 0 to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to 0-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck=flag* 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. The structure fields referenced in the table are in Sm_PolicyApi_Scheme_t.

Information Type	Value Assignment and Meaning
Scheme type	*nType*=Sm_Api_SchemeType_X509ClientCert The scheme type X.509 Client Certificate.
Description	*pszDesc=description* The description of the authentication scheme.
Protection level	*nLevel=value* A value of 1 through 1000. The higher the number, the greater degree of protection provided by the scheme. Default is 5.
Library	*pszLib*="smauthcert" The default library for this scheme type.

Information Type	Value Assignment and Meaning
Parameter	*pszParam=param* 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	*pszSecret*="" Set to an empty string. Not applicable to this scheme.
Is template?	*bIsTemplate*=0 Set to false (0) to indicate that the scheme is not a template. Any other value is ignored.
Is used by administrator?	*bIsUsedbyAdmin*=0 Set to false (0)-scheme is not used to authenticate administrators.
Save credentials?	*bAllowSaveCreds*=0 Set to false (0) to indicate that user credentials won't be saved.
Is RADIUS?	*bIsRadius*=0 Set to false (0)-scheme is not used with RADIUS agents.
Ignore password check?	*bIgnorePwCheck*=1 Set to true (1)-ignore password checking.

Chapter 5: Customizing the Policy Server

The following APIs allow you to develop custom extensions to the Policy Server:

- Authentication API
- Authorization API
- Event API
- Directory API

For example, you can create the following:

- An active policy that provides dynamic authorization based on external business logic
- An active response that returns values from external data sources
- An active rule that provides dynamic authorization based on external business logic
- A custom authentication scheme
- An event handler that captures events issued by the Policy Server,
- A user directory from data in a custom namespace

This section contains the following topics:

Work with Authentication and Authorization APIs

The Policy server exposes a number of hooks that can be used to perform custom authentication and authorization tasks. In addition, hooks can be used to enable delivery of custom data to applications that are protected by SiteMinder Agents. Using these hooks enables tight integration between SiteMinder and the existing infrastructure.

Authentication Hook

To use this hook, you must build a custom authentication scheme using the Authentication API. A custom authentication scheme can assume control over user disambiguation, authentication, and impersonation issues. A custom authentication scheme can accept or reject a user's authentication attempt, and can specify a numeric reason and a text message for doing so. The numeric reason and the text message are made available to the application so that the application can take the appropriate action. This enables the application to maintain communication with the authentication scheme until the user is finally authenticated.

Authorization Hook

To use this hook, you must build a custom authorization module using the Authorization API. A custom authorization module controls whether or not SiteMinder rules and policies allow the user to access a resource.

Custom Data Hook

To use this hook, you must build a custom module that obtains data from an external data source and returns this data to the Policy Server. The Policy Server then delivers the data to the application protected by SiteMinder. SiteMinder does not interpret this data in any way.

SiteMinder Events

The Policy Server defines a number of events that may be triggered after each authentication and authorization. SiteMinder policies are used to catch these events and return appropriate data to applications protected by SiteMinder Agents. Events can be used to implement application workflow. There are two types of events: authentication and authorization.

Authentication Events

The Policy Server fires an authentication event based on the outcome of the authentication. Authentication events include:

- OnAuthAccept
- OnAuthReject
- OnAuthChallenge
- OnAuthAttempt

By using proper authentication events, responses can be accumulated and delivered to the application immediately after a user is authenticated.

Authorization Events

The Policy Server fires an authorization event based on the outcome of the authorization. Authorization events include:

- OnAccessAccept
- OnAccessReject

By using proper authorization events, responses can be accumulated and delivered to the application immediately after a user is authorized to access a resource.

Saving Data Between Module Invocations

When the Policy Server makes calls to custom modules, it may be convenient for these modules to communicate some request-specific information between themselves. Custom modules built with the Authentication, Authorization or Directory APIs can preserve state between module invocations. To preserve state, use the memory management hooks provided in the Sm_Api_Context_t structure (defined in SmApi.h). State has request-only scope and is limited to a single instance of a process; that is, a custom module running in the context of the authentication server cannot exchange state with a module running in the context of the authorization server.

Well-known User Attributes

Sometimes it may be necessary to return data that describes a user property that is not stored in the user profile. Typically, this data is some kind of dynamic information that is maintained by and known to SiteMinder. The dynamic information is provided using well-known or pseudo user attributes. These attributes can be accessed by the standard response mechanism or by a custom module if the attributes represent standard user attributes. These attributes begin with SM_ and are listed here:

SM_USERNAME

For an authenticated user, this is the user DN as disambiguated by SiteMinder. For an unauthenticated user, this is the user ID as specified by the user in the login attempt.

SM_USERIMPERSONATORNAME

If the authentication scheme performs impersonation, this is the user DN that is authenticated by SiteMinder.

SM_USERLOGINNAME

This is the user ID as specified by the user in the login attempt.

SM_USERLOGINFAILURESCOUNT

The number of attempted logins that failed, expressed as an int value. One of the password policy state attributes.

SM_USERIPADDRESS

The user’s IP address at the time of authentication or authorization.

SM_USERPATH

For an authenticated user, this is a string that represents the directory namespace and directory server (both as specified in the user directory definition), and user DN (as disambiguated by SiteMinder). For example:

```
“LDAP://123.123.0.1/uid=scarter,ou=people,o=airius.com”
```

For an unauthenticated user, this is the same as SM_USERNAME.

SM_USERPREVIOUSLOGINTIME

Time of the user’s previous login, expressed as a GMT time_t value. One of the password policy state attributes. This value is read-only (get); it cannot be set. The value represents the time since the Policy Server system epoch time, which is 00:00:00 UTC, January 1, 1970. (If a Mac OS machine is running as a client, its epoch time is 00:00:00, January 1, 1904, local time.) One of the password policy state attributes.

SM_USERPASSWORD

This is the password as specified by the user in the login attempt. The value is returned only on authentication, not on authorization.

SM_TRANSACTIONID

The transaction ID that is generated by the agent.

SM_USERDISABLEDTIME

Time that the user has been disabled, expressed as a GMT time_t value. One of the password policy state attributes. This value is read-only (get); it cannot be set. The value represents the time since the Policy Server system epoch time, which is 00:00:00 UTC, January 1, 1970. (If a Mac OS machine is running as a client, its epoch time is 00:00:00, January 1, 1904, local time.) One of the password policy state attributes.

SM_USERLASTPWCHANGETIME

Time that the user last changed the password, expressed as a GMT time_t value. The value represents the time since the Policy Server system epoch time, which is 00:00:00 UTC, January 1, 1970. (If a Mac OS machine is running as a client, its epoch time is 00:00:00, January 1, 1904, local time.) One of the password policy state attributes.

Note: If you set this value, and the associated password is reset outside of SiteMinder, the password policy preventing password reuse may not work as expected

SM_USERPASSWORDHISTORY

One of the password policy state attributes. Contains all the password change information for up to 32 entries, expressed as a string value. Includes encrypted, structured, binary information. The Password State API does not expose any detail about this structure to the external applications.

Note: You can "set" the value of this attribute only by using an empty string to clear the entire history. No other history changes are allowed. You cannot add or remove only one history entry. There is no read (get) for this attribute.

SM_USERSESSIONSPEC

The user’s session specification.

SM_USERSESSIONID

The session ID of a user who has already been authenticated. This is the session ID that will be assigned to the user upon successful authentication.

SM_USERSESSIONIP

The IP address that was used during the original user authentication (upon establishment of a session).

SM_USERSESSIONUNIVID

The user’s universal ID. If no universal ID directory attribute is specified in the user directory definition, this defaults to the user’s DN.

SM_USERSESSIONDIRNAME

The name of the user directory that the Policy Server is configured to use.

SM_USERSESSIONDIROID

The object ID of the user directory that the Policy server is configured to use.

SM_USERSESSIONTYPE

The user’s session type. One of the following:

- 2 - session
- 1 - identity

SM_USERLASTLOGINTIME

The user’s last login time, expressed as a GMT time_t value. Available only during authentication with applicable password services policies. One of the password policy state attributes.

SM_USERGROUPS

Groups to which the user belongs. If the user belongs to a nested group, this attribute contains the group furthest down in the hierarchy. For all nested groups to which the user belongs, use SM_USERNESTEDGROUPS.

For example, if user JSmith belongs to the group Accounts Payable, which is contained in group Accounting, SM_USERGROUPS contains Accounts Payable. If you want both Accounting and Accounts Payable, use SM_USERNESTEDGROUPS.

SM_USERNESTEDGROUPS

Nested groups to which the user belongs. For only the group furthest down in the hierarchy, use SM_USERGROUPS.

For example, if user JSmith belongs to the group Accounts Payable, which is contained in group Accounting, SM_USERNESTEDGROUPS contains Accounting and Accounts Payable. If you want only Accounting, use SM_USERGROUPS.

SM_USERSCHEMAATTRIBUTES

User attributes associated with the DN, or properties associated with the user. If the user directory is a SQL database, then SM_USERSCHEMAATTRIBUTES holds the names of the columns in the table where user data is stored. For example, using the SmSampleUsers schema, SM_USERSCHEMAATTRIBUTES holds the names of the columns in the SmUser table.

SM_USERPOLICIES

These contain the names of the policies that authorize the user for a resource. For example, suppose that to purchase an item, a user must be associated with the Buyer policy. When the user is successfully authorized to buy an item, SM_USERPOLICIES will contain Buyer.

SM_USERPOLICIES will only contain a value when the action for the associated rule is set to Authorization events and the specified event is OnAccessAccept.

SM_USERPRIVS

When a user is authenticated or is authorized for a resource, SM_USERPRIVS holds all of the response attributes for all policies that apply to that user, in all policy domains.

SM_USERREALMPRIVS

When a user is authenticated or is authorized for a resource under a realm, SM_USERREALMPRIVS holds all the response attributes for all rules under that realm.

For example, suppose that there is a realm called Equipment Purchasing. Under that realm, there is a CheckCredit rule. Associated with the CheckCredit rule is a response that returns the buyer’s credit limit, such as limit = $15000, as a response attribute. If the buyer attempts to purchase equipment worth $5000, the CheckCredit rule fires. SM_USERREALMPRIVS would contain all of the response attributes for all of the rules under the Equipment Purchasing realm.

SM_USERDISABLEDSTATE

Returns a decimal number that represents a bit mask of reasons that a user is disabled. The bits are defined in SmApi.h under the Sm_Api_DisabledReason_t data structure.

For example, a user may be disabled as a result of inactivity, Sm_Api_Disabled_Inactivity. In Sm_Api_DisabledReason_t, the reason Sm_Api_Disabled_Inactivity, corresponds to the value 0x00000004. So, in this case, SM_USERDISABLEDSTATE is 4..

NTUSERNAME

Returns the username portion of the fully qualified WinNT name—for example, for mydomain\myname, myname is returned.

NTFULLUSERNAME

Returns the fully qualified WinNT name.

Retrieving a Password

Some applications require knowledge of user credentials. SiteMinder can make the user password available to the application by returning a well-known attribute SM_USERPASSWORD. This attribute is only available after a successful authentication through the OnAuthAccept event.

The following example demonstrates retrieving a password.

```
char  lpszSmPassword(128);
int   nBytes =

lpUserContext->fGetProp(lpUserContext->lpParam,
                        “SM_USERPASSWORD”,
                        sizeof (lpszSmPassword),
                        lpszSmPassword);
```

Integration with the Web Agent

The Policy Server works in tandem with the Web Agent. While it is authenticating and authorizing, the Policy server sends certain information about the outcome of the operation to the Web Agent. The Web Agent makes this information available to the application as follows:

Authentication events without a redirect:

- The user message is exposed as the HTTP_SM_USRMSG header variable.
- The reason code is exposed as the HTTP_SM_AUTHREASON header variable.

Authentication events with a redirect:

- The user’s message is stored in the SMUSERMSG cookie as SMUSRMSG=<text>.
- The reason code is appended to the redirect URL as ?SMAUTHREASON=<reason code>

  If the reason code is included in the following form:

  SMAUTHREASON=$$SMAUTHREASON$$

  the value is used in the authentication reason field for SM_AGENTAPI_LOGIN, along with the username and password.
- The redirect text is stored in the SMTEXT cookie as SMTEXT=<text>.

Authorization events without a redirect:

- No reason code is available.

Authorization events with a redirect:

- The reason code is stored in the SMREASON cookie as SMREASON=<reason code>.
- The redirect text is stored in the SMTEXT cookie as SMTEXT=<text>.

Chapter 6: Context Structures

This section contains the following topics:

Sm_Api_AppSpecificContext_t

This structure contains shared memory information.

Syntax

The syntax of this structure is:

```
typedef struct
{
   void* pHandle;
   Sm_Api_GetBufferFunc fGetBuffer;
   Sm_Api_AllocBufferFunc fAllocBuffer;
   Sm_Api_FreeBufferFunc fFreeBuffer;
   Sm_Api_GetBufferSizefunc fGetBufferSize;
} Sm_Api_AppSpecificContext_t;
```

Field	Description
pHandle	A handle to be passed, required by memory management functions. This handle is obtained from the *pAppSpecific* field of Sm_Api_Context_t.
fGetBuffer	Function that returns a pointer to the previously allocated request-specific memory or NULL if no memory was allocated. *pHandle* is the function’s only parameter.

Field	Description
fAllocBuffer	Function that allocates request-specific memory and returns a pointer to the allocated buffer. It returns NULL if the buffer has already been allocated or no memory is available. Parameters are *pHandle* and an int specifying the size of the buffer.
fFreeBuffer	Function that releases any request-specific memory associated with the passed *pHandle* argument. No value is returned.
fGetBufferSize	Function that retrieves the size of the buffer.

Function Declarations

In structure Sm_Api_AppSpecificContext_t, the functions fGetBuffer, fAllocBuffer, fFreeBuffer, and fGetBufferSize are declared in SmApi.h as follows:

fGetBuffer

```
typedef void* (SM_EXTERN *Sm_Api_GetBufferFunc)
    (const void *pHandle);
```

fAllocBuffer

```
typedef void* (SM_EXTERN *Sm_Api_AllocBufferFunc)
    (const void *pHandle, const int nSize);
```

fFreeBuffer

```
typedef void (SM_EXTERN *Sm_Api_FreeBufferFunc)
   (const void *pHandle);
```

fGetBufferSize

```
typedef int (SM_EXTERN *Sm_Api_GetBufferSizeFunc)
   (const void *pHandle);
```

Example

```
// Allocate 64 bytes
char* pBuffer = (char*) lpApiContext->pAppSpecific->fAllocBuffer
                (lpApiContext->pAppSpecific->pHandle, 64);
strcpy (pBuffer, "id=5");
...

// Use it somewhere else
char id[20];
char* pBuffer = (char*) lpApiContext->pAppSpecific->fGetBuffer
                (lpApiContext->pAppSpecific->pHandle);
if (pBuffer != NULL)
{
   strcpy (id, pBuffer);
}

//Free the memory when done
...
(char*) lpApiContext->pAppSpecific->fFreeBuffer
                (lpApiContext->pAppSpecific->pHandle);
```

Sm_Api_Context_t

This structure contains general API information.

Syntax

```
typedef struct
{
   int nVersion;
   Sm_Api_LogFunc fLog;
   Sm_Api_TraceFunc fTrace;
   Sm_Api_ErrorFunc fError;
   Sm_Api_AppSpecificContext_t*  pAppSpecific;
} Sm_Api_Context_t;
```

Field	Description
nVersion	Version number of the API being used. Possible values: ■ Sm_Api_Version_V3 ■ Sm_Api_Version_V4 ■ Sm_Api_Version_V4_1 Version constants are defined in SmApi.h.

fLog	Function for accessing the SiteMinder logging utility. SiteMinder must be running with logging enabled.
	The calling syntax for this function is:
	char* lpszMessage = "Log this text";
	lpApiContext->fLog (lpszMessage);

fTrace	Function for accessing the SiteMinder trace utility. SiteMinder must be running with tracing enabled.
	The calling syntax for this function is:
	char* lpszCheckpoint = "MyLib";
	char* lpszTraceMessage = "Trace text";
	lpApiContext->fTrace (lpszCheckpoint,
	                    lpszTraceMessage);

fError	Function for accessing the SiteMinder error utility. SiteMinder must be running with logging enabled.
	The calling syntax for this function is:
	char* lpszMessage = "Log this Error text";
	lpApiContext->fError (lpszMessage);

Field	Description
pAppSpecific	The *pAppSpecific* pointer has access to the methods in the Sm_Api_AppSpecificContext_t structure. These methods allocate, delete, and release request-specific memory. This memory persists during the entire processing of a single agent request and is local to each server process such as authentication and authorization daemons. There is no limit as to the number of times memory can be allocated and released by a module. Once a request is processed, memory is released automatically.

Function Declarations

In structure Sm_Api_Context_t, the functions fLog, fTrace, and fError are declared in SmApi.h as follows:

fLog

```
/* string to log (null-terminated) */
typedef void (SM_EXTERN *Sm_Api_LogFunc) (const char* lpszBuffer);
```

fTrace

```
/* string to log (null-terminated) */
typedef void (SM_EXTERN *Sm_Api_TraceFunc)
   (const char* lpszCheckpoint, const char* lpszBuffer);
```

fError

```
/* string to log (null-terminated) */
typedef void (SM_EXTERN *Sm_Api_ErrorFunc)
   (const char* lpszBuffer);
```

Sm_Api_RequestContext_t

Contains information about the request being made.

Syntax

```
typedef struct
{
   char* lpszServer;
   char* lpszResource;
   char* lpszAction;
} Sm_Api_RequestContext_t;
```

Field	Description
lpszServer	Server part of the user access request. For example, if the user accesses http://www.server.com/index.html, the server is www.server.com.
lpszResource	Resource part of the user access request. For example, if the user accesses http://www.server.com/index.html, the resource is /index.html.
lpszAction	The action as requested by the user. For example, the typical action when accessing Web resources is a GET.

Sm_Api_TunnelContext_t

Holds information passed from the Tunnel Agent.

Syntax

```
typedef struct
{
   char* lpszClientIp;
   char* lpszTransactionId;
   char* lpszParameter;
} Sm_Api_TunnelContext_t;
```

Field	Description
lpszClientIP	The IP Address of the Tunnel Agent making the tunnel calls.
lpszTransactionId	(Optional) The ID that the agent uses to associate application activity with security activity. The Policy Server logs this ID.
lpszParameter	Arbitrary string parameter passed from the Tunnel Agent. On the Tunnel Agent side, this information is passed in *lpszParameter* in Sm_AgentApi_TunnelServiceRequest_t.

Sm_Api_UserContext_t

Contains information about the user.

Syntax

```
typedef struct
{
   unsigned char  bIsUserContext;
   char* lpszUserName;
   char* lpszUserPath;
   char* lpszDirPath;
   void* lpReserved1;
   char* lpszDirServer;
   char* lpszDirNamespace;
   char* lpszSessionId;
   Sm_Api_GetDnProp fGetDnProp;
   Sm_Api_SetDnProp fSetDnProp;
   void* lpParam;
   Sm_Api_GetUserProp fGetProp;
   Sm_Api_SetUserProp fSetProp;
   Sm_Api_AuthenticateUser fAuthenticate;
} Sm_Api_UserContext_t;
```

Field	Description
bIsUserContext	Flag indicating that SiteMinder has established the user's identity and that the user context is available. When this flag is set, the *fGetProp*, *fSetProp*, *fGetDnProp,* and *fSetDnProp* user attribute functions are available.
lpszUserName	Full distinguished name of the user.
lpszUserPath	User path in the following format: *directory-namespace* + *server* + / + *user-DN* For example: ldap://server.company.com/ uid=user1,ou=people,o=company.com
lpszDirPath	Directory path of the SiteMinder user directory where the user context was established, in the following format: *directory-namespace* + *server* For example: ldap://server.company.com
lpReserved1	Reserved for internal use.

Field	Description
lpszDirServer	Directory server of a SiteMinder user directory where user's context was established.
lpszDirNamespace	Directory namespace such as LDAP:, AD:, WinNT:, or ODBC:.
lpszSessionId	Session ID that has been or will be assigned to the user's session, depending on whether or not the session has been established.

fGetDnProp	Function that returns an attribute of a directory entry. If the user context flag bIsUserContext is set, developers can call this function to retrieve a well-known attribute of any DN that the user is related to in the context of a directory (for example, user is a member of a group).
	The calling syntax for this function is:
	`if (lpUserContext->bIsUserContext)`
	`{`
	`char lpszDN[]="cn=group,ou=org unit,o=org";`
	`char lpszCommonName[100];`
	`int nBytes = lpUserContext->fGetDnProp(`
	`        lpUserContext->lpParam,`
	`        lpszDN,`
	`        "accesslevel",`
	`        sizeof (lpszCommonName),`
	`        lpszCommonName);`
	`}`
	If no error occurs, the function places the value of the requested attribute in the null-terminated output buffer and returns its length. Otherwise, the function returns –1.
	The attribute returned from this function should not be larger than the maximum buffer size specified in the *nBytesValueBuf* argument. Larger attributes are truncated to *nBytesValueBuf*.

Field	Description

fSetDnProp	Function that sets an attribute of a directory entry. If the user context flag bIsUserContext is set, developers can call this function to set a well-known attribute of any DN that the user is related to in the context of a directory (for example: user is a member of a group). At this time only attributes of type ‘string’ are supported.
	The calling syntax for this function is:
	if (lpUserContext->bIsUserContext)
	{
	char lpszDN[]="cn=group,ou=org unit,o=org";
	char lpszTimestamp[] = "<timestamp>";
	int nErr = lpUserContext->fSetDnProp (
	        lpUserContext->lpParam,
	        lpszDN,
	        "lastaccess",
	        sizeof (lpszTimestamp),
	        lpszTimestamp);
	}

lpParam	Pointer to the parameters to be passed to *fGetProp*, *fSetProp*, *fGetDnProp*, and, *fSetDnProp* functions.

Field	Description

fGetProp	Function that returns user attributes. If the user context flag bIsUserContext is set, developers can call this function to retrieve a well-known user attribute.
	The calling syntax for this function is:
	`if (lpUserContext->bIsUserContext)`
	`{`
	`char lpszCommonName[100];`
	`int nBytes = lpUserContext->fGetProp (`
	`        lpUserContext->lpParam,`
	`        "cn",`
	`        sizeof (lpszCommonName),`
	`        lpszCommonName);`
	`}`
	If no error occurs, the function places the value of the requested attribute in the null-terminated output buffer and returns its length. Otherwise, the function returns -1.
	The attribute returned from this function should not be larger than the maximum buffer size specified in the *nBytesValueBuf* argument. Larger attributes are truncated to *nBytesValueBuf*.

fSetProp	Function that sets a user attribute. If the user context flag bIsUserContext is set, developers can call this function to set a well-known user attribute. At this time, only attributes of type “string” are supported.
	The calling syntax for this function is:
	`if (lpUserContext->bIsUserContext)`
	`{`
	`char lpszCommonName[] = "John Smith";`
	`int nErr = lpUserContext->fSetProp (`
	`        lpUserContext->lpParam,`
	`        "cn",`
	`        sizeof (lpszCommonName),`
	`        lpszCommonName);`
	`}`
	If no error occurs, the function returns 0. Otherwise, the function returns -1.

Function Declarations

In structure Sm_Api_UserContext_t, the functions fGetDnProp, fSetDnProp, fGetProp, fSetProp, and fAuthenticate are declared in SmApi.h as follows:

fGetDnProp

```
typedef int (SM_EXTERN *Sm_Api_GetDnProp)
(
const void* lpParam,        /* The function parameter */
const char* lpDn,           /* The DN of a directory object */
const char* lpszPropName,   /* User property name (null-term) */
const int nBytesValueBuf,   /* Max size of user property buffer */
char* lpszValueBuf /* Output buffer to hold the user property */
);
```

fSetDnProp

```
typedef int (SM_EXTERN *Sm_Api_SetDnProp)
(
const void* lpParam,        /* The function parameter */
const char* lpDn,           /* The DN of a directory object */
const char* lpszPropName,   /* User property name (null-term) */
const int nBytesValueBuf,   /* Size of user property buffer */
const char* lpszValueBuf    /* The user property buffer */
);
```

fGetProp

```
typedef int (SM_EXTERN *Sm_Api_GetUserProp)
(
const void* lpParam,        /* The function parameter */
const char* lpszPropName,   /* User property name (null-term) */
const int nBytesValueBuf,   /* Max size of user property buffer */
char* lpszValueBuf   /* Output buffer to hold the user property */
);
```

fSetProp

```
typedef int (SM_EXTERN *Sm_Api_SetUserProp)
(
const void* lpParam,        /* The function parameter */
const char* lpszPropName,   /* User property name (null-term) */
const int nBytesValueBuf,   /* Size of user property buffer */
const char* lpszValueBuf    /* The user property buffer */
);
```

fAuthenticate

```
typedef int (SM_EXTERN *Sm_Api_AuthenticateUser)
(
const void* lpParam,        /* The function parameter */
const char* lpszPassword,   /* User password (null-terminated) */
const int nBytesUserMsg,    /* Max size of user message buffer */
char* lpszUserMsg,   /* Output buffer to hold the user message */
const int nBytesErrMsg,    /* Maximum size of the error buffer */
char* lpszErrMsg    /* Output buffer to hold the error message */
);
```

More Information:

Multi-Valued Attributes in LDAP (see page 581)

Multi-Valued Attributes in LDAP

When setting or retrieving multi-valued attributes in an LDAP user store, the values are presented in a single string, delimited by a carat character (^).

For example, you might set three different telephone numbers as follows:

```
char lpszTemp[] = "111-1234^111-5678^111-0000";

int getResult = lpUserContext->fSetProp (lpUserContext->lpParam,
                                         "telephonenumber",
                                         strlen(lpszTemp),
                                         lpszTemp);
```

Custom code that sets or retrieves multi-valued attributes must support the expected multi-valued string format.

Note: ODBC user stores do not support multi-valued attribute settings.

Policy Server Version Support

With a 4.61 or later Policy Server, the telephone numbers in the example above are put into the user's telephone number attribute in the LDAP user store as follows:

```
111-1234
111-5678
111-0000
```

Prior to the 4.61 Policy Server, the above code would set the telephone number attribute as follows:

```
111-1234^111-5678^111-0000
```

Sm_AuthApi_UserCredentials_t

Contains information about user credentials.

SiteMinder specifies as much information as it has available. Typically, the username and password fields are specified. Each authentication scheme expects a subset of possible credentials.

Fields in this structure are filled in on the basis of requested credentials.

Syntax

```
typedef struct
{
   char* lpszUsername;
   char* lpszPassword;
   int   nBytesChapPassword;
   char* lpszChapPassword;
   int   nBytesChapChallenge;
   char* lpszChapChallenge;
   char* lpReserved1;
   char* lpszCertUserDN;
   char* lpszCertIssuerDN;
   int   nCertBinLen;
   void* lpCertBinary;
   char* lpszDirPath;
   char* lpszDirServer;
   char* lpszDirNamespace;
   char* lpszNewPassword;
   char* lpszPasswordToken;
} Sm_AuthApi_UserCredentials_t;
```

Field	**Description**
lpszUsername	User's DN as disambiguated by SiteMinder from the username specified by the user.
lpszPassword	User password as specified by the user.
nBytesChapPassword	Length of the CHAP password.
lpszChapPassword	CHAP password.
nBytesChapChallenge	Length of the CHAP challenge.
lpszChapChallenge	CHAP challenge.
lpReserved1	Reserved for internal use.
lpszCertUserDN	User DN part of the X.509 user's Certificate.

Field	**Description**
lpszCertIssuerDN	Issuer DN part of the X.509 user’s Certificate.
nCertBinLen	Length of the user’s binary X.509 Certificate.
lpCertBinary	User’s binary X.509 Certificate.
lpszDirPath	Directory path in the SiteMinder notation of a SiteMinder user directory where user’s context was established.
lpszDirServer	Directory server in the SiteMinder user directory where user’s context was established.
lpszDirNamespace	Directory namespace in the SiteMinder user directory where user’s context was established.
lpszNewPassword	The user’s new password.
lpszPasswordToken	The password token from Password Services.

Chapter 7: Authentication API for C

This section contains the following topics:

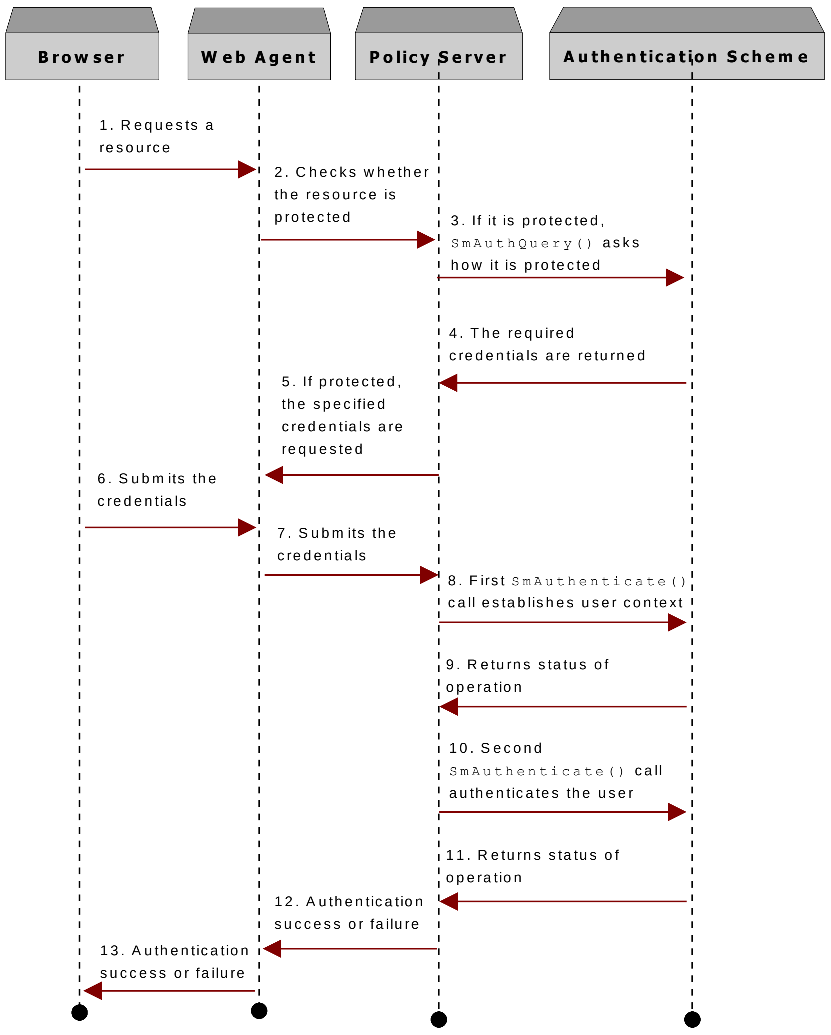
Authentication API Overview

Each SiteMinder Authentication Scheme is an instance of a shared library that supports the Authentication API provider interface.

Typically, when you define an authentication scheme in the Administrative UI, you accept the default library for the authentication scheme type you want to use. For example, if you want to use an authentication scheme based on the HTML Form Template, you would accept its default library, smauthhtml.

If you want your authentication scheme to support custom authentication functionality, build a new library for the authentication scheme. You build an authentication scheme library using the Authentication API.

The following figure shows how the authentication scheme library is used in the authentication process:


Browser
Web Agent
Policy Server
Authentication Scheme
1. Requests a resource
2. Checks whether the resource is protected
3. If it is protected, SmAuthQuery() asks how it is protected
4. The required credentials are returned
5. If protected, the specified credentials are requested
6. Submits the credentials
7. Submits the credentials
8. First SmAuthenticate() call establishes user context
9. Returns status of operation
10. Second SmAuthenticate() call authenticates the user
11. Returns status of operation
12. Authentication success or failure
13. Authentication success or failure

Install an Authentication Scheme Library

Install your custom-built library in one of the following default locations:

- On UNIX platforms, in the SiteMinder lib directory
- On Windows platforms, in the SiteMinder bin directory

Load an Authentication Scheme

The authentication scheme library is loaded by both the authentication server and the authorization server. Both servers use the scheme to retrieve the required credentials.

Immediately after the scheme is loaded, the following operations occur:

1. The SmAuthQuery() function is called to get the Authentication API version number and description.
2. After SmAuthQuery() is called, SmAuthInit() is called by the authentication server. The SmAuthInit() function is not called by the authorization server.

User Context

An authentication scheme verifies the user credentials passed to it by SiteMinder and returns the authentication result. An authentication scheme operates in these modes:

- **User context is unknown**—The user has yet to be located in the user store. Either SiteMinder or the authentication scheme looks up the user so that the user's stored credentials can be compared with the credentials supplied at login.

  Looking up the user in the user store is called *user disambiguation*.

- **User context is known**—The user has been located in the user store. The custom authentication scheme can now verify the user's credentials.

Authentication Events

Authentication results are tied to SiteMinder events. If authentication events are enabled in the realm where the user is being authenticated, SiteMinder evaluates optional policies tied to OnAuthAccept**,** OnAuthReject**,** OnAuthAttempt**,** and OnAuthChallenge rules. You can configure these policies to return custom responses based on a user's identity, redirect the user to another location based on the result of the authentication, or update the user data in an external database.

Redirection

The authentication scheme can tell the Policy Server to instruct the agent to perform a redirect. To build an authentication scheme that provides redirection capabilities, place the URL in the *lpszErrMsg* parameter and return a status code that includes reason code Sm_Api_Reason_ErrorMessageIsRedirect.

For example:

```
strcpy (lpszErrMsg, "https://12.12.1.1/display/user.cgi?dn=");
strcat (lpszErrMsg, lpUserContext->lpszUserName);
return SM_MAKEAUTH_STATUSVALUE (Sm_AuthApi_Accept,
                          Sm_Api_Reason_ErrorMessageIsRedirect);
```

This functionality is useful when customizing the workflow of a Web application using a standard Agent. However, configuring redirection is also useful when using custom agents.

Supported Credentials

The Authentication API supports user authentication based on the following types of credentials:

- Username/Password
- X.509 Certificate
- Custom user attributes

More Information:

SmAuthQuery() (see page 593)

Create a Custom Authentication Scheme Library

To create a custom authentication scheme library:

Include the SmApi.h file, as follows:

```
#include "SmApi.h"
```

Make the following functions externally visible:

Function	Description
SmAuthenticate() (see page 590)	Performs user authentications. The scheme authenticates user credentials and returns the result.
SmAuthInit() (see page 592)	Initializes the scheme. The scheme should initialize whatever resources it needs at this time.
SmAuthQuery() (see page 593)	Requests scheme information.
SmAuthRelease() (see page 597)	Releases the scheme only when SiteMinder is shutting down. The scheme should release its resources at this time.

Each entry point in the shared library must be defined according to specified syntax.

Note: If you are using Microsoft Visual Studio, export the function addresses to a modular definition file (.DEF) file. To export the function addresses, create a .DEF file, and in the export section of the .DEF file, list all of the authentication scheme functions, described in the previous table. Once you have created the .DEF file, add it to the Microsoft Visual Studio project.

Compile the code into a DLL or shared library. When you define the authentication scheme in the Administrative UI, you will specify this library name in the Authentication Scheme Properties dialog box.

After you have written and compiled a custom authentication scheme library, you define the authentication scheme that will use the custom library. You do so using the Administrative UI.

SmAuthenticate()

The SmAuthenticate() function authenticates user credentials.

Syntax

This function has the following format:

```
Sm_AuthApi_Status_t SM_EXTERN SmAuthenticate (
   const Sm_Api_Context_t*            lpApiContext,
   const Sm_Api_UserContext_t*        lpUserContext,
   const Sm_AuthApi_UserCredentials_t* lpUserCredentials,
   const Sm_Api_Reason_t              nChallengeReason,
   const char*                        lpszParam,
   const char*                        lpszSharedSecret,
   const int                          nBytesUserMsg,
   char*                              lpszUserMsg,
   const int                          nBytesErrMsg,
   char*                              lpszErrMsg
);
```

Parameters

This function has the following parameters:

lpApiContext

[in] Indicates a pointer to the API context structure.

lpUserContext

[in] Indicates a pointer to the user context structure.

lpUserCredentials

[in] Indicates a pointer to the user credentials context structure.

nChallengeReason

[in] Indicates the reason for the original challenge; otherwise, set to zero.

lpszParam

[in] Indicates a pointer to the buffer containing the null-terminated parameter string as specified for the authentication scheme.

lpszSharedSecret

[in] Indicates a pointer to the buffer containing the null-terminated shared secret string as specified for the authentication scheme.

nBytesUserMsg

[in] Indicates the maximum size of the lpszUserMsg buffer to receive the user message—4097 bytes, including the string termination character.

lpszUserMsg

[out] Indicates a pointer to an output buffer to receive the user message. This message can be the challenge text or any other message that the scheme developer wants to present to the user through a mechanism external to SiteMinder. The Agent stores this message in the HTTP_SM_USERMSG HTTP variable. For RADIUS authentication, the user message is returned in the REPLY-MESSAGE response attribute.

nBytesErrMsg

[in] Indicates the maximum size of the lpszErrMsg error buffer—4097 bytes, including the string termination character.

lpszErrMsg

[out] Indicates a pointer to an output buffer to receive the error text. Use this buffer to return an error message to SiteMinder.

Returns

This function uses the SM_MAKEAUTH_STATUSVALUE macro to construct the return value. This macro is defined in SmApi.h. The syntax of the macro is as follows:

```
SM_MAKEAUTH_STATUSVALUE(status, reason)
```

This macro ha sthe following two parameters::

- *status*—An Sm_AuthApi_Status_t enumeration. Different status codes can be returned when SmAuthenticate() is called during disambiguation and when it is called during authentication, as follows:

  During the disambiguation phase:

  - Sm_AuthApi_NoUserContext
  - Sm_AuthApi_Success
  - Sm_AuthApi_SuccessUserDN
  - Sm_AuthApi_SuccessUserFilter
  - Sm_AuthApi_Attempt
  - Sm_AuthApi_Failure

  During the authentication phase:

  - Sm_AuthApi_Accept
  - Sm_AuthApi_Reject
  - Sm_AuthApi_Challenge
  - Sm_AuthApi_Failure

- *reason*—An Sm_Api_Reason_t enumeration:
  - Values 0 - 31999 are reserved for use by SiteMinder.
  - Values 32000 - 32767 are available for user-defined reasons.

SmAuthInit()

The SmAuthInit() function lets the authentication scheme perform its own initialization procedure. This call is made once, when the scheme is first loaded.

Syntax

This function has the following format:

```
Sm_AuthApi_Status_t SM_EXTERN SmAuthInit (
   const char*   lpszParam,
   const char*   lpszSharedSecret
);
```

Parameters

This function has the following parameters:

lpszParam

[in] Indicates a pointer to the buffer containing the null-terminated parameter string as specified for the authentication scheme.

lpszSharedSecret

[in] Indicates a pointer to the buffer containing the null-terminated shared secret string as specified for the authentication scheme.

Returns

This function returns one of the following values:

Sm_AuthApi_Success.

Indicates the function completed successfully.

Sm_AuthApi_Failure.

Indicates the function was unsuccessful. The scheme is not loaded.

SmAuthQuery()

The SmAuthQuery() function returns information about an authentication scheme.

Syntax

This function has the following format:

```
Sm_AuthApi_Status_t SM_EXTERN SmAuthQuery (
   const char*                  lpszParam,
   const char*                  lpszSharedSecret,
   const Sm_AuthApi_QueryCode_t  code,
   char*                        lpszStatusBuffer,
   int*                         lpnStatusCode
);
```

Parameters

This function has the following parameters:

lpszParam

[in] Indicates a pointer to the buffer containing the null-terminated parameter string as specified for the authentication scheme.

lpszSharedSecret

[in] Indicates a pointer to the buffer containing the null-terminated shared secret string as specified for the authentication scheme.

code

[in] Request code as defined by an enum type Sm_AuthApi_QueryCode_t, which contains the following values**:**

- Sm_AuthApi_QueryDescription. Requests the scheme's description.
  The scheme returns a description through lpszStatusBuffer and the Authentication API version number through *lpnStatusCode*.
- Sm_AuthApi_QueryCredentialsReq.
  Requests the credentials required by the scheme and where to obtain them. The scheme should return a bit mask through lpnStatusCode, which represents the credentials needed for authentication.

Individual flags are defined by the enum type Sm_Api_Credentials_t. The scheme may return the location where the credentials can be obtained (that is, an HTTP URL) using the *lpszStatusBuffer* parameter.

lpszStatusBuffer

Receives a character string response. Up to Sm_AuthApi_StatusBufSize characters can be returned.

lpnStatusCode

If the call requests the scheme's description, receives a numeric response indicating the version number of the Authentication API. Supported versions are Sm_Api_Version_V4 and Sm_Api_Version_V4_1.

If the call requests the credentials required by the scheme, the scheme returns one or more credentials. See Remarks. These constants are defined in SmApi.h.

Returns

This function returns one of the following values:

Sm_AuthApi_Success

Indicates the function completed successfully.

Sm_AuthApi_Failure

Indicates the caller has specified an invalid request code.

Remarks

The scheme supports request codes as specified by enumeration type Sm_AuthApi_QueryCode_t. The caller queries the scheme to find out what kind of credentials are required for authentication and where to obtain them.

The credentials are defined by the enum type Sm_Api_Credentials_t and can be combined to request multiple credentials. The caller collects the requested credentials, places them into the Sm_AuthApi_UserCredentials_t context structure, and calls the SmAuthenticate() function.

The individual flags in Sm_Api_Credentials_t are as follows:

- Sm_AuthApi_Cred_None—No credentials are required.
- Sm_AuthApi_Cred_Basic—Username and password are required.
- Sm_AuthApi_Cred_Digest—Required user name and password are exchanged using the digest protocol.
- Sm_AuthApi_Cred_X509Cert—Full X.509 Client Certificate is required. Sm_AuthApi_Cred_SSLRequired must be specified.
- Sm_AuthApi_Cred_X509CertUserDN—User DN from an X.509 Client Certificate is required. Sm_AuthApi_Cred_SSLRequired must be specified.
- Sm_AuthApi_Cred_X509CertIssuerDN—Issuer DN from an X.509 Client Certificate is required. Sm_AuthApi_Cred_SSLRequired must be specified.
- Sm_AuthApi_Cred_CertOrBasic—Either a certificate is required or a user name and password is required.
- Sm_AuthApi_Cred_CertOrForm—Either a certificate is required or a form-based username and password are required.

- Sm_AuthApi_Cred_SSLRequired—An SSL connection is required. A redirect to an https URL must be specified.
- Sm_AuthApi_Cred_NTChalResp—Required user name and password are exchanged using the NT Challenge Response protocol.
- Sm_AuthApi_Cred_AllowSaveCreds—Signifies whether the credentials can be saved for 30 days by the user. If a user saves their credentials, they do not need to enter their credentials, such as user name and password, each time they access the protected resource.
- Sm_AuthApi_Cred_FormRequired—A form-based username and password are required. A redirect to a URL containing a form must be specified.
- Sm_AuthApi_Cred_PreserveSessionId—The session ID should be preserved if the current session is still valid.
- Sm_AuthApi_Cred_DoNotChallenge—Do not challenge for credentials
- Sm_AuthAPI_Cred_AllowAnonymous—Allow validation of anonymous identity.

Examples: Setting status codes

- An Anonymous scheme combines these credential flags:

  ```
  *lpnStatusCode = Sm_AuthApi_Cred_None|Sm_AuthApi_Cred_AllowAnonymous;
  ```

- Collect username and password. The Agent uses HTTP basic authentication to challenge the user:

  ```
  *lpnStatusCode = Sm_AuthApi_Cred_Basic;
  ```

- Collect username and password. The Agent challenges the user over SSL using HTTP basic authentication:

  ```
  *lpnStatusCode = Sm_AuthApi_Cred_Basic |
                             Sm_AuthApi_Cred_SSLRequired;
  strcpy(lpszStatusBuffer,
                  "https://xxx.yyy.zzz/secure.ssc?basic")
  ```

- Collect username and password. The Agent challenges the user over SSL using an HTML form:

  ```
  *lpnStatusCode = Sm_AuthApi_Cred_Basic |
                            Sm_AuthApi_Cred_FormRequired;
  strcpy(lpszStatusBuffer,
                 "https://xxx.yyy.zzz/getcredentials.fcc")
  ```

- Collect an X.509 Certificate. The Agent challenges the user for a certificate:

  ```
  *lpnStatusCode = Sm_AuthApi_Cred_SSLRequired |
                                       Sm_AuthApi_Cred_X509Cert;
  strcpy(lpszStatusBuffer,
                       "https://xxx.yyy.zzz/getcert.scc?cert")
  ```

- Collect an X.509 Certificate, username, and password. The Agent challenges the user over SSL using HTTP basic authentication:

  ```
  *lpnStatusCode = Sm_AuthApi_Cred_Basic |
                                Sm_AuthApi_Cred_SSLRequired |
                                Sm_AuthApi_Cred_X509Cert;
  strcpy(lpszStatusBuffer,
                "https://xxx.yyy.zzz/getcert.scc?cert+basic")
  ```

SmAuthRelease()

The SmAuthRelease() function lets an authentication scheme perform its own rundown procedure.The caller makes this call once when SiteMinder is shutting down.

Syntax

This function has the following format:

```
Sm_AuthApi_Status_t SM_EXTERN SmAuthRelease (
   const char*    lpszParam
   const char*    lpszSharedSecret
);
```

Parameters

This function has the following parameters:

lpszParam

[in] Indicates a pointer to the buffer containing the null-terminated parameter string as specified for the authentication scheme.

lpszSharedSecret

[in] Indicates a pointer to the buffer containing the null-terminated shared secret string as specified for the authentication scheme.

Returns

This function returns one of the following values:

Sm_AuthApi_Success.

Indicates the function completed successfully.

Sm_AuthApi_Failure.

Indicates the function was unsuccessful. The scheme is not loaded.

Chapter 8: Authorization API for C

This section contains the following topics:

Authorization API Overview

Using the Authorization API, you can implement custom access control functionality. To implement custom access control functionality, you must:

1. Develop a shared library that supports the Authorization API and provides the custom functionality you need.

   The shared library must contain one or more functions defined as exportable symbols. SmApi.h defines all of the data structures necessary to create custom policy, rule, and response plug-ins.

2. Install the shared library in one of the following default locations:

   - On UNIX platforms, in the SiteMinder lib directory
   - On Windows platforms, in the SiteMinder bin directory

3. Define one or more of the following in the Administrative UI:

   - Active policy—A policy that provides dynamic authorization based on external business logic.

     For example, you might define an active policy that returns true if the user belongs to a particular organizational unit (ou) in an LDAP directory as defined in the parameter *(param*) field of the active policy expression.

   - Active response—A custom response returned from a shared library. Using an active response is one way you can define user-specific privilege information.

     For example, you might define an active response that returns a user's common name (cn) if the user belongs to the ou specified in the *param* field of the active response expression.

   - Active rule—A rule that provides dynamic authorization based on external business logic.

     For example, you might define an active rule that returns true if a user is a member of a group, such as Directory Administrator, that has permission to view a realm.

Include File

When extending the authorization API, include the SmApi.h header file:

```
#include "SmApi.h"
```

Active Expressions

An *active expression* is a string of variable definitions that comprises an active policy, rule, or response. Active expressions are constructed in the Administrative UI using the following syntax:

```
<@ lib=<lib-spec> func=<func-spec> param=<func-params>@>
```

In the syntax example:

- *lib-spec* is the path to a custom shared library. This clause is required.

  If you place the library in the default location, you need only specify the library file name rather than a path. Also, the extension .dll or .so is optional.

- *func-spec* is the name of a user-defined, externally-visible function defined in the shared library. This clause is required.
- *func-params* is a parameter string to be passed to the function. This clause is optional.

SiteMinder constructs the active expression from information provided in the Active Rule Editor, Active Policy Editor, or Active Response Attribute Editor dialog box.

How SiteMinder Interprets Active Expressions

An active expression in an application initiates the following tasks:

- Loads the shared library specified in the active expression.
- Calls the user-defined function specified in the active expression.
- Passes to the user-defined function the optional parameter string plus contextual information—that is, API context (Sm_Api_Context_t), request context (Sm_Api_RequestContext_t), and user context (Sm_Api_UserContext_t).

The specified user-defined function in the shared library returns a result to SiteMinder in the *lpszOutBuf* parameter. SiteMinder interprets this result according to the type of active expression, as follows:

- Active Policy—If the function call fails or the result returned in *lpszOutBuf* is empty, authorization is denied.

  The policy does not fire if the result returned in *lpszOutBuf* matches any of the following strings (not case-sensitive): FALSE, F, or 0.

  Any other result value causes the policy to fire.

- Active Rule—If the function call fails or the result returned in *lpszOutBuf* is empty, the following behavior occurs:

  - With Allow Access rules, the rule does not fire.

  - With Deny Access rules, the rule fires.

  Otherwise, the behavior is the same as for Active Policies.

- Active Response—The result is a string representing the response attribute value. How SiteMinder uses this value is determined by the response attribute specified in the Administrative UI. For example:

  - WebAgent-OnReject-Redirect. Given this attribute, SiteMinder expects the response value to specify a location, such as a URL, to redirect a user who is denied access to a resource.

    For example, you could specify a group name in the optional *param* variable of the active expression, then test for the group name in the function to determine the URL to pass back.

  - WebAgent-HTTP-Cookie-Variable. Given this attribute, SiteMinder expects that the response value, such as the user's common name, is to be assigned to a cookie variable. You can use the response value any way you like, such as displaying the user's common name to personalize a form.

    You specify the cookie name in the SiteMinder Response Attribute Editor.

Define Active Rules

Active rules are defined in the Administrative UI using the Active Rule Editor dialog box. To access this editor from the Rule Properties dialog box, select the Active Rule tab in the Advanced group box, then click Edit.

Define Active Responses

Active responses are defined in the Administrative UI using the Response Attribute Editor dialog box.

From the Response Properties dialog box, access the editor by clicking Create and select the Active Response button in the Attribute Kind group box on the Attribute Setup tab.

Define Active Policies

Active policies are defined in the Administrative UI using the Active Policy Editor dialog box.

From the Policies Properties dialog box, access this editor by selecting the Advanced tab and clicking Edit.

Pass HTTP Headers and Cookies to Policy Server

You can add arbitrary custom key/value pairs to the current session to pass HTTP headers and cookies to the Policy Server. These key/value pairs are kept in the session store; they have the same lifetime as the session. The name/value pairs are stored in the Expiry Data Table in the session store database. There can be maximum 5 entries for a session. The Active Plugin code can use the UserContext structure (Sm_Api_UserContext_t* lpUserContext) to set and retrieve these name/value pairs by calling fSetProp and fGetProp respectively. To set the value, fSetProp is called with lpszPropName as SM_SESSIONVAR(<name>) and lpszValueBuf as the value. To retrieve the value, fGetProp is called with lpszPropName as SM_SESSIONVAR(<name>)/ SM_SESSIONVAR and the value/values is returned in lpszValueBuf.

Note the following:

- To set the name/value pair in Expiry Data Table, the name should be passed in SM_SESSIONVAR(<name>) format only. The name can be maximum 32 characters long, can start with only a letter or underscore and can contain only letters, digits or underscore. The length of the value which can be set is 4000 characters maximum.
- To fetch the value for a name, the name should be passed in SM_SESSIONVAR(<name>) format. The value in Expiry Data Table corresponding to <name> is returned. If only SM_SESSIONVAR is passed (no name is passed within), then all the names for that session are returned in caret-delimited format.
- If you remove name/value pair in Expiry Data Table, the name is passed as SM_SESSIONVAR(<name>) and the value should be passed as blank.

Authorization Function Declarations

The shared library requires proper entry points. Each entry point in the shared library represents one or more active expressions and must be defined according to the specified syntax.

Note: If you are using Microsoft Visual Studio, export the function addresses to a modular definition file (.DEF) file. To export the function addresses, create a .DEF file, and in the export section of the .DEF file, list the functions that you want to invoke from the Active Rule or Active Policy. After you have created the .DEF file, add it to the Microsoft Visual Studio project.

User-Defined Function

The Policy Server calls a user-defined function to perform a custom policy, rule, or response operation.

You can assign the function any name. Through the active expression that you define in the Administrative UI, you advise SiteMinder of the function name and the name of the shared library where the function resides.

Syntax

```
int SM_EXTERN <func-spec> (
   const Sm_Api_Context_t*          lpApiContext,
   const Sm_Api_UserContext_t*      lpUserContext,
   const Sm_Api_RequestContext_t*   lpReqContext,
   const char*                      lpszParam,
   const int                        nBytesOutBuf,
   char*                            lpszOutBuf,
   const int                        nBytesErrBuf,
   char*                            lpszErrBuf
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
lpUserContext	I	Pointer to the user context structure.
lpReqContext	I	Pointer to the request context structure.
lpszParam	I	Pointer to the buffer containing the null-terminated parameter string specified in *<Param-String>*.
nBytesOutBuf	I	Maximum size of the output result buffer (4097 bytes).

Parameter	I/O	Description
lpszOutBuf	O	Output buffer for the result to send to SiteMinder.
nBytesErrBuf	I	Maximum size of the output error buffer (4097 bytes).
lpszErrBuf	O	Output buffer to receive error text. SiteMinder displays the error text in the debug log file.

Returns

- Upon successful execution, the function returns a value greater than 0—that is, the total number of bytes in the result buffer *lpszOutBuf*.
- If an error occurs, the function returns -1 and stores the applicable error message in the error buffer *lpszErrBuf*.
- When used for an Active Rule, a return value of 0 (an empty string was passed back in *lpszOutbuf*) results in one of the following actions:
  - With Allow Access rules, the rule does not fire.
  - With Deny Access rules, the rule fires.

SmQueryVersion()

The Policy Server calls this function to determine the Authorization API version that the custom library is compliant with.

Syntax

```
int SmQueryVersion (
   const Sm_Api_Context_t* lpApiContext
);
```

Parameter	I/O	Description
lpApiContext	I	A pointer to the API context structure.

Returns

Returns the version number of the Authorization API. Currently the version supported is Sm_Api_Version_V3. This constant is defined in SmApi.h.

Active Expression Examples

This samples provided in the following sections are located in:

```
sdk\samples\smazapi\smazapi.cpp
```

The syntax that precedes each sample, such as:

```
<@ lib="SmAzAPI" func="activeRule" param="" @>
```

is an example of the Generated Script that SiteMinder constructs in the SiteMinder Active Rule Editor, Active Policy Editor, or Active Response Attribute Editor dialog box from the information you provide in the dialog box.

To build sample active expressions for UNIX, use the makefile found in <install_path>\sdk\samples\smazapi\makefile.

Example of an Active Rule

The example below returns true if the user has special access permission to view the realm. If the user has directory manager privileges, the user can view the realm.

```
<@ lib="SmAzAPI" func="activeRule" param="" @>
*************************************************************
int SM_EXTERN activeRule(
const Sm_Api_Context_t* lpApiContext,
// the structure that provides API context
const Sm_Api_UserContext_t* lpUserContext,
// the structure that provides user context
const Sm_Api_RequestContext_t* lpReqContext,
// the structure that provides request context
const char* lpszParam,
// the parameter string (null-terminated)
const int nBytesOutBuf,
// the maximum size of the output buffer
char* lpszOutBuf,
// the output buffer to hold the null-terminated result
const int nBytesErrBuf,
// the maximum size of the error message buffer
char* lpszErrBuf)
// the output buffer to hold the null-terminated error message
{
/* User Context is required to use the functions like fGetProp, fSetProp.. */
if(!lpUserContext->bIsUserContext)
   {
   strncpy (lpszErrBuf, "No User Context ", nBytesErrBuf);
   lpszErrBuf[nBytesErrBuf-1] = '\0';
   return -1;
   }
/*
// The DN to look for the attribute "uniquemember"
// If the user is listed as the member of the above attribute,
// it has directory manager privileges.
*/
char lpszDn[] = "cn=Directory Administrators,ou=Groups,o=airius.com";
char lpszDnvalue[256];
memset(lpszDnvalue, 0, sizeof(lpszDnvalue));
/*
// fGetDnProp function is used to retrieve an attribute value
// in a directory entry.
*/
int getResult = lpUserContext->fGetDnProp(
   lpUserContext->lpParam,
   lpszDn,
   "uniquemember",
   sizeof(lpszDnvalue),
   lpszDnvalue);
/*
// If no error occurs, fGenDnProp will return the length of the
// buffer lpszDnvalue. Otherwise the function returns 0.
*/
if(getResult > 0)
   {

   /* Check to see if the user is present in the list. */
   if(strpbrk(lpszDnvalue, lpUserContext->lpszUserName) != NULL)
      {

      /* The result "true" is placed in the output buffer. */
      strncpy(lpszOutBuf, "true", nBytesOutBuf);
      lpszOutBuf[nBytesOutBuf-1] = '\0';
      return strlen(lpszOutBuf);
      }

      else

      {
      strncpy(lpszOutBuf, "false", nBytesOutBuf);
      lpszOutBuf[nBytesOutBuf-1] = '\0';
      return strlen(lpszOutBuf);
      }
   }

   else

   {
   strncpy(lpszErrBuf, "Failed to get attribute value for the DN ",
                        nBytesErrBuf);
   strncat( (lpszErrBuf + strlen(lpszErrBuf)), lpszDn,
             (nBytesErrBuf-strlen(lpszErrBuf)));
   lpszErrBuf[nBytesErrBuf-1] = '\0';
   return -1;
   }

/* everything failed.... */

return 0;

}
```

Example of an Active Response

This active response returns the common name (cn) of the user, if the user belongs to the organizational unit specified in the parameter (param) field of the active response expression.

```
<@ lib="SmAzAPI" func="activeResponse" param="Human Resources" @>
************************************************************
int SM_EXTERN activeResponse(
const Sm_Api_Context_t*   lpApiContext,
/* the structure that provides API context */
const Sm_Api_UserContext_t*  lpUserContext,
/* the structure that provides user context */
const Sm_Api_RequestContext_t*  lpReqContext,
/* the structure that provides request context */
const char* lpszParam,
/* the parameter string (null-terminated) */
const int  nBytesOutBuf,
/* the maximum size of the output buffer */
char*  lpszOutBuf,
/* the output buffer to hold the null-terminated attribute value */
const int   nBytesErrBuf,
/* the maximum size of the error message buffer */
char*   lpszErrBuf)
/* the output buffer to hold the null-terminated error message */
{
memset(lpszOutBuf, 0, sizeof(lpszOutBuf));
if(!lpUserContext->bIsUserContext)
      {
      strncpy (lpszErrBuf, "No User Context ", nBytesErrBuf);
      lpszErrBuf[nBytesErrBuf-1] = '\0';
      return -1;
      }
/* Store all the organizational units to which the user belongs. */
char lpszOrgUnit[30];
memset(lpszOrgUnit, 0, sizeof(lpszOrgUnit));
/* store the common name of the user. */
char lpszCN[30];
memset(lpszCN, 0, sizeof(lpszCN));
/* Check to see if a parameter is requested. */
if(lpszParam == NULL || strlen(lpszParam) == 0)
   {
   strncpy (lpszErrBuf, "Organizational unit is not entered ",
            nBytesErrBuf);
   lpszErrBuf[nBytesErrBuf-1] = '\0';
   return -1;
   }
/* Get all the organization units to which the user belongs. */
```

```
int getResult = lpUserContext->fGetProp (
   lpUserContext->lpParam,
   "ou",              /* Attribute name */
   sizeof (lpszOrgUnit), lpszOrgUnit);
if (getResult < 0)
   {
   strncpy (lpszErrBuf,
        "Failed to get organization unit for the user's profile ",
        nBytesErrBuf);
   strncat( (lpszErrBuf + strlen(lpszErrBuf)),
            lpUserContext->lpszUserName,
            (nBytesErrBuf-strlen(lpszErrBuf)));
   lpszErrBuf[nBytesErrBuf-1] = '\0';
   return -1;
   }
    else
    {
/* Check if the user belongs to the organization unit that is requested. */
   if(strstr(lpszOrgUnit, lpszParam) != NULL)
      {
      if((lpUserContext->fGetProp(lpUserContext->lpParam,
           "cn",sizeof(lpszCN),lpszCN)) > 0)
         {
         strncpy(lpszOutBuf, lpszCN, nBytesOutBuf);
         lpszOutBuf[nBytesOutBuf-1] = '\0';
         return strlen(lpszOutBuf);
         } /* end of fGetProp */
         else
         {
         strncpy (lpszErrBuf,
          "Failed to get user common name from user's profile attribute ",
           nBytesErrBuf);
         strncat( (lpszErrBuf + strlen(lpszErrBuf)),
                    lpUserContext->lpszUserName,
                    (nBytesErrBuf-strlen(lpszErrBuf)));
         lpszErrBuf[nBytesErrBuf-1] = '\0';
         return -1;
         }
        } /* end of strstr */
        else
        {
      strncpy (lpszErrBuf,
       "The user does not belong to the requested organizational unit ",
        nBytesErrBuf);
      lpszErrBuf[nBytesErrBuf-1] = '\0';
      return -1;
      }
   }
```

```
    /* everything failed.... */
    return 0;
 }
#ifndef _WIN32
}

#endif
```

Chapter 9: Tunnel Service API Guidance

This section contains the following topics:

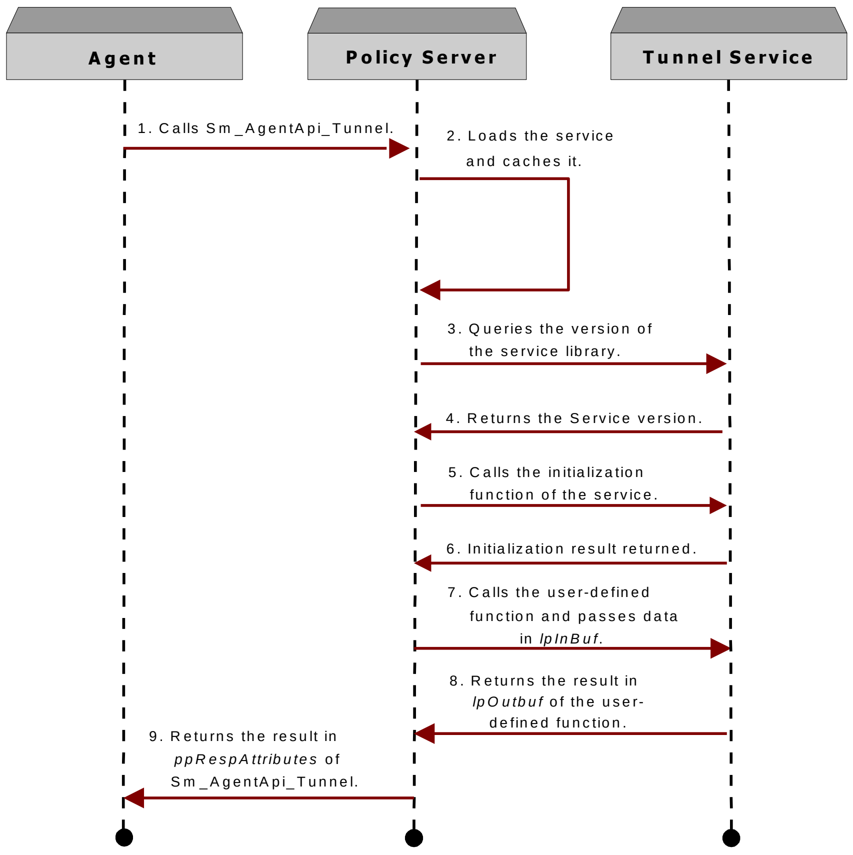
Tunnel Service API Overview

The Tunnel Service API provides secure transfer of data between an agent and a shared library that supports the Tunnel Service.

When an agent sends a tunnel request to the Policy Server, the request contains:

- The name of the service library
- The function to be called in the service library
- The data to be passed to the function

The Policy Server initializes the appropriate service, invokes the requested function, and passes the data to the function. Once the service has performed its task, the Policy Server returns the results to the agent. The following graphic shows the tunnel service process:


Agent
Policy Server
Tunnel Service
1. Calls Sm_AgentApi_Tunnel.
2. Loads the service and caches it.
3. Queries the version of the service library.
4. Returns the Service version.
5. Calls the initialization function of the service.
6. Initialization result returned.
7. Calls the user-defined function and passes data in *lpInBuf*.
8. Returns the result in *lpOutbuf* of the user-defined function.
9. Returns the result in *ppRespAttributes* of Sm_AgentApi_Tunnel.

Develop a Custom Tunnel Service

Each tunnel service is an instance of a shared library that supports the Tunnel Service API. To support a tunnel service, you must build a new shared library.

Install the shared library in one of the following locations:

- On UNIX platforms, in the SiteMinder lib directory
- On Windows platforms, in the SiteMinder bin directory

Include File

To develop a tunnel service, include the SmApi.h header file:

```
#include "SmApi.h"
```

Tunnel Service API Reference

The shared library must provide the following functions as externally visible entry points:

Function	Description
SmQueryVersion()	Requests the Tunnel Service API version that the custom library is compliant with.
SmTunnelInit()	Initializes the tunnel service.
SmTunnelRelease()	Releases the tunnel service.
User-Defined Function	Calls the function that the tunnel agent is requesting.

Each entry point in the shared library must be defined according to specified syntax.

Note: If you are using Microsoft Visual Studio, export the function addresses to a modular definition file (.DEF) file. To export the function addresses, create a .DEF file, and in the export section of the .DEF file, list all of the tunnel service functions, described in the previous table. Once you have created the .DEF file, add it to the Microsoft Visual Studio project.

SmQueryVersion()

SiteMinder calls this function to request the Tunnel Service API version that the custom library is compliant with.

Syntax

```
int SM_EXTERN SmQueryVersion (
   const Sm_Api_Context_t* lpApiContext
);
```

Parameter	I/O	Description
lpApiContext	I	A pointer to the API context structure.

Returns

Returns the version number of the Tunnel Service API. Currently the versions supported are Sm_Api_Version_V4 and Sm_Api_Version_V4_1. Version constants are defined in SmApi.h.

SmTunnelInit()

SiteMinder calls this function so that a tunnel service can perform its own initialization procedure. This call is made once when the tunnel service is loaded for the first time. The information is cached for subsequent use.

Syntax

```
int SM_EXTERN SmTunnelInit (
   void**                     ppServiceHandle,
   const Sm_Api_Context_t*    lpApiContext,
   const int                  nBytesStatusBuf,
   char*                      lpszStatusBuf
);
```

Parameter	I/O	Description
ppServiceHandle	I	An internal pointer used by the tunnel service library.
lpApiContext	I	A pointer to the API context structure.
nBytesStatusBuf	O	Maximum size of the output status buffer.
lpszStatusBuf	O	Output buffer receives any status messages from the tunnel service.

Returns

Returns 0 if successful or -1 if unsuccessful.

SmTunnelRelease()

The tunnel service can perform its own rundown procedure. This call is made one time when SiteMinder is shutting down.

Syntax

```
void SM_EXTERN SmTunnelRelease (
   void*                    pServiceHandle,
   const Sm_Api_Context_t*   lpApiContext
);
```

Parameter	I/O	Description
pServiceHandle	I	An internal pointer used by the tunnel service library.
lpApiContext	I	A pointer to the API context structure.

User-Defined Function

The tunnel agent requests this function:

Syntax

```
int SM_EXTERN <func-spec> (
   void*                           pServiceHandle,
   const Sm_Api_Context_t*         lpApiContext,
   const Sm_Api_RequestContext_t*  lpReqContext,
   const Sm_Api_TunnelContext_t*   lpTunnelContext,
   const int                       nBytesInBuf,
   void*                           lpInBuf,
   const int                       nBytesOutBuf,
   void*                           lpOutBuf,
   const int                       nBytesStatusBuf,
   char*                           lpszStatusBuf
);
```

Parameter	I/O	Description
pServiceHandle	I	An internal pointer used by the tunnel service library.
lpApiContext	I	A pointer to the API context structure.
lpReqContext	I	A pointer to the API request structure.

Parameter	I/O	Description
lpTunnelContext	I	A pointer to the API tunnel context.
nBytesInBuf	I	Number of bytes in the input buffer.
lpInBuf	I	Input buffer containing information sent from the remote agent.
nBytesOutBuf	O	Maximum size of the output result buffer.
lpOutBuf	O	Output buffer to receive the result.
nBytesStatusBuf	O	Maximum size of the status buffer.
lpszStatusBuf	O	Status buffer to receive status.

Chapter 10: Event API Guidance

This section contains the following topics:

Event API Overview

The SiteMinder Event API lets you create custom event handlers.

Through the Event API, SiteMinder can log events using outside sources, providers, or applications. You can then access the logged information through these other sources, providers, or applications.

Using the Event API, you can build applications to alert administrators of SiteMinder activity. For example, an event handler can send an e-mail to the administrator when the accounting server starts or someone creates a new policy.

Event API Setup

Each event handler is an instance of a shared library that supports the Event API provider interface. To support custom event handlers, you must build a shared library.

Install the shared library in one of the following locations:

- On UNIX platforms, in the SiteMinder lib directory
- On Windows platforms, in the SiteMinder bin directory

The shared library must export the following entry points:

- SmEventInit()
- SmEventRecord()
- SmEventRelease()

To build an event handler, include the SmEventApi.h header file:

```
#include "SmEventApi.h"
```

Event Provider Structures

The following table lists the structure definitions used with Event Provider API functions:

Structure	Description
SmLog_Access_t (see page 619)	Contains information about access events.
SmLog_EMS_t (see page 621)	Contains information about Entitlement Management Services (EMS) events. EMS events result from actions performed on directory objects.
SmLog_Obj_t (see page 622)	Contains information about object events.
SmLog_System_t (see page 623)	Contains information about system events.

SmLog_Access_t

Contains information about an access event.

Syntax

```
typedef struct SmLog_Access_s
{
   long nVersion;
   long nCurrentTime;
   Sm_Api_Reason_t nReason;
   char* szAgentName;
   char* szSessionId;
   char* szClientIp;
   char* szUserName;
   char* szDomainOid;
   char* szRealmName;
   char* szRealmOid;
   char* szAuthDirName;
   char* szAuthDirServer;
   char* szAuthDirNamespace;
   char* szServer;
   char* szResource;
   char* szAction;
   char* szTransactionId;
   char* szStatusMsg;
   char* szDomainName;
   char* szImpersonatorName;
   char* szImpersonatorDirName;
} SmLog_Access_t;
```

Field	**Description**
nVersion	Version number of the Policy Server.
nCurrentTime	Policy Server time (in GMT) when the event occurred.
nReason	Contains the reason identifier.
szAgentName	Name of the agent.
szSessionId	Session ID.
szClientIp	The IP address of the client.
szUserName	Full distinguished name of the user or administrator.
szDomainOid	The object identifier of the domain object.

Field	Description
szRealmName	Name of the realm in which the resource is protected.
szRealmOid	Object identifier for the realm object.
szAuthDirName	The name of the directory.
szAuthDirServer	Directory Server of a SiteMinder user directory where user's context was established.
szAuthDirNamespace	Directory Namespace such as LDAP, WinNT, or ODBC.
szServer	Name of the server holding the resource.
szResource	Name of the resource.
szAction	Type of action performed on the resource. A typical action when accessing web resources is a GET.
szTransactionId	Identifier of a transaction between the agent and Policy Server. The agent sets this ID and the Policy Server logs it. When the agent makes a request to the Policy Server, it associates an ID with the request. Since the agent may make many such requests, the agent uses the ID to match information from the Policy Server with the originating request.
szStatusMsg	Status message.
szImpersonatorName	Specifies the name of the impersonator. Set to NULL if there is no impersonated session in progress.
szDomainName	Specifies the name of the domain.
szImpersonator DirName	Specifies the name of the user directory used to authenticate the impersonator. Set to NULL if there is no impersonated session in progress.

More Information:

Sm_Api_Reason_t (see page 755)

SmLog_EMS_t

Contains information about an Entitlement Management Services (EMS) event. EMS events result from actions performed on directory objects.

Syntax

```
typedef struct SmLog_EMS_s
{
   long nVersion;
   long nCurrentTime;
   char* szUserName;
   char* szSessionId;
   char* szDirName;
   char* szObjName;
   char* szObjPath;
   char* szObjClass;
   char* szOrgName:
   char* szRoleName;
   int   szFieldDesc;
   char* szStatusMsg;
} SmLog_EMS_t;
```

Field	Description
nVersion	Version number of the Policy Server.
nCurrentTime	Policy Server time (in GMT) when the event occurred.
szUserName	If the user is an administrator, the ID of the administrator who initiated the EMS event. If the event is an end-user event, the user name is Registration.
szSessionID	EMS Service Session ID.
szDirName	Name of the SiteMinder directory affected by the EMS event.
szObjName	Name of the object targeted by this event.
szObjPath	Full distinguished name of the object.
szObjClass	Class name of the object.
szOrgName	Name of the object’s organization.
szRoleName	Name of the role to which the object is related (only for events that involve roles).
szFieldDesc	Description of the event.

Field	Description
szStatusMsg	The status message.

SmLog_Obj_t

Contains information about an object event.

Syntax

```
typedef struct SmLog_Obj_s
{
   long nVersion;
   long nCurrentTime;
   char* szUserName;
   char* szSessionId;
   char* szDomainOid;
   char* szObjName;
   char* szObjOid;
   char* szFieldDesc;
   char* szStatusMsg;
} SmLog_Obj_t;
```

Field	Description
nVersion	Version number of the Policy Server.
nCurrentTime	Policy Server time (in GMT) when the event occurred.
szUserName	The name of the user who triggered the event.
szSessionId	Session ID.
szDomainOid	Object identifier for the domain object.
szObjName	The name of the object.
szObjOid	Object identifier for the object.
szFieldDesc	User DN.
szStatusMsg	Status message.

SmLog_System_t

Contains information about a system event.

Syntax

```
typedef struct SmLog_System_s
{
   long nVersion;
   long nCurrentTime;
   char* szName;
   char* szIpAddress;
   int   nIpPort;
   char* szMsg;
} SmLog_System_t;
```

Field	Description
nVersion	Version number of the Policy Server.
nCurrentTime	Time when the event occurred.
szName	Name of the client, database, or directory.
szIpAddress	IP address of the client.
nIpPort	Port number that the client communicates on.
szMsg	Buffer to receive string message for an event.

Event API Reference

SiteMinder logs four types of events. All event classifications have well-defined data. The types of events are:

- Access events (associated with structure SmLog_Access_t).
- EMS events (associated with structure SmLog_EMS_t).
- Object events (associated with structure SmLog_Obj_t).
- System events (associated with structure SmLog_System_t).

Access Event Type

Access events result from user-related activities. They are called in the context of authentication, authorization, administration, and affiliate activity.

There are four categories of access events. Each of these event categories responds with its own set of events. The following table lists the access event categories and their associated response events.

Event Category	SiteMinder Activity	SiteMinder Event
Authentication	■ User authentication accepted ■ User authentication rejected ■ User authentication attempted ■ User authentication challenged ■ User session validated	SmLogAccessEvent_AuthAccept SmLogAccessEvent_AuthReject SmLogAccessEvent_AuthAttempt SmLogAccessEvent_AuthChallenge SmLogAccessEvent_ValidateAccept SmLogAccessEvent_ValidateReject SmLogAccessEvent_AuthLogout
Authorization	■ User authorization accepted ■ User authorization rejected	SmLogAccessEvent_AzAccept SmLogAccessEvent_AzReject SmLogAccessEvent_AzUnresolved
Administration	■ Administrator login ■ Administrator rejected ■ Administrator logout	SmLogAccessEvent_AdminLogin SmLogAccessEvent_AdminLogout SmLogAccessEvent_AdminReject
Affiliate	—	SmLogAccessEvent_Visit

Filter Access Events

Beginning with SiteMinder v5.x, you can filter the kinds of access events you want to audit and log using the Auditing tab on the Policy Server Management Console. For example, for each of the four event categories you can select Log All Events or Log No Events.

In addition, for the Authentication, Authorization, and Administration categories, you can select Log Rejection Events Only. For example, if this option is selected for the Authentication category, SmLogAccessEvent_AuthReject events would be logged, but SmLogAccessEvent_AuthAccept events would not be. Also, note the following behavior when Log Rejection Events Only is selected:

- SmLogAccessEvent_AuthAttempt events are not logged.

  A login attempt that does not result in an accepted authentication is considered a failure. However, because the authentication was not actually rejected, events are not logged if Log Rejection Events Only is selected.

  You can use SmLogAccessEvent_AuthAttempt events for intrusion detection.

- SmLogAccessEvent_AuthChallenge events are logged.

  A challenge is not considered a failure. It simply indicates a need for additional authentication information. However, because a challenge involves a rejected authentication, events are logged if Log Rejection Events Only is selected.

EMS Event Type

EMS events result from actions performed on directory objects.

SiteMinder calls EMS events when:

- Directory objects are created, updated, or deleted
- Relationships, such as membership, are formed between objects

After calling an event, SiteMinder logs session activities to the objects. When an EMS-based application, such as Delegated Management Services (DMS), logs in to EMS, a new session is created. EMS validates the login session and reports an appropriate event.

These Directory objects are associated with EMS events:

- Users
- Roles
- Organizations
- Generic, or user-defined directory objects

Each of the objects above is associated with the following object events:

- Create
- Delete
- Modify

EMS events are classified according to category:

- **Administrative** events are generated by a user with sufficient privilege to modify objects in a directory.
- **Session** events are generated when a session is initialized or terminated.
- **End-user** events are generated by users who self-register or modify their own user profiles.
- **Workflow Preprocess** events are generated when a workflow preprocess step is complete.
- **Workflow Postprocess** events are generated when a workflow postprocess step is complete.

Event Category	EMS Event
SmLogEmsCat_DirectoryAdmin	SmLogEmsEvent_CreateUser
	SmLogEmsEvent_DeleteUser
	SmLogEmsEvent_ModifyUser
	SmLogEmsEvent_AssignUserRole
	SmLogEmsEvent_RemoveUserRole
	SmLogEmsEvent_EnableUser
	SmLogEmsEvent_DisableUser
	SmLogEmsEvent_CreateOrg
	SmLogEmsEvent_DeleteOrg
	SmLogEmsEvent_ModifyOrg
	SmLogEmsEvent_CreateRole
	SmLogEmsEvent_DeleteRole
	SmLogEmsEvent_ModifyRole
	SmLogEmsEvent_PasswordModify
	SmLogEmsEvent_CreateObject
	SmLogEmsEvent_DeleteObject
	SmLogEmsEvent_ModifyObject

Event Category	EMS Event
SmLogEmsCat_DirectoryUser	SmLogEmsEvent_CreateUser SmLogEmsEvent_ModifyUser SmLogEmsEvent_PasswordModify
SmLogEmsCat_DirectorySession	SmLogEmsEvent_Login SmLogEmsEvent_Logout SmLogEmsEvent_LoginFail SmLogEmsEvent_SessionTimeout SmLogEmsEvent_AuthFail

Logging Workflow Events

When a preprocess or postprocess event is handled, the event will be logged. In this case, the category in the log is either SmLogEmsCat_EventPreprocess or SmLogEmsCat_EventPostprocess. The Event ID is the original Event ID (for example, SmLogEmsEvent_CreateUser).

In addition, the following fields of SmLog_EMS_t apply to preprocess and postprocess events:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who generated the original event.
szSessionId	The EMS session ID.
szDirName	The name of the SiteMinder directory where the target object is located.
szObjPath	Full distinguished name of the object targeted by this event.
szFieldDesc	Name of the workflow library that executed the event.

Field	Description
szStatusMsg	This includes a workflow completion code and an optional error message. The following workflow completion codes are defined in SmApi.h: ■ Sm_DmsWorkflowApi_Success ■ Sm_DmsWorkflowApi_Ignore ■ Sm_DmsWorkflowApi_Failure ■ Sm_DmsWorkflowApi_Reject ■ Sm_DmsWorkflowApi_NoUserContext ■ Sm_DmsWorkflowApi_SkipSuccess

More Information:

SmLog_EMS_t (see page 621)

Object Event Type

The SiteMinder environment contains elements, called objects, such as domains, policies, realms, and user directories. Collectively, these persistent objects form an object store.

SiteMinder calls object events when objects are created, updated, or deleted. The object events are:

- Object created
- Object updated
- Object deleted

The following SiteMinder objects are associated with object events:

Agents	Agent Groups
Agent Types	Agent Type Attributes
Agent Keys	Key Management
Domains	Administrators
Policies	Policy Links

Password Policies	Registration
User Policies	User Directories
Realms	Management Commands
Responses	Response Groups
Response Attributes	Certificate Mapping
Rules	Rule Groups
Authentication	Authentication and Authorization Mapping
Authentication schemes	ODBC Query
Root	Root Configuration

After calling an object event, SiteMinder logs session activities to the objects. When an application logs in to the object store, a new session is created. SiteMinder validates the login session and reports an appropriate event.

Authentication Events

Authentication reports the following events:

- Login by an application or a user for creating, modifying, or updating an object
- Logout by an application or a user
- Login rejected

Management Command Events

The management commands are not persistent. They log information about management functions, such as flushing cache and changing keys. Management commands are associated with the following events:

- Flush All Caches
- Flush All User Caches
- Flush a Single User from Cache
- Flush Resources
- Change Dynamic Keys
- Change Persistent Key

System Event Type

System events indicate system- and server-related activities. The following table lists the system events:

Activity Type	SiteMinder Event
Server activity	SmLogSystemEvent_ServerInit
	SmLogSystemEvent_ServerInitFail
	SmLogSystemEvent_ServerUp
	SmLogSystemEvent_ServerDown
	SmLogSystemEvent_LogFileOpenFail
	SmLogSystemEvent_ServerHeartBeat
System activity	SmLogSystemEvent_AgentInfo
	SmLogSystemEvent_AgentConnectionStart
	SmLogSystemEvent_AgentConnectionFail
	SmLogSystemEvent_AgentConnectionEnd
	SmLogSystemEvent_DbConnect
	SmLogSystemEvent_DbConnectFail
	SmLogSystemEvent_LdapConnect
	SmLogSystemEvent_LdapConnectFail
	SmLogSystemEvent_AmbiguousResourceMatch
	SmLogSystemEvent_AmbiguousRADIUSMatch
	SmSystemEvent_AgentHeartBeat

SmLogAccessEvent_AuthAccept

This event is called when the user is authenticated.

The following table lists the sssociated SmLog_Access_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szAgentName	Name of the agent.
szSessionId	Contains the session identifier of the user being authenticated.
szClientIp	Contains the IP address of the agent.

Field	Description
szUserName	Name of the user.
SzDomainOid	Contains the object identifier of the domain under which the resource was accessed.
szRealmName	Contains the name of the realm being accessed.
szRealmOid	Contains the object identifier of the realm.
szAuthDirName	Contains the name of the directory against which the user was authenticated.
szAuthDirServer	Contains the authentication directory data source.
szAuthDirNamespace	Contains the namespace of the authentication directory.
szResource	Contains the name of the resource that the user is accessing.
szAction	Contains the action associated with the resource.
szTransactionId	Contains information about the newly created session in the following format: idletime=N;maxtime=N;authlevel=N In the format example: ■ idletime is the idle timeout in seconds of the newly created session ■ maxtime is the maximum timeout in seconds of the newly created session ■ authlevel is the security level of the newly created session

SmLogAccessEvent_AuthReject

This event is called when the user is not authenticated.

The following table lists the sssociated SmLog_Access_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szAgentName	Name of the agent.

Field	Description
szClientIp	Contains the IP address of the agent.
szUserName	Name of the user.
SzDomainOid	Contains the object identifier of the domain under which the resource was accessed.
szRealmName	Contains the name of the realm being accessed.
szRealmOid	Contains the object identifier of the realm.
szAuthDirName	Contains the name of the directory against which the user was authenticated.
szAuthDirServer	Contains the authentication directory data source.
szAuthDirNamespace	Contains the namespace of the authentication directory.
szResource	Contains the name of the resource that the user is accessing.
szAction	Contains the action associated with the resource.

SmLogAccessEvent_AuthAttempt

This event is called when the authentication attempt has failed.

The following table lists the associated SmLog_Access_t fields

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szAgentName	Name of the agent.
szClientIp	Contains the IP address of the agent.
szUserName	Name of the user.
SzDomainOid	Contains the object identifier of the domain under which the resource was accessed.
szRealmName	Contains the name of the realm being accessed.
szRealmOid	Contains the object identifier of the realm.
szResource	Contains the name of the resource that the user is accessing.

Field	Description
szAction	Contains the action associated with the resource.

SmLogAccessEvent_AuthChallenge

This event is called when authentication is challenged.

The following table lists the associated SmLog_Access_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szAgentName	Name of the agent.
szClientIp	Contains the IP address of the agent.
szUserName	Name of the user.
SzDomainOid	Contains the object identifier of the domain under which the resource was accessed.
szRealmName	Contains the name of the realm being accessed.
szRealmOid	Contains the object identifier of the realm.
szResource	Contains the name of the resource that the user is accessing.
szAction	Contains the action associated with the resource.

SmLogAccessEvent_AzAccept

This event is called when the user is authorized to access the resource.

The following table lists the associated SmLog_Access_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szAgentName	Name of the agent.

Field	Description
szSessionId	Contains the session identifier of the user whose authorization is accepted.
szClientIp	Contains the IP address of the agent.
szUserName	Name of the user.
SzDomainOid	Contains the object identifier of the domain under which the resource was accessed.
szRealmName	Contains the name of the realm being accessed.
szRealmOid	Contains the object identifier of the realm.
szResource	Contains the name of the resource that the user is accessing.
szAction	Contains the action associated with the resource.
szTransactionId	Contains the transaction identifier, which is set by the agent to track information returned from the Policy Server. The Policy Server logs this ID.

SmLogAccessEvent_AzReject

This event is called when the user is not authorized to access the resource.

The following table lists the associated SmLog_Access_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szAgentName	Name of the agent.
szSessionId	Contains the session identifier of the user whose authorization is rejected.
szClientIp	Contains the IP address of the agent.
szUserName	Name of the user.
SzDomainOid	Contains the object identifier of the domain under which the resource was accessed.
szRealmName	Contains the name of the realm being accessed.
szRealmOid	Contains the object identifier of the realm.

Field	Description
szResource	Contains the name of the resource that the user is accessing.
szAction	Contains the action associated with the resource.

SmLogAccessEvent_AdminLogin

This event is called when an administrator login is successful.

The following table lists the associated SmLog_Access_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szSessionId	Contains the session identifier of the administrator whose login is successful.
szClientIp	Contains the IP address of the agent.
szUserName	Name of the administrator.

SmLogAccessEvent_AdminLogout

This event is called when an administrator logs out.

The following table lists the associated SmLog_Access_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szSessionId	Contains the session identifier of the administrator who logged out.
szClientIp	Contains the IP address of the agent.
szUserName	Name of the administrator.

SmLogAccessEvent_AdminReject

This event is called when an administrator login is rejected.

The following table lists the associated SmLog_Access_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
nReason	Contains the reason identifier.
szClientIp	Contains the IP address of the agent.
szUserName	Name of the administrator.
szStatusMsg	Contains the reason for the login reject.

SmLogAccessEvent_AuthLogout

This event is called when the authentication server logs out a session.

The following table lists the associated SmLog_Access_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
nReason	The reason identifier, of type Sm_Api_Reason_t. .

SmLogAccessEvent_ValidateAccept

This event is called when a session validation is successful.

The following table lists the associated SmLog_Access_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.

SmLogAccessEvent_ValidateReject

This event is called when a session validation is rejected.

The following table lists the associated SmLog_Access_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.

SmLogEmsEvent_CreateUser, SmLogEmsEvent_DeleteUser, SmLogEmsEvent_ModifyUser

Creating, deleting, or modifying a user object results in an EMS event. User objects are LDAP entities of the class inetorgperson or entities of a class that extends inetorgperson.

The following table lists the associated SmLog_Ems_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	If this is an administrative event, the name of the administrator who is creating, modifying, or deleting the user object. If this is an end-user event, the user name is Registration.
szSessionId	The EMS session ID.
szDirName	The name of the SiteMinder directory where the user is located.
szObjName	The name (user ID) of the user object targeted by this event.
szObjPath	Full distinguished name of the user object targeted by this event.
szObjClass	Class name of the user object.

Field	Description
szOrgName	Name of the user object's organization.
szFieldDesc	Description of the event.
szStatusMsg	Status message. Used only if the event was not handled successfully.

SmLogEmsEvent_PasswordModify

Changing a user password results in an EMS event.

The following table lists the associated SmLog_Ems_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	If this is an Administrative event, the name of the administrator who is modifying the password. If this is an end-user event, the user name is Registration.
szSessionId	The EMS session ID.
szDirName	The name of the SiteMinder directory where the user is located.
szObjName	The name (user ID) of the user object targeted by this event.
szObjPath	Full distinguished name of the user object targeted by this event.
szObjClass	Class name of the user object.
szOrgName	Name of the user object's organization.
szFieldDesc	Description of the event.
szStatusMsg	Status message. Used only if the event was not handled successfully.

SmLogEmsEvent_AssignUserRole, SmLogEmsEvent_RemoveUserRole

Assigning a user to a role or removing a user from a role results in an EMS event. Roles are LDAP entities of the class groupofuniquenames or entities that extend groupofuniquenames.

The following table lists the associated SmLog_Ems_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is assigning or removing the role.
szSessionId	The EMS session ID.
szDirName	The name of the SiteMinder directory where the role is located.
szObjName	The name (user ID) of the user object targeted by this event.
szObjPath	Full distinguished name of the user object targeted by this event.
szObjClass	Class name of the user object.
szOrgName	Name of the user's organization.
szRoleName	Name of the role that is being assigned or removed.
szFieldDesc	Description of the event.
szStatusMsg	Status message. Used only if the event was not handled successfully.

SmLogEmsEvent_EnableUser, SmLogEmsEvent_DisableUser

These events are called when a user's access rights are enabled or disabled. Access rights pertain to the user's ability to get resources that are under SiteMinder protection.

The following table lists the associated SmLog_Ems_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.

Field	Description
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is enabling or disabling the user's access rights.
szSessionId	The EMS session ID.
szDirName	The name of the SiteMinder directory where the user is located.
szObjName	The name (user ID) of the user object targeted by this event.
szObjPath	Full distinguished name of the user object targeted by this event.
szObjClass	Class name of the user object.
szOrgName	Name of the user's organization.
szFieldDesc	Description of the event.
szStatusMsg	Status message. This will be 0 if the user's access rights are enabled, and non-zero if the user's access rights are disabled.

SmLogEmsEvent_CreateOrg, SmLogEmsEvent_DeleteOrg, SmLogEmsEvent_ModifyOrg

These events are called when an organization object is created, deleted, or modified. Organization objects are LDAP entities of the class organizationalunit or entities of a class that extends organizationalunit.

The following table lists the associated SmLog_Ems_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is creating, modifying, or deleting the organization.
szSessionId	The EMS session ID.
szDirName	The name of the SiteMinder directory where the organization is located.

Field	Description
szObjName	The name (user ID) of the organization object targeted by this event.
szObjPath	Full distinguished name of the organization object targeted by this event.
szObjClass	Class name of the organization object.
szFieldDesc	Description of the event.
szStatusMsg	Status message. Used only if the event was not handled successfully.

SmLogEmsEvent_CreateRole, SmLogEmsEvent_DeleteRole, SmLogEmsEvent_ModifyRole

These events are called when a role object is created, deleted, or modified. Role objects are LDAP entities of the class groupofuniquenames or entities of a class that extends groupofuniquenames.

The following table lists the associated SmLog_Ems_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is creating, modifying, or deleting the group.
szSessionId	The EMS session ID.
szDirName	The name of the SiteMinder directory where the group is located.
szObjName	The name of the group targeted by this event.
szObjPath	Full distinguished name of the group object targeted by this event.
szObjClass	Class name of the group object.
szOrgName	Name of the user group’s organization.
szFieldDesc	Description of the event.
szStatusMsg	Status message. Used only if the event was not handled successfully.

SmLogEmsEvent_CreateObject, SmLogEmsEvent_DeleteObject, SmLogEmsEvent_ModifyObject

These events are called when an LDAP entry with an object class of top is created, deleted, or modified. The top object class is a superclass for all other object classes in an LDAP directory. These events apply to any LDAP entry that uses the attribute objectclass.

The following table lists the associated SmLog_Ems_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is creating, modifying, or deleting the object.
szSessionId	The EMS session ID.
szDirName	The name of the SiteMinder directory where the object is located.
szObjName	The name of the object targeted by this event.
szObjPath	Full distinguished name of the object targeted by this event.
szObjClass	Class name of the object.
szFieldDesc	Description of the event.
szStatusMsg	Status message. Used only if the event was not handled successfully.

SmLogEmsEvent_Login, SmLogEmsEvent_Logout

These events are called when an administrator logs in or logs out.

The following table lists the associated SmLog_Ems_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator.

Field	Description
szSessionId	The EMS session ID.
szDirName	The name of the SiteMinder directory where the administrator is located.
szObjName	The name (UID) of the administrator.
szObjPath	Full distinguished name of the administrator.
szOrgName	Name of the user administrator's organization.
szFieldDesc	Description of the event.
szStatusMsg	Status message. Used only if the event was not handled successfully.

SmLogEmsEvent_AuthFail

This event is called when an administration authentication fails.

The following table lists the associated SmLog_Ems_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator.
szSessionId	The EMS session ID. This will probably not exist.
szFieldDesc	Description of the event.
szStatusMsg	Status message.

SmLogEmsEvent_SessionTimeout

This event is called when an EMS server session times out.

The following table lists the associated SmLog_Ems_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.

Field	Description
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator.
szSessionId	The EMS session ID.
szOrgName	Name of the administrator's organization
szFieldDesc	Description of the event.
szStatusMsg	Status message.

SmLogObjEvent_Create

This event is called when system or domain objects are created.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is creating the object.
szSessionId	The session ID of the administrator who is creating the object.
szDomainOid	If the object being created is a domain object, this value is set to the domain object identifier.
szObjName	Name of the object being created.
szObjOid	Object identifier of the object being created.

SmLogObjEvent_Update

This event is called when system or domain objects are updated.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is updating the object.
szSessionId	The session ID of the administrator who is updating the object.
szDomainOid	If the object being created is a domain object, this value is set to the domain object identifier under which the object is updated.
szObjName	Name of the object being updated.
szObjOid	Object identifier of the object being updated.
szFieldDesc	Administrator’s DN.

SmLogObjEvent_Delete

This event is called when system or domain objects are deleted.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is deleting the object.
szSessionId	The session ID of the administrator who is deleting the object.
szDomainOid	If the object being created is a domain object, this value is set to the domain object identifier under which the object is deleted.

Field	Description
szObjName	Name of the object being deleted.
szObjOid	Object identifier of the object being deleted.

SmLogObjEvent_Login

This event is called when an administrator or an application (such as smobjimport) logs into the policy store.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator or application logging into the policy store.
szSessionId	The session ID of the administrator or application logging into the policy store.

SmLogObjEvent_FailedLoginAttemptsCount

This event is called when a user login fails and there is a password policy that applies.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The user whose login attempt failed.
szSessionId	The session ID of the user.
szObjName	Name of directory where the user was found.
szFieldDesc	User’s DN.
szStatusMsg	Number of times that the login was attempted. This number cannot be higher than the number of attempts that results in a disabled account.

SmLogObjEvent_Logout

This event is called when an administrator or an application (such as smobjimport) logs out of the policy store.

The following table lists the associated SmLog_Obj_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator or application logging out of the policy store.
szSessionId	The session ID of the administrator or application logging out of the policy store.

SmLogObjEvent_LoginReject

This event is called when an administrator or an application login to the policy store is rejected.

The following table lists the associated SmLog_Obj_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator or application that was rejected.

SmLogObjEvent_FlushAll

This event is called when all the SiteMinder caches are flushed, including user sessions, resource information, and user directory caches.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is flushing the cache.
szSessionId	The session ID of the administrator who is flushing the cache.

SmLogObjEvent_FlushUser

This event is called when a user is flushed from user session cache.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is flushing the user.
szSessionId	The session ID of the administrator who is flushing the user.
szObjName	Name of the directory where the user is defined.
szObjOid	Object identifier of the directory where the user is defined.
szFieldDesc	DN of the user being flushed.

SmLogObjEvent_FlushUsers

This event is called when all users are flushed from user session cache.

The following table lists the associated SmLog_Obj_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is flushing the users.
szSessionId	The session ID of the administrator who is flushing the users.

SmLogObjEvent_FlushRealms

This event is called when all the realms are flushed from resource information cache.

The following table lists the associated SmLog_Obj_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is flushing the realms.
szSessionId	The session ID of the administrator who is flushing the realms.

SmLogObjEvent_ChangeDynamicKeys

This event is called when the dynamic key is changed.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is changing the keys.
szSessionId	The session ID of the administrator who is changing the keys.

SmLogObjEvent_ChangePersistentKey

This event is called when the persistent key is changed.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is changing the keys.
szSessionId	The session ID of the administrator who is changing the keys.

SmLogObjEvent_ChangeSessionKey

This event is called when the session key is changed.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.

Field	Description
nCurrentTime	Time when the event occurred.

SmLogObjEvent_ChangeUserPassword

This event is called whenever an administrator changes a user's password.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is changing the password.
szSessionId	The session ID of the administrator who is changing the password.
szObjName	Name of the directory where the user is defined.
szObjOid	Object identifier of the directory where the user is defined.
szFieldDesc	DN of the user whose password is being changed.

SmLogObjEvent_CreateUserSuccess

This event is called when a new user is created.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is creating the user account.

Field	Description
szSessionId	The session ID of the administrator who is creating the user account.
szObjName	Name of the directory where the user is defined.
szFieldDesc	DN of the user whose account is being created.

SmLogObjEvent_DeleteUserSuccess

This event is called when a user account is deleted in the directory.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is deleting the user account.
szSessionId	The session ID of the administrator who is deleting the user account.
szObjName	Name of the directory where the user is defined.
szFieldDesc	DN of the user whose account is being deleted.

SmLogObjEvent_ChangeDisabledUserState

This event is called under the following conditions:

- When Admin enables or disables the user account.
- When you enable the account through the Administrative UI or the SDK.
- When the user is disabled because of account inactivity.
- When the user is disabled because the number of login failures was exceeded.
- When the user is disabled because the password has expired.
- When you set or unset "User must change password" in the Administrative UI.
- When you change a user password because changing the password triggers the user must enter a new password.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is changing the state of the account.
szSessionId	The session ID of the administrator who is changing the state of the account.
szObjName	Name of the directory where the user is defined.
szFieldDesc	DN of the user whose account is being changed.
szStatusMsg	Reason for changing the state of the account.

SmLogObjEvent_ModifyUserSuccess

This event is called when a user account is modified.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.

Field	Description
szUserName	The name of the administrator who is modifying the user account.
szSessionId	The session ID of the administrator who is modifying the user account.
szObjName	Name of the directory where the user is defined.
szFieldDesc	DN of the user whose account is being modified.

SmLogObjEvent_CreateUserFail

This event is called when the user account cannot be created.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is attempting to create the user account.
szSessionId	The session ID of the administrator who is attempting to create the user account.
szObjName	Name of the directory where the user is defined.
szFieldDesc	DN of the user whose account could not be created.

SmLogObjEvent_DeleteUserFail

This event is called when the user account cannot be deleted.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.

Field	Description
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is attempting to delete the user account.
szSessionId	The session ID of the administrator who is attempting to delete the user account.
szObjName	Name of the directory where the user is defined.
szFieldDesc	DN of the user whose account could not be deleted.

SmLogObjEvent_ModifyUserFail

This event is called when the user account cannot be modified.

The following table lists the associated SmLog_Obj_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szUserName	The name of the administrator who is attempting to modify the user account.
szSessionId	The session ID of the administrator who is attempting to modify the user account.
szObjName	Name of the directory where the user is defined.
szFieldDesc	DN of the user whose account could not be modified.

SmLogSystemEvent_ServerInit

The server is initializing.

The following table lists the associated SmLog_System_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	Name of the server.
szMsg	Information about the server in the following format: Product=%,Platform=%,Version=%,Update=%, Label=%,Crypto=%,UTC=%,TZ=% In the format example: Product is the name of the server. Platform is the supported platform of the server software. Version is the version number of the SiteMinder server software. Update is the update number of the server software. Label is the build number of the server software. Crypto is the crypto strength. UTC is the time in the Universal Time Format. TZ is the time zone. Unknown clauses are left blank. Example: Product=smservacct,Platform=Windows NT, Version=4.0,Update=None,Label=C144, Crypto=56,UTC=949621705,TZ=5

SmLogSystemEvent_ServerInitFail

Called by the event handler to find out which server initialization failed according to the system category in the event call.

This event has no associated SmLog_System_t fields.

SmLogSystemEvent_ServerUp

Called by the event handler to find out which server is up and running according to the system category in the event call.

This event has no associated SmLog_System_t fields.

SmLogSystemEvent_ServerDown

Called by the event handler to find out which server is down according to the system category in the event call.

This event has not associated SmLog_System_t fields.

SmLogSystemEvent_LogFileOpenFail

This event is called when the text file used for audit logging could not be opened.

The following table lists the associated SmLog_System_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szMsg	The name of the audit log file that could not be opened.

SmLogSystemEvent_ServerHeartbeat

This event provides periodic information about the server. It is called every 30 seconds.

The following table lists associated SmLog_System_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	Name of the server.

Field	**Description**
szIpAddress	Name of the host where the server is running.
nIpPort	Port on which the server is listening.

SmLogSystemEvent_AgentInfo

Contains information about the agent.

The following table lists the associated SmLog_System_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	Name of the agent.
szMsg	Information about the agent in the following format: Product=%,Platform=%,Version=%,Update=%, Label=%,Crypto=%,UTC=%,TZ=% In the format example: Product is the name of the agent software. Platform is the supported platform of the agent software. Version is the version number of the agent software. Update is the update number of the agent software. Label is the build number of the agent software. Crypto is the crypto strength. UTC is the time in the Universal Time Format. TZ is the time zone. Unknown clauses are left blank. For example, the Web Agent issues the following: Product=WebAgent,Platform=NT/ISAPI, Version=4.0,Update=SP1,Label=C134, Crypto=56,UTC=949621705,TZ=5

SmLogSystemEvent_AgentConnectionStart

This event is called when the agent connects to the Policy Server.

The following table lists the associated SmLog_System_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	Name of the agent.
szIpAddress	The IP Address of the agent machine.
nIpPort	Port on the Policy Server machine that the agent is connected to.

SmLogSystemEvent_AgentConnectionFail

This event is called when the agent connection fails.

The following table lists the associated SmLog_System_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	Name of the agent.
szIpAddress	The IP Address of the agent machine.
nIpPort	Port on the Policy Server machine that the agent tried to connect to.

SmLogSystemEvent_AgentConnectionEnd

This event is called when the agent connection ends.

The following table lists the associated SmLog_System_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.

Field	**Description**
nCurrentTime	Time when the event occurred.
szName	Name of the agent.
szIpAddress	The IP Address of the agent machine.
nIpPort	Port on the Policy Server machine that the agent was connected to.

SmLogSystemEvent_DbConnect

This event is called when the Policy Server connects to the database.

The following table lists the associated SmLog_System_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	The name of the data source.

SmLogSystemEvent_DbConnectFail

This event is called when the Policy Server connection to the database fails.

The following table lists the associated SmLog_System_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	The name of the data source.
szMsg	The error message.

SmLogSystemEvent_LdapConnect

This event is called when the Policy Server connects to an LDAP directory.

The following table lists the associated SmLog_System_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	The name of the LDAP directory.
szIpAddress	The IP Address of the LDAP directory.
nIpPort	Port of the LDAP directory.

SmLogSystemEvent_LdapConnectFail

This event is called when the Policy Server connection to the LDAP directory fails.

The following table lists the associated SmLog_System_t fields:

Field	**Description**
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	The name of the LDAP directory.
szIpAddress	The IP Address of the LDAP directory.
nIpPort	Port of the LDAP directory.
szMsg	The error message.

SmLogSystemEvent_AmbiguousResourceMatch

This event is called when there is an ambiguous resource match.

The following table lists the associated SmLog_System_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	Name of the machine requesting the resource.
szMsg	The resource requested.

SmLogSystemEvent_AmbiguousRadiusMatch

This event is called when there is an ambiguous RADIUS match.

The following table lists the associated SmLog_System_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	The name of the RADIUS device.

SmLogSystemEvent_AgentHeartbeat

This event is called whenever an agent makes a DoManagement request.

The following table lists the associated SmLog_System_t fields:

Field	Description
nVersion	Version number of the SiteMinder server.
nCurrentTime	Time when the event occurred.
szName	Name of the agent.
szIpAddress	IP address of the agent.

Field	Description
nIpPort	Port on the Policy Server machine to which the agent is connected.

Event Function Declarations

The table below lists the functions used in the Event API. Your shared library must export these entry points.

Function Definition	Description
SmEventInit() (see page 663)	Called by the Policy Server so that an event handler can perform its own initialization procedure.
SmEventRecord() (see page 664)	Called by the Policy Server when an event has been signalled.
SmEventRelease() (see page 665)	Called by the Policy Server so that an event handler can perform its own rundown procedure.

SmEventInit()

The Policy Server calls this function to let an event provider perform its own initialization procedure. The call is made once when the provider is first loaded.

Syntax

```
int SM_EXTERN SmEventInit();
```

Returns

Returns 1 if successful or 0 if unsuccessful.

SmEventRecord()

The Policy Server calls this function when an event has been signalled.

Syntax

```
void SM_EXTERN SmEventRecord (
   const int   nCategoryType,
   const int   nCategory,
   const int   nEventId,
   void*       pData
);
```

Parameter	I/O	Description
nCategoryType	I	The type of the event being logged. One of the following valid values: ■ SmLog_Access ■ SmLog_EMS ■ SmLog_Obj ■ SmLog_System
nCategory	I	The event category. A list of valid event categories follows this parameter table.
nEventID	I	The event.
pData	I	The pointer to the property instance. The structure could be one of the following: ■ SmLog_Access_t (see page 619) ■ SmLog_EMS_t (see page 621) ■ SmLog_Obj_t (see page 622) ■ SmLog_System_t (see page 623)

Valid values for the *nCategory* parameter are as follows:

SmLogAccessCat_Admin	SmLogObjCat_PropertyCollection
SmLogAccessCat_Affiliate	SmLogObjCat_PropertySection
SmLogAccessCat_Auth	SmLogObjCat_Realm
SmLogAccessCat_Az	SmLogObjCat_Response
SmLogEmsCat_DirectoryAdmin	SmLogObjCat_ResponseAttr
SmLogEmsCat_DirectorySession	SmLogObjCat_ResponseGroup
SmLogEmsCat_DirectoryUser	SmLogObjCat_Root
SmLogEmsCat_EventPostprocess	SmLogObjCat_RootConfig
SmLogEmsCat_EventPreprocess	SmLogObjCat_Rule
SmLogObjCat_ActiveExpr	SmLogObjCat_RuleGroup
SmLogObjCat_Admin	SmLogObjCat_Scheme
SmLogObjCat_Agent	SmLogObjCat_SelfReg
SmLogObjCat_AgentGroup	SmLogObjCat_TaggedString
SmLogObjCat_AgentKey	SmLogObjCat_UserDirectory
SmLogObjCat_Auth	SmLogObjCat_UserPolicy
SmLogObjCat_AuthAzMap	SmLogObjCat_Variable
SmLogObjCat_CertMap	SmLogObjCat_VariableType
SmLogObjCat_Domain	SmLogObjCat_Vendor
SmLogObjCat_KeyManagement	SmLogObjCat_VendorAttr
SmLogObjCat_ManagementCommand	SmLogSystemCat_Acct
SmLogObjCat_ODBCQuery	SmLogSystemCat_Admin
SmLogObjCat_PasswordPolicy	SmLogSystemCat_Auth
SmLogObjCat_Policy	SmLogSystemCat_Az
SmLogObjCat_PolicyLink	SmLogSystemCat_Combined
SmLogObjCat_Property	SmLogSystemCat_System

SmEventRelease()

The Policy Server calls this function to let the event handler perform its own rundown procedure. This call is made once when SiteMinder is shutting down.

Syntax

```
void SM_EXTERN SmEventRelease();
```

Example of an Active Policy

This function returns true if the user belongs to the organizational unit specified in the parameter *(param*) field of the active policy expression.

```
<@ lib="SmAzAPI" func="activePolicy" param="Accounting" @>
*************************************************************
int SM_EXTERN activePolicy(
const Sm_Api_Context_t* lpApiContext,
// the structure that provides API context
const Sm_Api_UserContext_t*  lpUserContext,
// the structure that provides user context
const Sm_Api_RequestContext_t*  lpReqContext,
// the structure that provides request context
const char* lpszParam,
// the parameter string (null-terminated)
const int nBytesOutBuf,
// the maximum size of the output buffer
char*  lpszOutBuf,
// the output buffer to hold the null-terminated attribute value
const int  nBytesErrBuf,
// the maximum size of the error message buffer
char* lpszErrBuf)
// the output buffer to hold the null-terminated error message
{
/* User Context is required to use the functions like fGetProp, fSetProp.. */
if(!lpUserContext->bIsUserContext) {
   strncpy (lpszErrBuf, "No User Context ", nBytesErrBuf);
   lpszErrBuf[nBytesErrBuf-1] = '\0';
   return -1;
   }
/* Buffer to store all the organizational units user belongs to. */
char lpszOrgUnit[30];
memset(lpszOrgUnit, 0, sizeof(lpszOrgUnit));
/*
// Check to see if an organizational unit has been
// entered in the parameter.
*/
if(lpszParam == NULL || strlen(lpszParam) == 0)
   {
   strncpy (lpszErrBuf, "Organizational unit is not entered ",
            nBytesErrBuf);
   lpszErrBuf[nBytesErrBuf-1] = '\0';
   return -1;
   }
/* Get all the organizational units to which the user belongs. */
```

```
   int getResult = lpUserContext->fGetProp (
   lpUserContext->lpParam,
   "ou",         /* Attribute name */
   sizeof (lpszOrgUnit),lpszOrgUnit);

if (getResult < 0)
   {
   strncpy (lpszErrBuf,
       "Failed to get user password from user's profile attribute ",
       nBytesErrBuf);
   lpszErrBuf[nBytesErrBuf-1] = '\0';
   return -1;
   }
    else
    {
   /* Check if the user belongs to the organization unit that is
   requested. */
   if(strstr(lpszOrgUnit, lpszParam) != NULL)
      {
/*
   // Yes the user belongs to the organization unit
   // mentioned in the parameter field of active policy.
*/
      strncpy(lpszOutBuf, "true", nBytesOutBuf);
      lpszOutBuf[nBytesOutBuf-1] = '\0';
      return strlen(lpszOutBuf);
      }
      else
      {
      strncpy (lpszErrBuf,
         "The user does not belong to the requested organizational unit ",
         nBytesErrBuf);
      lpszErrBuf[nBytesErrBuf-1] = '\0';
      return -1;
      }
   }
    /* everything failed.... */
    return 0;
}
```

Configuring the Policy Server for the Event Handler

You can add additional event handler libraries to the CA SiteMinder® Policy Server.

Note: If you do not have write access to the CA SiteMinder® binary files (XPS.dll, libXPS.so, libXPS.sl), an Administrator must grant you permission to use the related XPS command line tools using the Administrative UI or the XPSSecurity tool.

To add event handler libraries

1. Open a command line on the Policy Server, and enter the following command:

   ```
   xpsconfig
   ```

   The tool starts and displays the name of the log file for this session, and a menu of choices opens.

2. Enter the following:

   ```
   xps
   ```

   A list of options appears.

3. Enter the following:

   ```
   5 (AuditSMHandlers)
   ```

   The settings for the event handler libraries appear.

4. Type C, and then enter the path and file name of the event handler library you want to add. Separate multiple library locations with commas.

   The settings for the event handler libraries appear. The value you added is shown at the bottom of the settings as a "pending value."

5. Do the following:

   a. Enter Q twice.

   b. Enter L.

   c. Enter Q to end your XPS session.

   Your changes are saved and the command prompt appears.

Chapter 11: DMS Workflow API

This section contains the following topics:

DMS Workflow API Guidance

Using the DMS Workflow API, you can add simple pre- and postprocess workflow for DMS events. To implement these processes, you must:

1. Develop a shared library that supports the DMS Workflow API and provides the custom functionality you need. The shared library must contain the workflow functions defined as exportable symbols.

   SmApi.h defines all of the data structures necessary to create custom workflows. SmEventApi.h defines all of the workflow events.

2. Install the shared library in one of the following locations:
   - On UNIX platforms, in the SiteMinder lib directory
   - On Windows platforms, in the SiteMinder bin directory
3. Define the workflow library in the Administrative UI. Every DMS Administration realm and every resource realm into which an end-user may self-register is associated with a Registration Scheme. The Registration Scheme Properties UI contains a text field where you may enter the name of the shared library to use for DMS workflow.

Note: The DMS API (available in Java only) has different functionality than the DMS Workflow API (available in C/C++ only). The DMS API lets you develop directory management applications that perform similar operations as the SiteMinder DMS product. The DMS Workflow API works in conjunction with DMS and fires when certain pre-process and post-process DMS events occur, allowing you to develop applications that perform additional functionality before and/or after these events.

DMS Sessions

A user working with a DMS application is assigned a session to maintain the application context according to the user's administrative privileges.

There are two types of DMS user sessions:

- An **Administration session** is created for a user who has administrative privileges in the DMS application. In order to start a DMS session, the user must be authenticated and authorized to use the application. Administrative sessions generate events which are categorized as administrative events. Such events provide a user context to the workflow libraries.
- **Registration sessions** are created for registration realms where users will be adding themselves to the directory. A registration session is created for a realm upon the first request to self-register for the resources in that realm. The session remains live on the DMS server until SiteMinder is shut down. Registration sessions generate events that are categorized as registration events. Such events will not provide a user context to the workflow libraries. Registration events are limited to the creation of a user entry and the self-modification of a user entry.

DMS Workflow

When a user initiates a DMS session, either by requesting self-registration or by entering a DMS administration application, the following SiteMinder objects are located:

- The realm of the requested resource. For self-registration, this is the realm containing the page for which the user is signing up. For DMS administrators, the realm is the location of the DMS Administration servlet. The realm is obtained by the SiteMinder Web Agent when the user requests access.
- The registration scheme associated with the realm. The registration scheme contains the name of the workflow library, if any.

  Because workflow is associated with a realm, it is possible to set up several workflow libraries, depending on the desired DMS event processing for a particular resource. For example, self-registration may be required in order to receive documents on separate sites, but the rules for pre- and postprocessing users for each site may be different.

  For example, it is possible to set up a number of self-registering sites; each may have its own workflow library.

If the registration scheme includes a workflow library name, the library is loaded. The loader checks to see that all the required functions are exported by the library. If the library can not be loaded or a function is missing, session initialization will fail and DMS requests for that realm will not be processed. To correct this situation:

1. Make sure the library has been installed in the correct location, as specified in DMS Workflow API Overview.
2. Ensure that the required functions are exported by the library.
3. If desired, disable the workflow by removing the library name from the registration scheme, using the Administrative UI.

When the workflow library is loaded, it will be called for every DMS event that involves adding, modifying, or deleting entities in the user directory during the session.

Workflow Events

Workflow events are documented in EMS Event Type. These events are used for logging as well as for workflow. Events are divided into these categories:

- SmLogEmsCat_DirectoryUser. Assigned to events generated by an end-user, such as self-registration and modification.
- SmLogEmsCat_DirectoryAdmin. Assigned to events generated by a DMS administrator.
- SmLogEmsCat_DirectorySession. Assigned to events associated with DMS session management.
- SmLogEmsCat_EventPreprocess. Assigned to events generated by a workflow preprocess step; used for logging the result of a workflow event.
- SmLogEmsCat_EventPostprocess. Assigned to events generated by a workflow postprocess step; used for logging the result of a workflow event.

More Information:

EMS Event Type (see page 625)

Preprocess Events

A preprocess event takes place before the DMS request is processed. A workflow library may evaluate the request and decide whether to accept or reject it. If accepted, the request will be carried out. If rejected, the request will not be carried out and a preprocess error will be returned to the DMS application. The preprocess function may optionally set an error message for DMS to display to the user.

The workflow library is called to preprocess all events in the categories SmLogEmsCat_DirectoryUser and SmLogEmsCat_DirectoryAdmin.

Postprocess Events

A postprocess event takes place after the DMS request is successfully processed. A workflow library may evaluate the request, take any action as dictated by the business process, and return success or fail status. The postprocess function may optionally set an error message for DMS to display to the user. It is important to note that postprocess failure will not result in rolling back the transaction. Unacceptable requests should be detected in preprocessing.

The workflow library is called to postprocess all events in the SmLogEmsCat_DirectoryUser and SmLogEmsCat_DirectoryAdmin categories, as well as the SmLogEmsCat_DirectorySession category. This last category provides session status only.

DMS Workflow API Reference

To create a custom workflow library:

1. Include the SmApi.h file, as follows:
   ```
   #include "SmApi.h"
   ```
2. Include the SmEventApi.h file, as follows:
   ```
   #include " SmEventApi.h"
   ```
3. Make sure the following functions are externally visible:

Function	Description
SmDmsWorkflowInit() (see page 677)	Initializes the workflow library. The library should initialize whatever resources it needs at this time.
SmDmsWorkflowPostprocess() (see page 677)	Performs the workflow postprocess step.

SmDmsWorkflowPreprocess() (see page 679)	Performs the workflow preprocess step.
SmDmsWorkflowRelease() (see page 680)	The library should release whatever resources were created by initialization at this time.

Each entry point in the shared library must be defined according to specified syntax.

Note: If you are using Microsoft Visual Studio, export the function addresses to a modular definition file (.DEF) file. To export the function addresses, create a .DEF file. In the file's export section, list all the functions described in the previous table. Once you have created the .DEF file, add it to the Microsoft Visual Studio project.

Compile the code into a DLL or shared library. The name of this file will be specified as the library in the SiteMinder Registration Properties dialog.

When you have written a custom workflow, such as the example provided at the end of this chapter, you must define the workflow in the Registration Properties dialog using the Administrative UI.

Sm_Api_DmsContext_t

This structure provides DMS-specific context.

Syntax

```
typedef struct
{
   unsigned char       bIsUserContext;
   char*               lpszAdminUserName
   char*               lpszAdminUserPath;
   char*               lpszDirPath;
   void                lpReserved1;
   char*               lpszDirServer;
   char*               lpszDirNamespace;
   char*               lpszSessionId;
   void*               lpDirParam;
   void*               lpAdminParam;
   Sm_Api_GetUserProp  fGetAdminProp;
   Sm_Api_GetDmsDnProp fGetDnProp;
} Sm_Api_DmsContext_t;
```

Field	Description
bIsUserContext	Flag indicating that SiteMinder has established the user's identity and stored it in *lpszAdminUserName*. When this flag is set, the function referenced in *fGetAdminProp* can be used to obtain properties of this user.
lpszAdminUserName	The DMS administrator user name. This will be a full distinguished name for Administrator events, else it will be an empty string.
lpszAdminUserPath	The DMS administrator user path in the SiteMinder notation, for Administrator events only.
lpszDirPath	Directory path, in the SiteMinder notation, of a SiteMinder user directory specified by the registration scheme.
lpReserved1	Reserved for internal use.
lpszDirServer	The directory server of a SiteMinder user directory specified by the registration scheme.
lpszDirNamespace	The directory namespace.
lpszSessionId	The DMS session ID.

Field	Description
lpDirParam	Pointer to the parameters to be passed to *fGetDnProp*.
lpAdminParam	Pointer to the parameters to be passed to *fGetAdminProp*.

fGetAdminProp	Function that returns attributes of the administrative user. The calling syntax for this function is:
	`int nMaxBufSize = 256;`
	`char lpszResult[256];`
	`int nBuflen = lpDmsContext->fGetProp (`
	`  lpDmsContext->lpAdminParam,`
	`  "mail",`
	`  nMaxBufSize,`
	`  lpszResult);`
	If the function succeeds, the return value is the number of bytes in the output buffer. If the function fails, the return value is -1.

fGetDnProp	Function that returns attributes given a distinguished name. Any object may be examined as long as it can be retrieved from the directory server. The calling syntax for this function is:
	`int nMaxBufSize = 256;`
	`char lpszResult[256];`
	`char lpszDn[] =`
	`  "uid=jsmith,ou=People,o=dms.com";`
	`int nBuflen = lpDmsContext->fGetProp (`
	`  lpDmsContext->lpDirParam,`
	`  lpszDn,`
	`  "mail",`
	`  nMaxBufSize,`
	`  lpszResult);`
	If the function succeeds, the return value is the number of bytes in the output buffer. If the function fails, the return value is -1.

Function Declarations

In structure Sm_Api_DmsContext_t, the functions fGetAdminProp and fGetDnProp are declared in SmApi.h as follows:

fGetAdminProp

```
typedef int (SM_EXTERN *Sm_Api_GetUserProp)
(
const void* lpParam,        /* The function parameter */
const char* lpszPropName,   /* User property name (null-term) */
const int nBytesValueBuf,   /* Max size of user property buffer */
char* lpszValueBuf   /* Output buffer to hold the user property */
);
```

fGetDnProp

```
typedef int (SM_EXTERN *Sm_Api_GetDmsDnProp)
(
const void* lpParam,       /* The function parameter */
const char* lpszDN,        /* The DN of the object */
const char* lpszPropName,  /* Object property name (null-term) */
const int nBytesValueBuf,  /* Max size of object property buffer */
char* lpszValueBuf  /* Output buffer to hold the object property */
);
```

Sm_DmsWorkflow_Attribute_t

This structure pairs a request or response attribute name with a string representation of the attribute value. An array of these structures is used to represent a DMS request to create or modify a directory object.

Syntax

```
typedef struct
{
   const char* lpszAttrName;
   const char* lpszAttrValue;
} Sm_DmsWorkflow_Attribute_t;
```

Field	Description
lpszAttrName	Name of the request or response attribute.
lpszAttrValue	String representing the attribute value.

SmDmsWorkflowInit()

SiteMinder calls this function so that a workflow library can perform its own initialization procedure. This call is made once, when the library is loaded for the first time.

Syntax

```
Sm_DmsWorkflowApi_Status_t SM_EXTERN SmDmsWorkflowInit();
```

Returns

- Sm_DmsWorkflowApi_Success. Function was successful.
- Sm_DmsWorkflowApi_Failure. Function was not successful, and the workflow will not be loaded. If the workflow is not loaded successfully, DMS events that depend on the registration scheme that specified the workflow will fail until:
    - The workflow library can load successfully
    - The workflow library is removed from the registration scheme

SmDmsWorkflowPostprocess()

SiteMinder calls this function so that a workflow library can postprocess a DMS event.

Syntax

```
Sm_DmsWorkflowApi_Status_t SM_EXTERN SmDmsWorkflowPostprocess(
   const Sm_Api_Context_t*           lpApiContext,
   const Sm_Api_DmsContext_t*        lpDmsContext,
   const char *                      lpszWorkflowOid,
   const int                         nCategoryId,
   const int                         nEventId,
   const char *                      lpszObjectPath,
   const int                         nAttributes,
   const Sm_DmsWorkflow_Attribute_t*  pAttributes,
   const int                         nBytesErrMsg,
   char *                            lpszErrMsg
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
lpDmsContext	I	Pointer to the DMS context structure.
lpszWorkflowOid	I	Unique workflow identifier.

Parameter	I/O	Description
nCategoryId	I	DMS event category.
nEventId	I	DMS event identifier
lpszObjectPath	I	Distinguished name of the object that is the target of this event, if any. The target object DN will be available for requests to create, delete, and modify directory objects.
nAttributes	I	Number of attributes in the array of response attributes.
pAttributes	I	Array of DMS response attribute/value pairs.
nBytesErrMsg	I	Maximum size of the output error buffer.
lpszErrMsg	O	Output buffer to receive the error text. Use this buffer to return an error message to the DMS application.

Returns

- Sm_DmsWorkflowApi_Success. Function was successful.
- Sm_DmsWorkflowApi_Failure. Function was not successful.
- Sm_DmsWorkflowApi_Ignore. The event was of no interest to this step.

Remarks

The event has already been processed. Return values of Sm_DmsWorkflowApi_Success or Sm_DmsWorkflowApi_Ignore result in a success status returned to the application. Sm_DmsWorkflowApi_Failure causes an error code to be returned to the application. The error code indicates that there was a failure in the DMS postprocessing step.

The character array *lpszErrMsg* may be used by the postprocess function to send an error message back to the application. The maximum size of the character array is specified in *nBytesErrMsg*.

SmDmsWorkflowPreprocess()

SiteMinder calls this function so that a workflow library can preprocess a DMS event.

Syntax

```
Sm_DmsWorkflowApi_Status_t SM_EXTERN SmDmsWorkflowPreprocess(
   const Sm_Api_Context_t*            lpApiContext,
   const Sm_Api_DmsContext_t*         lpDmsContext,
   const char *                       lpszWorkflowOid,
   const int                          nCategoryId,
   const int                          nEventId,
   const char *                       lpszObjectPath,
   const int                          nAttributes,
   const Sm_DmsWorkflow_Attribute_t*  pAttributes,
   const int                          nBytesErrMsg,
   char *                             lpszErrMsg
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
lpDmsContext	I	Pointer to the DMS context structure.
lpszWorkflowOid	I	Unique workflow identifier.
nCategoryId	I	DMS event type.
nEventId	I	DMS event identifier
lpszObjectPath	I	Distinguished name of the object that is the target of this event, if any. The target object DN will be available for requests to create, delete, and modify directory objects.
nAttributes	I	Number of attributes in the array of request attributes.
pAttributes	I	Array of DMS request attribute/value pairs.
nBytesErrMsg	I	Maximum size of the output error buffer.
lpszErrMsg	O	Output buffer to receive the error text. Use this buffer to return an error message to the DMS application.

Returns

- Sm_DmsWorkflowApi_Success. Function was successful.
- Sm_DmsWorkflowApi_Failure. Function was not successful.
- Sm_DmsWorkflowApi_NoUserContext. An administrative user event was generated and the user context could not be determine.
- Sm_DmsWorkflowApi_SkipSuccess. The event will be handled by the preprocess function and no further processing is desired.
- Sm_DmsWorkflowApi_Ignore. The event was of no interest to this step.

Remarks

A return of Sm_DmsWorkflowApi_Success or Sm_DmsWorkflowApi_Ignore causes processing to continue. Any other return causes processing to stop, and an error code is returned to the application. The error code indicates that there was a failure in the DMS preprocessing step.

Sm_DmsWorkflowApi_SkipSuccess causes processing to stop, but a success status is returned to the application. It is up to the implementation of the preprocess function to fulfill the desired goal. If the function fails, it should return a failure code rather than Sm_DmsWorkflowApi_SkipSuccess.

The character array *lpszErrMsg* may be used by the preprocess function to send an error message back to the application. The maximum size of the character array is specified in *nBytesErrMsg*.

SmDmsWorkflowRelease()

SiteMinder calls this function so that a workflow library can perform its own rundown procedure. This call is made once, when SiteMinder is shutting down.

Syntax

```
Sm_DmsWorkflowApi_Status_t SM_EXTERN SmDmsWorkflowRelease();
```

Returns

- Sm_DmsWorkflowApi_Success. Function was successful.
- Sm_DmsWorkflowApi_Failure. Function was not successful.

Chapter 12: Directory API Guidance

This section contains the following topics:

Purpose of the Directory API

The Directory API accesses data stored in a type of database or directory that SiteMinder does not support. With the Directory API, you can:

- Add a non-supported directory entry (user) to a SiteMinder policy
- Manage non-supported directory entries

SiteMinder supports the following namespaces for user directories:

- LDAP
- ODBC
- Microsoft Windows NT
- Custom

The LDAP, ODBC and NT namespaces can be used without the Directory API. To access another type of user directory, create an interface to the directory with the Directory API.

Before You Use the Directory API

Before using the Directory API, go to the SiteMinder Administrative UI and create and configure a user directory object with a Custom namespace.

To use the Directory API, you must have the following:

- One or more application developers who have experience with the following tasks:
  - Coding in the C programming language
  - Building shared libraries
  - Using relevant APIs provided by the vendor of the desired directory type
- A supported development environment for building a shared library.

  Because no linking is required, there is no restrictive list of supported development environments. For example, you can use the GNU Compiler Collection (GCC). The Directory API has been tested using the following development environments:
  - UNIX platforms: Sun Visual WorkShop C++ 5.0
  - Windows platforms: Microsoft Visual C++ 6.0
- The SiteMinder Policy Server

  Although you don't need the SiteMinder Policy Server to build a Directory API application, you need the SiteMinder Policy Server to use the application. The SiteMinder Policy Server calls the Directory API.

How to Use the Directory API

When you have met the prerequisites, follow these steps:

1. Review the sample provided with the Directory API.
2. Write source code to implement the Directory API.
3. Build a shared library from the source code.
4. Place the shared library in the default location, as follows:
   - On UNIX platforms, in the SiteMinder lib directory
   - On Windows platforms, in the SiteMinder bin directory

   Optionally, you can place the shared library in a different location, as long as the SiteMinder Policy Server can access it. If you use an alternate location, indicate the fully qualified path to the shared library in the Library field of the SiteMinder User Directory Dialog box.

Build a Directory Application

When you build a Directory API application, include the SmAPI.h file. Your source code must contain the statement:

```
#include "SmApi.h"
```

When building a shared library that implements the Directory API, you need not link to any other shared libraries or import libraries. The Directory API is built as a shared library with the exportable functions defined in the include file SmApi.h.

Exported Enumerations

SmApi.h includes the following enumerations used by the Directory API:

- Sm_DirApi_Capability_t (directory capabilities)
- Sm_PolicyResolution_t (policy resolutions)

Directory Capabilities

Sm_DirApi_Capability_t enumerates the capabilities that can be configured for a custom directory.

The following table lists the directory capabilities enumerated in Sm_DirApi_Capability_t. Descriptions of each capability follow the table.

Name	Value
Sm_DirApi_Capability_ForceResetUserPassword	0x00000001
Sm_DirApi_Capability_ChangeUserPassword	0x00000002
Sm_DirApi_Capability_DisableUser	0x00000004
Sm_DirApi_Capability_SetUserAttributes	0x00000008
Sm_DirApi_Capability_Recursive	0x00000010

For a custom directory to have a specific capability, you must define the required user attributes for that capability. For example, to enable SiteMinder to change a user's password, you need to identify a Password Attribute. SiteMinder then uses that attribute to get and set the user password.

- Sm_DirApi_Capability_ForceResetUserPassword. The custom directory is capable of forcing user password reset.

  To enable SiteMinder to force a reset of the password, define the following user attributes:

  - Password attribute. An attribute that SiteMinder can use to get and set the user password. In the SiteMinder Administrative UI, enter that attribute name in the Password Attribute field on the User Attributes tab on the User Directory Dialog box. In the sample, the attribute name is password.
  - Disabled Flag. An attribute that SiteMinder can use to get and set the disabled state of a user. In the SiteMinder Administrative UI, enter that attribute name in the Disabled Flag field on the User Attributes tab on the User Directory Dialog box. In the sample, the attribute name is Disabled.

  When users are forced to change their passwords, the Policy Server calls SmDirSetUserDisabledState(). In the sample code, the user's Disabled Flag is set to Sm_Api_Disabled_PWMustChange (the disabled reason).

- Sm_DirApi_Capability_ChangeUserPassword. The custom directory is capable of changing the user password. To change the password, you need to identify a Password Attribute, which is an attribute that SiteMinder can use to get and set the user password. In the SiteMinder Administrative UI, enter that attribute name in the Password Attribute field on the User Attributes tab on the User Directory Dialog box.

  The SiteMinder Policy Server calls SmDirChangeUserPassword() so that you can change the value in the password field for an entry in your custom directory.

- Sm_DirApi_Capability_DisableUser. The custom directory is capable of disabling a user account. To disable a user, you need to identify a Disabled Flag, which is an attribute that SiteMinder can use to get and set the disabled state of a user. In the SiteMinder Administrative UI, enter that attribute name in the Disabled Flag field on the User Attributes tab on the User Directory Dialog box.

  When an administrator uses the SiteMinder Administrative UI to disable or enable a user account, or when Password Services disables a user account, the SiteMinder Policy Server calls SmDirSetUserDisabledState().

  In SiteMinder, user accounts can be disabled for a number of reasons, and these reasons are represented by the members of the data structure Sm_Api_DisabledReason_t.

- Sm_DirApi_Capability_SetUserAttributes. SiteMinder can set user attributes in the custom directory.

  The SiteMinder Policy Server calls SmDirSetUserAttr() so that you can use SiteMinder to set a user attribute in your custom directory.

- Sm_DirApi_Capability_Recursive. The custom directory is capable of supporting recursion. For example, the custom directory may support nested groups.

  The following functions in the Directory API have a parameter to hold a recursive flag:

  - SmDirGetUserGroups()
  - SmDirValidateUserPolicyRelationship()

To send information about the directory capabilities to the SiteMinder Policy Server, implement the function SmDirQueryVersion(). Use the capabilities parameter (*pnCapabilites*) to pass one or more values enumerated in Sm_DirApi_Capability_t. SiteMinder then checks for those capabilities.

For example, if a user attempts to change a password, the SiteMinder Policy Server calls SmDirQueryVersion() to check for the capability Sm_DirApi_Capability_ChangeUserPassword. If the custom directory does not have that capability, the user receives an error message.

An example of setting the directory capabilities is shown in the sample code. First, initialize **pnCapabilities* to zero, then set **pnCapabilities* as follows:

```
*pnCapabilities =
   *pnCapabilities | Sm_DirApi_Capability_<supported_capability>;
```

For example:

```
*pnCapabilities =
   *pnCapabilities | Sm_DirApi_Capability_ChangeUserPassword;
*pnCapabilities =
   *pnCapabilities | Sm_DirApi_Capability_DisableUser;
```

Ensure that no other application changes data in fields intended for use by SiteMinder. For example, no other application should change data in the field that holds the disabled state of a SiteMinder user.

Policy Resolutions

Sm_PolicyResolution_t, defined in SmApi.h, enumerates the values that describe the relationship between two policy objects. The following Directory API functions use Sm_PolicyResolution_t:

- SmDirAddEntry()
- SmDirGetDirObjInfo()
- SmDirRemoveEntry()
- SmDirValidateUserPolicyRelationship()

General Data Types and Structures

The data types and structures are used in the Directory API, but may also be used by other SiteMinder APIs.

Sm_Api_DisabledReason_t enumerates the reasons that a user account can be disabled.

The following Directory API functions use Sm_Api_DisabledReason_t:

- SmDirGetUserDisabledState()
- SmDirSetUserDisabledState()

When a user's account is enabled or disabled, the SiteMinder Policy Server calls SmDirSetUserDisabledState(). This call gives you the opportunity to set the disabled flag in your custom directory to one or more of the disabled reasons, as enumerated in Sm_Api_DisabledReason_t. If a user's account is disabled or enabled, SmDirGetUserDisabledState() returns the disabled reason(s). When implementing SmDirGetUserDisabledState(), return Sm_Api_Disabled_Enabled if your custom directory does not support a disabled flag.

Note: A user's account can be disabled for multiple reasons. For example, if the User must change password at next login checkbox is checked and the administrator then clicks Disable, the nDisabledReason holds both the Sm_Api_Disabled_PWMustChange bit and the Sm_Api_Disabled_AdminDisabled bit.

The disabled flag is a SiteMinder user attribute. In the SiteMinder Administrative UI, on the User Attributes tab of the User Directory Dialog box, enter the attribute name in the Disabled Flag field. In the sample, the attribute name is Disabled.

The structure Sm_Api_Context_t gives the function pointers for the SiteMinder logging utility, trace utility, and error utility.

Sm_Api_Reason_t enumerates the reasons for an access event, such as an authentication failure. When a user supplies credentials for authentication, the SiteMinder Policy Server, validating the username and DN, calls SmDirAuthenticateUser(). This call gives you the opportunity to return information about the access event.

Initialization and Release Functions

To initialize objects, the SiteMinder Policy Server calls the functions in the following table:

Function Name	Object Initialized by SiteMinder Policy Server
SmDirInit() (see page 720)	Directory provider. Set provider handle.
SmDirInitDirInstance() (see page 722)	Directory instance. Set directory instance handle.
SmDirInitUserInstance() (see page 725)	Directory Entry (User) instance. Set user entry instance handle.

Initializing the Directory Provider

The first time that the custom directory provider is required after the SiteMinder Policy Server is started, the Policy Server calls SmDirInit() to initialize the directory provider. At this point, set the provider handle as shown in the sample code. The SiteMinder Policy Server will not call SmDirInit() again until one of the Policy Server services is started (or re-started).

SmDirInit() is called once per custom directory provider library (.dll or .so).

Initializing the Directory Instance

The Policy Server calls SmDirInitDirInstance() to initialize the directory instance. Set the directory instance handle as shown in the sample code.

SmDirInitDirInstance() is called once per directory instance using this directory provider library. SiteMinder calls the function when it needs a directory context (to perform an operation such as search or get properties) while processing an authentication or authorization request. This function is typically called at the beginning of a request.

Initializing the Directory Entry (User) Instance

The SiteMinder Policy Server initializes the user instance by calling SmDirInitUserInstance(). Set the directory entry (user) instance handle as shown in the sample code.

To release objects, use the functions in the following table:

Function Name	Object Released by SiteMinder Policy Server
SmDirRelease() (see page 729)	Directory provider. Delete provider handle

Function Name	**Object Released by SiteMinder Policy Server**
SmDirReleaseInstance() (see page 730)	Directory or entry instance. Determine which handle is passed.

Releasing the User Instance

The SiteMinder Policy Server calls SmDirReleaseInstance() so that you can release the user instance handle if you choose. Ensure that the handle that is passed is the user instance handle, not the directory instance handle.

Releasing the Directory Instance

The SiteMinder Policy Server calls SmDirReleaseInstance() so that you can release the directory instance handle if you choose. Ensure that the handle that is passed is the directory instance handle, not the user instance handle.

SiteMinder calls SmDirReleaseInstance() once per every call to SmDirInitDirInstance(), after the directory context is no longer needed. It is typically called at the end of a request.

Releasing the Directory Provider

When an administrator starts to shut down the SiteMinder Policy Server, the SiteMinder Policy Server calls SmDirRelease() to release the directory provider.

More Information:

How To Distinguish between Handle Types (see page 751)

Utility Functions

These functions can be called either within a sequence of directory operations or within a sequence of directory entry (user) operations. If the function receives an instance handle through a parameter, determine whether it is a directory instance handle or a directory entry (user) instance handle.

Function Name	**Description**
SmDirFreeString() (see page 705)	Free memory allocated for a sting.
SmDirFreeStringArray() (see page 705)	Free memory allocated for a string array.

Function Name	Description
SmDirQueryVersion() (see page 728)	Check the Directory API version and the directory capabilities it supports.

Free Strings and String Arrays

After the SiteMinder Policy Server calls an operation function that takes string parameters, the SiteMinder Policy Server calls SmDirFreeString() or SmDirFreeStringArray() to release allocated memory. Calls may be repeated so that multiple strings can be freed.

For example, a SiteMinder Administrator can use the SiteMinder Administrative UI to perform a search for the user Mikel. The SiteMinder Administrator first selects the string User from the Search drop-down list box, then enters the string Mikel in the Search Expression field. SiteMinder calls SmDirLookup() and passes the strings (in the form “User = Mikel”) into the *lpszPattern* parameter. SiteMinder then calls SmDirFreeStringArray() twice. On the first call, SiteMinder passes the string array Mikel. On the second call, SiteMinder passes the string array User.

Query and Validation

The SiteMinder Policy Server frequently calls SmDirQueryVersion(), then SmDirValidateInstance(). This sequence may be repeated several times.

Operations on the Directory

The Policy Server calls the directory operations function(s) to let you define directory operations tasks for your custom directory. For example, if a user is using the SiteMinder Administrative UI to search for a user, the Policy Server calls SmDirLookup().

Function Name	Use this function to:
SmDirAddEntry() (see page 695)	Insert a directory entry (user) into your custom directory.
SmDirAddMemberToGroup() (see page 697)	Add a user or group to an existing group.
SmDirAddMemberToRole() (see page 698)	Assign a role to a user or to a group.
SmDirEnumerate() (see page 704)	Retrieve a list of directory entries and corresponding class names.
SmDirGetDirConnection() (see page 706)	Get the connection handle to the directory.

Function Name	**Use this function to:**
SmDirGetDirObjInfo() (see page 707)	Get information about the object specified in the object parameter.
SmDirGetGroupMembers() (see page 708)	Retrieve the members of a user group.
SmDirGetLastErrMsg() (see page 709)	Determine which instance handle is passed, and return the associated error message. Also used with directory entry (user) operations.
SmDirGetRoleMembers() (see page 710)	Retrieve the directory entries assigned to a role.
SmDirLookup() (see page 727)	Look up a pattern in the directory.
SmDirRemoveEntry() (see page 731)	Delete a directory entry (user) from your custom directory.
SmDirRemoveMemberFromGroup() (see page 732)	Remove a user or group from a existing group.
SmDirRemoveMemberFromRole() (see page 733)	Remove a user or group from an assigned role.
SmDirSearch() (see page 734)	Search on the criteria specified in the search filter parameter.
SmDirSearchCount() (see page 736)	Return a count of the entries that meet the criteria specified in the parameters.
SmDirValidateInstance() (see page 743)	Determine which instance handle is passed, and validate that instance. Also used with directory entry (user) operations.
SmDirValidateUserDN() (see page 744)	Perform any needed validation on the user ID.
SmDirValidateUsername() (see page 745)	Convert the credentials presented by the user to a user ID for the custom directory.

Operations on a Directory Entry (User)

The operations covered in this section apply to directory entries, such as users, groups and roles.

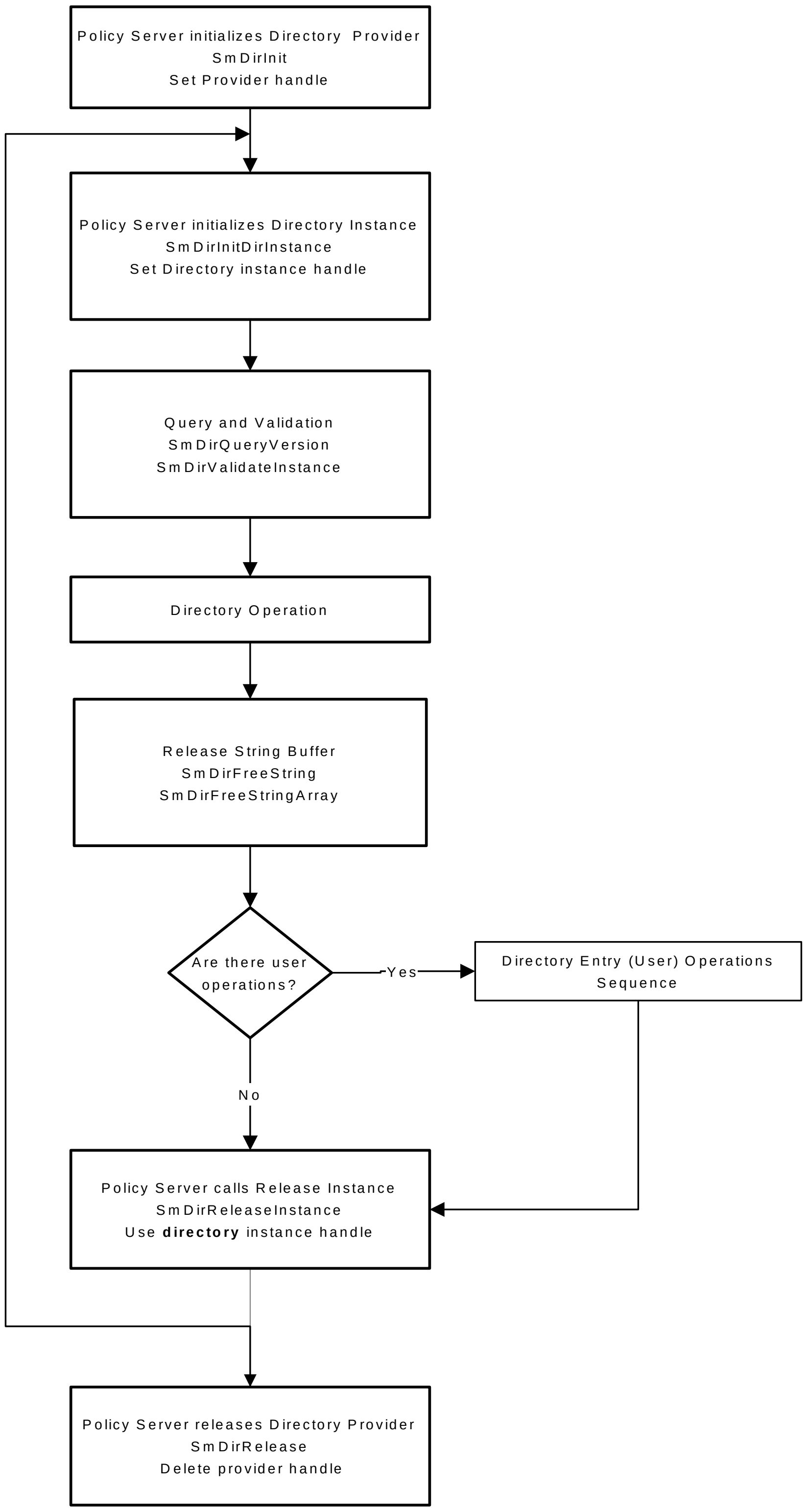
The SiteMinder Policy Server calls the directory entry (user) operation function(s) relevant to the operation performed. For example, if a user is using the SiteMinder Administrative UI to disable a user account, the SiteMinder Policy Server calls SmDirSetUserDisabledState().

Function Name	Use this function to:
SmDirAuthenticateUser() (see page 699)	Check the directory for the provided user name and password.
SmDirChangeUserPassword() (see page 702)	Change the value in the password field for the specified user.
SmDirGetLastErrMsg() (see page 709)	Determine which instance handle is passed, and return the associated error message. Also used with directory operations.
SmDirGetUserAttr() (see page 711)	Retrieve the value of the specified user attribute.
SmDirGetUserAttrMulti() (see page 712)	Retrieve an array of values for a single attribute.
SmDirGetUserClasses() (see page 713)	Get the object classes for the specified DN.
SmDirGetUserDisabledState() (see page 714)	For disabled user accounts, return the reason that an account is disabled. Otherwise, return enabled.
SmDirGetUserGroups() (see page 717)	Retrieve the groups to which a user belongs.
SmDirGetUserProperties() (see page 718)	Return the names of all user attributes or only required user attributes.
SmDirGetUserRoles() (see page 719)	Retrieve the roles to which a user belongs.
SmDirSetUserAttr() (see page 738)	Set the value for a user attribute.
SmDirSetUserAttrMulti() (see page 739)	Set an array of values for a single attribute.
SmDirSetUserDisabledState() (see page 740)	Enable or disable a user account.

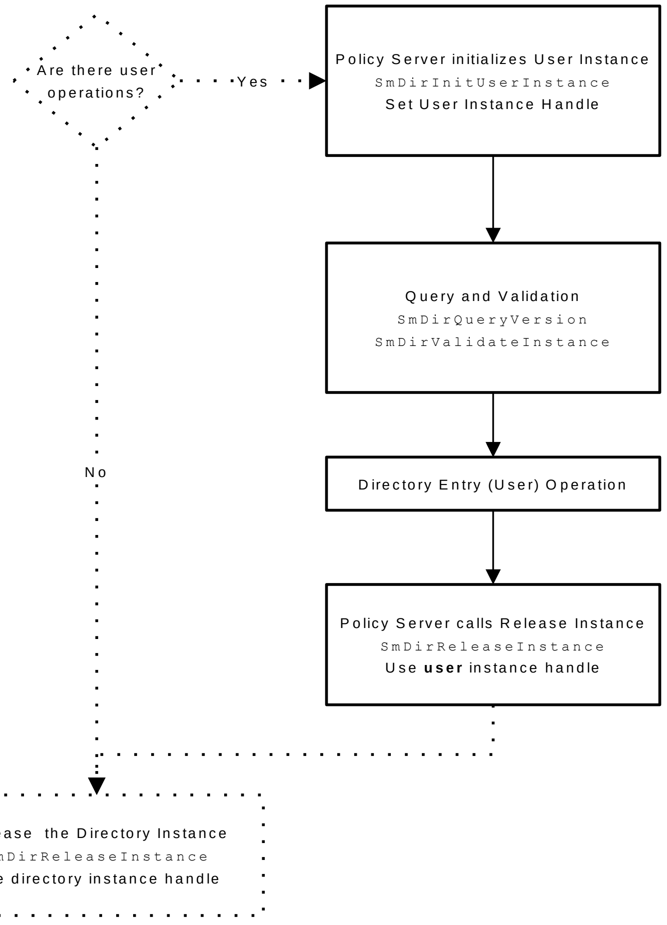
Function Name	Use this function to:
SmDirValidateInstance() (see page 743)	Determine which instance handle is passed, and validate that instance. Also used with directory operations.
SmDirValidateUserPolicy (see page 746) Relationship() (see page 746)	Validate the relationship between policy objects.

Directory API Reference

The following diagrams outline the order of function calls for procedures that perform operations on the directory and possibly perform operations on a directory entry (user). For example, using the SiteMinder Administrative UI to search for a particular user in a custom directory requires both operations on the directory and operations on a directory entry (user). Some procedures involve only operations on the directory. For example, using the SiteMinder Administrative UI only to view the properties of a custom user directory requires only operations on the directory.


Policy Server initializes Directory Provider
SmDirInit
Set Provider handle
Policy Server initializes Directory Instance
SmDirInitDirInstance
Set Directory instance handle
Query and Validation
SmDirQueryVersion
SmDirValidateInstance
Directory Operation
Release String Buffer
SmDirFreeString
SmDirFreeStringArray
Are there user operations?
Yes
Directory Entry (User) Operations Sequence
No
Policy Server calls Release Instance
SmDirReleaseInstance
Use directory instance handle
Policy Server releases Directory Provider
SmDirRelease
Delete provider handle

This second diagram shows an additional sequence that occurs only if directory entry (user) operations occur.


Are there user operations?
Yes
No
Policy Server initializes User Instance
SmDirInitUserInstance
Set User Instance Handle
Query and Validation
SmDirQueryVersion
SmDirValidateInstance
Directory Entry (User) Operation
Policy Server calls Release Instance
SmDirReleaseInstance
Use user instance handle
Release the Directory Instance
SmDirReleaseInstance
Use directory instance handle

To authenticate a user, the SiteMinder Policy Server requests a username from the user. SmDirValidateUsername() is called to translate the user-supplied username into the internal user ID key used by the directory as the primary key to the user's data. The username from the credentials is supplied in the *lpszUsername* parameter. If SmDirValidateUsername() is not implemented, the user-supplied username is passed into *lpszUserDN*.

If SmDirValidateUsername() is implemented, it should return the user's ID in the *lpszNewUsername* parameter. The value returned by *lpszNewUsername* becomes the *lpszUserDN* parameter value.

The *lpszUserDN* parameter value is passed into many other functions, such as SmDirValidateUserDN() and SmDirAuthenticateUser().

SmDirAddEntry()

The SiteMinder Policy Server calls SmDirAddEntry() so that you can insert a directory entry (user) into your custom directory. Examples of directory entries are users, groups and roles. For example, if you are using a SQL database and need to add a group, you could use SmDirAddEntry() to insert a record into the groups table (and all related tables) for the database.

When adding an entry to a hierarchical directory, it may be helpful to look at the attributes passed in with the entry, such as object class in LDAP.

Syntax

```
int SM_EXTERN SmDirAddEntry (
   const Sm_Api_Context_t*      lpApiContext,
   void*                        pHandle,
   void*                        pInstanceHandle,
   const Sm_PolicyResolution_t  nEntryType,
   const char*                  lpszEntryDN,
   const char**                 lpszAttrNames,
   const char**                 lpszAttrValues
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.

Parameter	I/O	Description
nEntryType	I	The Policy resolution of the entry. Policy resolutions are enumerated in Sm_PolicyResolution_t, which is defined in SmApi.h. The following elements of Sm_PolicyResolution_t are valid entry types: ■ Sm_PolicyResolution_Unknown ■ Sm_PolicyResolution_User ■ Sm_PolicyResolution_UserGroup ■ Sm_PolicyResolution_UserRole ■ Sm_PolicyResolution_Org
lpszEntryDN	I	Buffer containing the distinguished name for the entry being added.
lpszAttrNames	I	Buffer containing the names of the entry attributes.
lpszAttrValues	I	Buffer containing the values of the entry attributes.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

This function is called when Delegated Management Services is used to create directory entries, including users or roles.

SmDirAddMemberToGroup()

The SiteMinder Policy Server calls SmDirAddMemberToGroup() so that you can add a user or group to an existing group.

If you want to add a user or group to a role, use SmDirAddMemberToRole(). The difference between a group and a role is defined by your custom directory provider. For some providers, there is no difference.

Syntax

```
int SM_EXTERN SmDirAddMemberToGroup (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszMemberDN,
   const char*              lpszGroupDN
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.
lpszMemberDN	I	Buffer containing the distinguished name for the user or group being added to the existing group.
lpszGroupDN	I	Buffer containing the distinguished name for the group to which the member is being added.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

This function is called when Delegated Management Services is used to assign either a user or a group to an existing group.

SmDirAddMemberToRole()

The SiteMinder Policy Server calls SmDirAddMemberToRole() so that you can assign a role to a user or to a group.

For example, in Oracle, a role is a set of object or system privileges that can be granted to a user. A group is a set of users. If you want everyone that performs collections to be able to update the AR table and select from the CUSTOMER table, you could create a role named COLLECTIONS. You could then assign the COLLECTIONS role to each of the individual users who perform collections, or even to a group such as Accounts Receivable.

If you want to add either a user or group to an existing group, use SmDirAddMemberToGroup(). The difference between a group and a role is defined by the provider of the custom directory.

For some providers, there will be no difference between a role and a group.

Syntax

```
int SM_EXTERN SmDirAddMemberToRole (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszMemberDN,
   const char*              lpszRoleDN
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.
lpszMemberDN	I	Buffer containing the distinguished name for the user or group being added to the existing role.
lpszRoleDN	I	Buffer containing the distinguished name for the role to which the member is being added.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

This function is called when Delegated Management Services is used to assign a role to a user or a group.

SmDirAuthenticateUser()

Use the SmDirAuthenticateUser() function to check the directory for the provided user name and password.

After the call to SmDirAuthenticateUser(), SiteMinder calls SmDirFreeString() to free the *lpszUserMsg* buffer, then calls SmDirFreeString() again to free the *lpszErrMsg* buffer.

Syntax

```
int SM_EXTERN SmDirAuthenticateUser (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUserDN,
   const char*              lpszPassword,
   Sm_Api_Reason_t*         pnReason,
   char**                   lpszUserMsg,
   char**                   lpszErrMsg
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
Handle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.

Parameter	I/O	Description
lpszUserDN	I	Buffer containing the user DN that has to be authenticated. If SmDirValidateUsername() is not implemented, the user-supplied username is passed into *lpszUserDN*. If SmDirValidateUsername() is implemented, SmDirValidateUsername() should return the user's ID in the *lpszNewUsername* parameter. The value returned by *lpszNewUsername* becomes the *lpszUserDN* parameter value.
lpszPassword	I	Buffer containing the password that has to be authenticated.
pnReason	O	Pointer to the resulting reason of the authentication event, using the reasons enumerated in Sm_Api_Reason_t.
lpszUserMsg	O	Output buffer to receive a message for the user. This message can be the challenge text or any other message an authentication scheme developer wants to present to the user through a mechanism external to SiteMinder. In the sample, if a bogus username is presented, authentication fails and the string Failed to authenticate is copied to *lpszUserMsg*. The Web Agent stores this message in the HTTP variable HTTP_SM_USERMSG. For RADIUS authentication, the user message is returned in the REPLY-MESSAGE response attribute. The SiteMinder Policy Server writes the error message in *lpszUserMsg* to the SiteMinder Authentication log.

Parameter	I/O	Description
lpszErrMsg	O	Output buffer to receive the error message. Use this buffer to return an error message to SiteMinder. In the sample, if a bogus username is presented, authentication fails and the string Failed to authenticate is copied to *lpszErrMsg*. The SiteMinder Policy Server writes the error message in *lpszErrMsg* to the SiteMinder Authentication log. The string in *lpszErrMsg* follows Not Authenticated in the log. The string in *lpszUserMsg* follows the string in *lpszErrMsg*. For example, if the challenged user presents the bogus username impostor, the SiteMinder Policy Server writes the following status message to the log: 'impostor' Not Authenticated. ErrMsg. UserMsg

Returns

Returns 0 if authentication succeeds, or -1 if there is an error in processing or if the user-supplied credentials are invalid.

If authentication fails, convey the reason through the output parameter *pnReason* and return -1.

Remarks

This function is called when you use the SiteMinder Test Tool to run IsAuthenticated for a user in the custom directory.

SmDirChangeUserPassword()

The SiteMinder Policy Server calls SmDirChangeUserPassword() so that you can change the value in the password field for an entry in your custom directory.

To implement SmDirChangeUserPassword(), you must specify the name of the password field in your custom directory. In the SiteMinder Administrative UI, enter that attribute name in the Password Attribute field on the User Attributes tab of the User Directory Dialog box.

The user’s distinguished name is passed in with the following information:

- Old password
- New password
- Name of the password field in the custom directory
- Whether the old password is needed

Syntax

```
int SM_EXTERN SmDirChangeUserPassword (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUserDN,
   const char*              lpszOldPassword,
   const char*              lpszNewPassword,
   const char*              lpszPasswordAttr,
   const int                bDoNotRequireOldPassword
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.
lpszUserDN	I	Buffer containing the user DN whose password has to be changed.
lpszOldPassword	I	Buffer containing the old password of the user DN.

Parameter	I/O	Description
lpszNewPassword	I	Buffer containing the new password of the user DN.
lpszPasswordAttr	I	Directory attribute where the user's password is stored. Use this attribute to change the user password. The default value for this attribute name in a Netscape LDAP directory is *userpassword*.
bDoNotRequireOldPassword	I	A value indicating whether the user needs to specify the old password to perform the password change. The value 1 indicates that the old password is not required; 0 indicates that the old password is required. An administrator may not need to specify the old password, but an end user would need to specify.

Returns

Returns 0 if successful or -1 if not successful.

Sample Code Information

If you are changing a user password, SiteMinder passes the name of the password attribute through *lpszPasswordAttr*. To indicate the name of the directory attribute that holds this information:

1. In the SiteMinder Administrative UI, go to the User Directory Dialog box
2. In the User Attributes tab, complete the Password Attribute field. To use the sample, type password in this field.

SmDirEnumerate()

The SiteMinder Policy Server calls SmDirEnumerate() to retrieve a list of distinguished names and their corresponding class names (User or Group) in the user directory.

The SiteMinder Policy Server calls SmDirFreeStringArray() to free the *lpszDNs* and *lpszClasses* arrays.

Syntax

```
int SM_EXTERN SmDirEnumerate (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   char***                  lpszDNs,
   char***                  lpszClasses
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.
lpszDNs	O	List of user distinguished names present in the directory.
lpszClasses	O	List of corresponding class information for the user distinguished names.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

This function is called when you click the View Contents button in the SiteMinder User Directory Dialog box.

SmDirFreeString()

The SiteMinder Policy Server calls SmDirFreeString() to free memory allocated for the specified string in *lpszString*.

Syntax

```
void SM_EXTERN SmDirFreeString (
   char* lpszString
);
```

Parameter	I/O	Description
lpszString	I	String for which the allocated memory is to be freed.

SmDirFreeStringArray()

The SiteMinder Policy Server calls SmDirFreeStringArray() to free memory allocated for the specified string array in *lpszStringArray*. For example, after a call to SmDirEnumerate(), the SiteMinder Policy Server calls SmDirFreeStringArray() to free memory allocated for parameter *lpszDNs*, then calls SmDirFreeStringArray() again to free memory allocated for parameter *lpszClasses*.

Syntax

```
void SM_EXTERN SmDirFreeStringArray (
   char* lpszStringArray
);
```

Parameter	I/O	Description
lpszStringArray	I	String array for which the allocated memory is to be freed.

SmDirGetDirConnection()

This function is called to get the connection handle to the directory.

If you are using active rules, policies, or responses, the SiteMinder Policy Server calls SmDirGetDirConnection() when it calls an authentication scheme library or an active library. The SiteMinder Policy Server calls SmDirGetDirConnection() before authentication.

When implementing SmDirGetDirConnection(), return NULL.

Syntax

```
void* SM_EXTERN SmDirGetDirConnection (
   const Sm_Api_Context_t*   lpApiContext,
   void*                     pHandle,
   void*                     pInstanceHandle
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.

Returns

When implementing SmDirGetDirConnection(), return NULL, as shown in the sample.

SmDirGetDirObjInfo()

The SiteMinder Policy Server calls SmDirGetDirObjInfo() to get information about the object specified in the *lpszObject* buffer. You can use this function to get the following information about the object passed in as *lpszObject*:

- Distinguished Name (*lpszDN*)
- Class (*lpszClass*)
- Policy Resolution (*pnSmPolicyResolution*)

The SiteMinder Policy Server calls SmDirFreeString() to free the *lpszDN* and *lpszClass* buffers.

Syntax

```
int SM_EXTERN SmDirGetDirObjInfo (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszObject,
   char**                   lpszDN,
   char**                   lpszClass,
   int*                     pnSmPolicyResolution
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.
lpszObject	I	The string containing the null-terminated string of the user name.
lpszDN	O	Fill in the user distinguished name resolved from the *lpszObject*.
lpszClass	O	Fill in the class of the resolved user distinguished name.
pnSmPolicyResolution	O	Fill in the Policy resolution of the resolved user distinguished name. Policy resolutions are enumerated in Sm_PolicyResolution_t.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

This function is called in the following circumstances:

- Searching for a user
- Enabling or disabling a user account

Sample Code Information

If you are disabling user account Mikel, Mikel is passed in as *lpszObject*. Using the sample code, the following values are returned:

Parameter	Value
lpszDN	Mikel
lpszClass	User
pnSmPolicyResolution	1

SmDirGetGroupMembers()

The SiteMinder Policy Server calls SmDirGetGroupMembers() so that you can retrieve the members of a user group. This function is designed to support Delegated Management Services.

Syntax

```
int SM_EXTERN SmDirGetGroupMembers (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszGroupDN,
   char***                  lpszMembers,
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().

Parameter	I/O	Description
pInstanceHandle	I	The address of a pointer to the directory instance handle.
lpszGroupDN	I	Buffer containing the distinguished name for the group for which members are being retrieved.
lpszMembers	O	Buffer containing the members of the group.

Returns

Returns 0 if successful or -1 if not successful.

SmDirGetLastErrMsg()

Calling the function SmDirGetLastErrMsg() retrieves the last error message that resulted from a Directory API call.

If a user operation or directory operation fails, store the error message in *szErrMsg*, which is the third field of the instance handle structure. You can then call SmDirGetLastErrMsg() to retrieve the error message. The SiteMinder Policy Server also calls SmDirGetLastErrMsg() upon operation failure.

This function call is made for both the directory instance and the user instance. Either the directory instance handle or user instance handle can be passed through *pInstanceHandle*. Your code must determine which handle is passed and return the appropriate error message. See the sample code for an example.

After calling SmDirGetLastErrMsg(), the SiteMinder Policy Server calls SmDirFreeString() to free the error buffer.

Syntax

```
char* SM_EXTERN SmDirGetLastErrMsg (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().

Parameter	I/O	Description
pInstanceHandle	I	The addresses of a pointer to the user instance handle or directory instance handle.

Returns

The last error message occurred during a directory operation or user operation.

Remarks

This function is called when you are searching for users or groups and the lookup (SmDirLookup()) fails.

Sample Code Information

In the sample code for SmDirValidateUserPolicyRelationship(), under the condition that the policy resolution is UserGroup, there is a call to the function chk_grp(). The third parameter of chk_grp() is an output parameter that returns *szErrMsg* in the user handle. If you then called SmDirGetLastErrMsg() and passed the user instance handle, you would get the value stored in *szErrMsg*.

SmDirGetRoleMembers()

The SiteMinder Policy Server calls SmDirGetRoleMembers() so that you can retrieve the directory entries assigned to a role. This function is designed to support Delegated Management Services.

Syntax

```
int SM_EXTERN SmDirGetRoleMembers (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszRoleDN,
   char***                  lpszMembers
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.

Parameter	I/O	Description
lpszRoleDN	I	Buffer containing the distinguished name for the role for which members are being retrieved.
lpszMembers	O	Buffer containing the members of the role.

Returns

Returns 0 if successful or -1 if not successful.

SmDirGetUserAttr()

The SiteMinder Policy Server calls SmDirGetUserAttr() so that you can retrieve the value for a user attribute in your custom directory. For example, you may need to retrieve the last name of a user.

SiteMinder calls SmDirFreeString() to free the *lpszAttrData* buffer.

Syntax

```
int SM_EXTERN SmDirGetUserAttr (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUserDN,
   const char*              lpszAttrName,
   char**                   lpszAttrData
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.
lpszUserDN	I	Buffer containing the user DN whose user attribute has to be retrieved.
lpszAttrName	I	Buffer containing the name of the user attribute whose value you’re retrieving.

Parameter	I/O	Description
lpszAttrData	O	Buffer containing the retrieved value for the requested attribute. Look up the value of the attribute specified in *lpszAttrName* and return the value in *lpszAttrData*.

Returns

Returns 0 if successful or -1 if not successful.

SmDirGetUserAttrMulti()

The SiteMinder Policy Server calls SmDirGetUserAttrMulti() so that you can retrieve an array of values for a single attribute. The provider needs to handle the case where this function is called and the attribute has only a single value.

SiteMinder calls SmDirFreeStringArray() to free the *lpszAttrData* buffer.

Syntax

```
int SM_EXTERN SmDirGetUserAttrMulti (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUserDN,
   const char*              lpszAttrName,
   char***                  lpszAttrData
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.
lpszUserDN	I	Buffer containing the user DN whose user attributes has to be retrieved.
lpszAttrName	I	Buffer containing the name of the user attribute.

Parameter	I/O	Description
lpszAttrData	O	Buffer containing the value of the user attribute. Look up the value of the attribute named in *lpszAttrName* and return the value in *lpszAttrData*.

Returns

Returns 0 if successful or -1 if not successful.

SmDirGetUserClasses()

The SiteMinder Policy Server calls SmDirGetUserClasses() so that you can get the object classes of the specified distinguished name (DN). This function is designed to support Delegated Management Services.

Your custom directory may be hierarchical or flat. If your directory is hierarchical, as with an LDAP directory, the DN may belong to multiple object classes. If the directory is flat, as with a SQL database, the user DN belongs to a single class, such as User or Group.

Your code must determine the type of DN passed and handle it appropriately. For example, if a the name of a group is passed in, you need to be able to determine that it is a group and return Group.

SiteMinder calls SmDirFreeStringArray() to free the *lpszAttrData* buffer.

Syntax

```
int SM_EXTERN SmDirGetUserClasses (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUserDN,
   char***                  lpszClasses
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.

Parameter	I/O	Description
lpszUserDN	I	Buffer containing the distinguished name for which you must retrieve the classes.
lpszClasses	O	Buffer containing the classes for the specified distinguished name.

Returns

Returns 0 if successful or -1 if not successful.

If you decide not to implement this function, return -1.

SmDirGetUserDisabledState()

The SiteMinder Policy Server calls SmDirGetUserDisabledState() to get information about whether a user account is disabled.

If a user account is disabled, SmDirGetUserDisabledState() returns information in *pnDisabledReason* about how the account was disabled.

The possible reasons that a user is disabled are enumerated in Sm_Api_DisabledReason_t, which is defined in SmApi.h. The disabled reason is set when the SiteMinder Policy Server calls SmDirSetUserDisabledState().

If the custom directory does not support a disabled flag, use the following code to indicate that the user is always enabled:

```
*pnDisabledReason = Sm_Api_Disabled_Enabled;
return 0;
```

Syntax

```
int SM_EXTERN SmDirGetUserDisabledState (
   const Sm_Api_Context_t*   lpApiContext,
   void*                     pHandle,
   void*                     pInstanceHandle,
   const char*               lpszUserDN,
   const char*               lpszDisabledAttr,
   Sm_Api_DisabledReason_t*  pnDisabledReason
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.

Parameter	I/O	Description
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.
pszUserDN	I	Buffer containing the user DN whose disabled state has to be retrieved.
lpszDisabledAttr	I	The user directory attribute that is used by SiteMinder to track disabled users. Use this attribute to retrieve the disabled state.
pnDisabledReason	O	Attribute containing the user disabled state of the user. Store the user-disabled state that is fetched from the user directory attribute specified in *lpszDisabledAttr*.

Returns

Return values indicate whether the function successfully determines the user's disabled state:

- Return 0 if the function successfully determines the user's disabled state.
- Return -1 if the function fails to determine the user's disabled state.

Return values do not indicate the disabled state, which is stored in the parameter *pnDisabledReason*.

Remarks

This function is called in the following circumstances:

- When searching for a user, after calling SmDirLookup() to perform the lookup.
- After calling SmDirSetUserDisabledState() to change the disabled state of a user.
- After calling SmDirAuthenticateUser() to authenticate a user. If the disabled flag is set for a user, SiteMinder does not authenticate the user—even if the user supplies valid credentials.

Sample Code Information

If you are managing users, when you lookup the user, SiteMinder checks the disabled state for that user by checking *lpszDisabledAttr*. To indicate the name of the directory attribute that holds this information, in the SiteMinder Administrative UI, in the User Directory Dialog box, on the User Attributes tab, complete the Disabled Flag field. To use the sample, type Disabled in this field.

If the user is enabled, then running the sample code results in a value of Sm_Api_Disabled_Enabled for *pnDisabledReason*. If the SiteMinder administrator has used the Disable button to disable the user, then running the sample code results in a value of Sm_Api_Disabled_AdminDisabled for *pnDisabledReason*.

SmDirGetUserGroups()

The SiteMinder Policy Server calls SmDirGetUserGroups so that you can retrieve the groups to which a user belongs.

SiteMinder calls SmDirFreeStringArray() to free the *lpszGroups* array.

Syntax

```
int SM_EXTERN SmDirGetUserGroups (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUserDN,
   const int                bRecursive,
   char***                  lpszGroups
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.
lpszUserDN	I	Buffer containing the user DN.
bRecursive	I	A value specifying whether the custom directory supports recursion (for example, nested groups). The value 1 indicates recursion support; 0 indicates no support. If your custom directory supports recursion, you must search down any hierarchy of groups to find the user. Suppose that the value of the User DN is Bill Collector. Bill Collector may be in a group called AR, and AR may be in a group called Accounting.
lpszGroups	O	A list of groups associated with *lpszUserDN*.

Returns

Returns 0 if successful or -1 if not successful.

SmDirGetUserProperties()

SiteMinder calls SmDirGetUserProperties() so that you can return the names of all user attributes or only required user attributes.

Unlike SmDirGetUserAttr(), which is designed to return values for the attribute names passed, SmDirGetUserProperties() is designed for returning only attribute names.

SiteMinder calls SmDirFreeStringArray() to free the lpszProperties array.

Syntax

```
int SM_EXTERN SmDirGetUserProperties (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUserDN,
   const int                bMandatory,
   char***                  lpszProperties
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.
lpszUserDN	I	Buffer containing the user DN for which you must retrieve the names of user attributes.
bMandatory	I	A value specifying whether to return only mandatory attribute names. The value 1 indicates that only mandatory attribute names are returned; 0 indicates that all attribute names are returned. Some directories have interfaces that distinguish between the mandatory attributes of an object and the optional attributes. For example, IADs for Microsoft Active Directory makes this distinction.
pszProperties	O	Array containing a list of user attribute names.

Returns

Returns 0 if successful or -1 if not successful.

SmDirGetUserRoles()

The SiteMinder Policy Server calls SmDirGetUserRoles() so that you can retrieve the roles to which a user belongs.

SiteMinder calls SmDirFreeStringArray() to free the *lpszRoles* array.

Syntax

```
int SM_EXTERN SmDirGetUserRoles (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUserDN,
   char***                  lpszRoles
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.
lpszUserDN	I	Buffer containing the user DN whose roles has to be retrieved.
lpszRoles	O	A list of roles associated with the user in *lpszUserDN*.

Returns

Returns 0 if successful or -1 if not successful.

SmDirInit()

SmDirInit() is called when SiteMinder initializes the directory services provider for the custom namespace.

This function is called once before any other Directory API functions are called. The function returns the address of a pointer to the handle for the directory. The handle is passed in all subsequent function calls.

Once the administration process starts, the SiteMinder Policy Server calls SmDirInit() the first time you perform any task for which the custom directory provider is required.

For example, if you were to start the Policy Server, then immediately view the properties of a user directory with a Custom namespace, the SiteMinder Policy Server would call SmDirInit(), then call SmDirInitDirInstance().

Syntax

```
int SM_EXTERN SmDirInit (
   const Sm_Api_Context_t*  lpApiContext,
   void**                   ppHandle,
   const char*              lpszParameter
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
ppHandle	O	The address of a pointer to the handle for the Directory API. This parameter is initialized in the call to SmDirInit() and is passed to all subsequent function calls. For example, if you were using SmDirInit() to load a shared library on the directory side, you could use *ppHandle* to store function pointers to all the functions in that shared library.
lpszParameter	I	The null-terminated string specified in the Parameter field of the SiteMinder User Directory Dialog box.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

Instantiate ProviderHandle_t when SmDirInit() is called. This handle is then passed to almost all subsequent functions. The same value need not be carried through the entire process. You are permitted to change the value.

Release ProviderHandle_t when SmDirRelease() is called.

You could use SmDirInit() to load another shared library. The vendor of the directory containing your data may provide an interface that you can implement by building a shared library. You could use SmDirInit() to load that shared library by placing the path to the shared library in the Parameter field on the Directory Setup tab of the User Directory Dialog box. The string entered in the Parameter field is passed to *lpszParameter* in calls to SmDirInit() and SmDirInitDirInstance().

Note: The string entered in the Parameter field is also passed to *lpszSearchRoot* in calls to SmDirSearch() and SmDirSearchCount(). If your code for SmDirInit() needs to use *lpszParameter* and your code for the search functions needs a search root, you will have to parse the string from the Parameter field.

SmDirInitDirInstance()

SiteMinder calls this directory instance initialization function before it calls any of the directory functions on the given directory instance. This function provides you an opportunity to make a connection to the custom directory.

After SiteMinder completes the directory function call, it calls SmDirReleaseInstance(). This function is called multiple times. SiteMinder does not require you to make and break directory connections every time the initialization and release calls are made.

Syntax

```
int SM_EXTERN SmDirInitDirInstance (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void**                   ppInstanceHandle,
   const char*              lpszUniqueKey,
   const char*              lpszParameter,
   const char*              lpszUsername,
   const char*              lpszPassword,
   const int                bRequireCredentials,
   const int                bSecureConnection,
   const int                nSearchResults,
   const int                nSearchTimeout
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
ppInstanceHandle	O	Use this parameter to set the handle for the directory instance. For example, you could use *ppInstanceHandle* to hold information about a connection to the directory. At the end of the directory instance lifecycle, the SiteMinder Policy Server calls SmDirReleaseInstance() and passes the directory instance handle so that you can delete the handle.
lpszUniqueKey	I	A unique identifier for the directory instance session. This unique key holds the object identifier (OID) of the custom directory object.

Parameter	I/O	Description
lpszParameter	I	The null-terminated string specified in the Parameter field of the SiteMinder User Directory Dialog box. The value is the same as it was for the call to SmDirInit().
lpszUsername	I	The string containing the user name as specified in the Credentials and Connection tab of the SiteMinder User Directory Dialog box. When *bRequireCredentials* is set to true, *lpszUsername* holds the value in the Username field in the Administrator Credentials group box.
lpszPassword	I	The string containing the password as specified in the Credentials and Connection tab of the SiteMinder User Directory Dialog box. When *bRequireCredentials* is set to true, *lpszPassword* holds the value in the Password field in the Administrator Credentials group box.
bRequireCredentials	I	This boolean indicates whether credentials are required for user directory access. In the SiteMinder Administrative UI, on the Credentials and Connection tab, in the Administrator Credentials group box, there is a Require Credentials check box. Checking this check box sets *bRequireCredentials* to true (1). When *bRequireCredentials* is set to true, *lpszUsername* and *lpszPassword* will hold the values in the Username and Password fields in the Administrator Credentials group box. The SiteMinder Policy Server uses these credentials to access the directory.
bSecureConnection	I	This boolean indicates whether an SSL connection is required to access the user directory.

Parameter	I/O	Description
nSearchResults	I	This parameter indicates the maximum number of records to return as the result set of a single search by the Directory API. In the SiteMinder User Directory Dialog box, on the Directory Setup tab, in the Custom NameSpace group box, there is a Max results field. The *nSearchResults* parameter holds the value in the Max results field.
nSearchTimeout	I	This parameter indicates the time in seconds after which the Directory API will stop searching the user directory for results. In the SiteMinder User Directory Dialog box, on the Directory Setup tab, in the Custom NameSpace group box, there is a Max time field. The *nSearchResults* parameter holds the value in the Max time field.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

Instantiate DirHandle_t when SmDirInitDirInstance() is called. Set *nTag* to 0 to distinguish the directory instance handle from the user instance handle.

The handle referenced by DirHandle_t is passed to subsequent directory operations functions. You can change the handle value.

Release DirHandle_t when SmDirReleaseInstance() is called. Use the value of *nTag* to distinguish between the directory instance handle and the user instance handle.

More Information:

Operations on the Directory (see page 689)

SmDirInitUserInstance()

The SiteMinder Policy Server calls SmDirInitUserInstance() before it calls any of the directory entry (user) operations functions on the given directory instance.

SmDirInitUserInstance() can be called multiple times.

Syntax

```
int SM_EXTERN SmDirInitUserInstance (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void**                   ppInstanceHandle,
   void*                    pDirInstanceHandle,
   const char*              lpszUserDN
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
ppInstanceHandle	O	The address of a pointer to hold the handle for the user instance session.
pDirInstanceHandle	I	The address of a pointer handle for the directory instance session. This value is passed in from SmDirInitDirInstance().
lpszUserDN	I	The string containing the null-terminated string of the user distinguished name.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

Instantiate   UserHandle_t when SmDirInitUserInstance() is called. Set *nTag* to 1 to distinguish the user instance handle from the directory instance handle.

The handle referenced by UserHandle_t is passed to subsequent directory entry (user) operations functions. You can change the handle value.

After SiteMinder completes the calls to the relevant directory entry (user) operations functions, it calls SmDirReleaseInstance(). Release UserHandle_t when this call is made. Use the value of *nTag* to distinguish between the user instance handle and the directory instance handle.

SmDirLookup()

SiteMinder calls SmDirLookup() to look up a pattern in the directory.

Use the following search expression grammar for the search pattern:

[<*class*> =] <*value*>

In the search pattern format:

- <*class*> = empty-string | user | group (empty-string implies user & group)
- <*value*> = wildcard-string

Your code must be able to interpret a search pattern that begins with the string user=. For example:

```
...
CStringArray Paths, Classes;
CString szPattern = CString ("user=") + UserDir.m_szUniversalIDAttr
   + CString ("=") + Session.m_szUnivId;
if (m_pDsDir->Lookup (szPattern, Paths, Classes))
....
```

SiteMinder calls SmDirFreeStringArray() to free the *lpszDNs* and *lpszClasses* arrays.

Syntax

```
int SM_EXTERN SmDirLookup (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszPattern,
   char***                  lpszDNs,
   char***                  lpszClasses
);
```

Parameter	**I/O**	**Description**
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.
lpszPattern	I	Buffer containing the pattern to search in the directory.

Parameter	I/O	Description
lpszDNs	O	List of user distinguished names that match the pattern.
lpszClasses	O	List of class information corresponding to the user distinguished names.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

This function is called when you perform a user directory search.

Sample Code Information

When trying the sample code, go to the SiteMinder Administrative UI, navigate to the dialog box for the directory search, and enter the search criteria as follows:

- In the Search drop-down list box, select User.
- In the Search Expression field, enter the exact name of one of the sample users, such as Mikel. Do not use wild cards.

SmDirQueryVersion()

SmDirQueryVersion() queries the Directory API to find out its version and its directory capabilities. Supported capabilities are enumerated in Sm_DirApi_Capability_t.

Syntax

```
int SM_EXTERN SmDirQueryVersion (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   unsigned long*           pnCapabilities
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().

Parameter	I/O	Description
pnCapabilities	O	This parameter is used to pass the capabilities of the custom directory. The capabilities of a custom directory are enumerated in Sm_DirApi_Capability_t, which is defined in SmApi.h. For more information on Sm_DirApi_Capability_t, see Directory Capabilities.

Returns

Returns the version number of the Directory API that the custom library complies with. Currently the versions supported are Sm_Api_Version_V4 and Sm_Api_Version_V4_1. Version constants are defined in SmApi.h.

SmDirRelease()

The Policy Server calls SmDirRelease() when the directory service provider for the Custom Namespace is no longer required.

Syntax

```
void SM_EXTERN SmDirRelease (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().

Remarks

This function is called when you stop one of the Policy Server services.

SmDirReleaseInstance()

The SiteMinder Policy Server calls SmDirReleaseInstance() so that you can release the user instance or the directory instance if you choose.

Syntax

```
void SM_EXTERN SmDirReleaseInstance (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user or directory instance handle. To release the user instance, ensure that *pInstanceHandle* contains the user instance handle. To release the directory instance, ensure that *pInstanceHandle* contains the directory instance handle.

Remarks

You need to write your release code so that it will work with the type of instance handle— user or directory—that is passed.

During a process, the SiteMinder Policy Server may first call SmDirInitDirInstance() and then call SmDirInitUserInstance(). If so, the SiteMinder Policy Server calls SmDirReleaseInstance() twice at the end of the process. At the first call, the user instance handle (as *pInstanceHandle*) is passed so that you can release the user instance. At the second call, the directory instance handle is passed (as *pInstanceHandle*) so that you can release the directory instance.

Use the value of *nTag* to distinguish between a user handle and a directory handle, as follows:

- With user handles, *nTag* in structure DirHandle_t is 0.
- With directory handles, *nTag* in structure UserHandle_t is 1.

SmDirRemoveEntry()

The SiteMinder Policy Server calls SmDirRemoveEntry() so that you can delete a directory entry (user) from your custom directory.

Examples of directory entries are users, groups and roles. For example, if you are using an SQL database and need to remove a group, you could use SmDirRemoveEntry() to delete the relevant record from the groups table (and all related tables) for the database.

If your directory is hierarchical, as with an LDAP directory, you need to handle the process of deleting relevant data at different levels of the hierarchy. It may be helpful to look at the attributes of the entry, such as object class in LDAP.

Syntax

```
int SM_EXTERN SmDirRemoveEntry (
   const Sm_Api_Context_t*      lpApiContext,
   void*                        pHandle,
   void*                        pInstanceHandle,
   const Sm_PolicyResolution_t  nEntryType,
   const char*                  lpszEntryDN
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.
nEntryType	I	The Policy resolution of the entry. Policy resolutions are enumerated in Sm_PolicyResolution_t, which is defined in SmApi.h. The following Sm_PolicyResolution_t elements are valid entry types: ■ Sm_PolicyResolution_Unknown ■ Sm_PolicyResolution_User ■ Sm_PolicyResolution_UserGroup ■ Sm_PolicyResolution_UserRole ■ Sm_PolicyResolution_Org

Parameter	I/O	Description
lpszEntryDN	I	Buffer containing the distinguished name for the entry being removed.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

This function is called when Delegated Management Services is used to delete directory entries, including groups or roles.

SmDirRemoveMemberFromGroup()

The SiteMinder Policy Server calls SmDirRemoveMemberFromGroup() so that you can remove a user or group from a existing group.

Syntax

```
int SM_EXTERN SmDirRemoveMemberFromGroup (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszMemberDN,
   const char*              lpszGroupDN
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.
lpszMemberDN	I	Buffer containing the distinguished name for the user or group being removed from the parent group.
lpszGroupDN	I	Buffer containing the distinguished name for the group from which the member is being removed.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

This function is called when Delegated Management Services is used to remove users or groups from groups.

SmDirRemoveMemberFromRole()

The SiteMinder Policy Server calls SmDirRemoveMemberFromRole() so that you can remove a user or group from an assigned role.

Syntax

```
int SM_EXTERN SmDirRemoveMemberFromRole (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszMemberDN,
   const char*              lpszRoleDN
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.
lpszMemberDN	I	Buffer containing the distinguished name for the user or group being removed from the assigned role.
lpszRoleDN	I	Buffer containing the distinguished name for the role from which the member is being removed.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

This function is called when Delegated Management Services is used to remove users or groups from assigned roles.

SmDirSearch()

SiteMinder calls SmDirSearch() to search on the criteria specified in the search filter *lpszSearchFilter*. You could use SmDirSearch() to execute a query, such as an SQL select, on your custom directory.

In addition to the search filter, the function SmDirSearch() passes directory search parameters. In the SiteMinder Administrative UI, on the Directory Setup tab, you can specify parameters in the following fields:

- Max time
- Max results

The *nSearchTimeout* and *nSearchResults* parameters of SmDirSearch() pass the information entered in those fields.

SiteMinder calls SmDirFreeStringArray() to free the *lpszDNs* array.

Syntax

```
int SM_EXTERN SmDirSearch (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   char***                  lpszDNs,
   const char*              lpszSearchFilter,
   const char*              lpszSearchRoot,
   const int                nSearchResults,
   const int                nSearchTimeout,
   const int                nSearchScope
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().

Parameter	I/O	Description
pInstanceHandle	I	The address of a pointer to the directory instance handle.
lpszDNs	O	Distinguished names of the users found as a result of the search.
lpszSearchFilter	I	Buffer containing the search expression.
lpszSearchRoot	I	This parameter is designed to hold the search root—that is, the starting point for the search. In a hierarchical directory, when authenticating a user, SiteMinder starts at the root and works down the tree. In the SiteMinder Administrative UI, on the Directory Setup tab, in the Custom Namespace group box, the value entered in the Parameter field is passed in through *lpszSearchRoot*. **Note:** The string entered in the Parameter field is also passed to *lpszParameter* for use with SmDirInit() and SmDirInitDirInstance(). If your code for SmDirInit() needs to use *lpszParameter* and your code for SmDirSearch() needs a search root, you will have to parse the string from the Parameter field.
nSearchResults	I	This parameter holds the maximum number of records that can be returned for a single search of the directory.
nSearchTimeout	I	This parameter holds the maximum time in seconds that the API should keep searching the directory for results.
nSearchScope	I	This parameter indicates how far below the root (*lpszSearchRoot*) the API will query the directory to find a match. Depending on the value in *nSearchScope*, the search could go down only one level or through the entire subtree. The default value is 2.

Returns

Returns 0 if successful or -1 if not successful.

SmDirSearchCount()

Use the function SmDirSearchCount() to return a count of the entries retrieved through the search criteria specified in *lpszSearchFilter*, subject to the restrictions specified in the directory search parameters. See the Remarks section for more information.

Syntax

```
int SM_EXTERN SmDirSearchCount (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   int*                     pnCount,
   const char*              lpszSearchFilter,
   const char*              lpszSearchRoot,
   const int                nSearchResults,
   const int                nSearchTimeout,
   const int                nSearchScope
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.
pnCount	O	Number of entries returned by the search
lpszSearchFilter	I	Buffer containing search expression.
lpszSearchRoot	I	This parameter is designed to hold the search root, or the starting point for the search. In a hierarchical directory, when authenticating a user, SiteMinder starts at the root and works down the tree.
nSearchResults	I	The parameter holds the maximum number of records that can be returned for a single search of the directory.
nSearchTimeout	I	This parameter holds the maximum time in seconds that the API should keep searching the directory for results.

Parameter	I/O	Description
nSearchScope	I	This parameter indicates how far below the root (*lpszSearchRoot*) the API will query the directory to find a match. Depending on the value in *nSearchScope*, the search could go down only one level or through the entire subtree. The default value is 2.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

In the SiteMinder Policy Server User Interface, on the Directory Setup tab, parameters can be specified in the following fields:

- Parameter—Search root. Its value is passed in through parameter *lpszSearchRoot*.
- Max time—Maximum search time in seconds. Its value is passed in through parameter *nSearchTimeout*.
- Max results—Maximum number of results to return. Its value is passed in through parameter *nSearchResults*.

Note: The string entered in the Parameter field is also passed to *lpszParameter* for use with SmDirInit() and SmDirInitDirInstance(). If your code for SmDirInit() needs to use *lpszParameter* and your code for SmDirSearchCount() needs a search root, you will have to parse the string from the Parameter field.

SmDirSetUserAttr()

The SiteMinder Policy Server calls SmDirSetUserAttr() so that you can use SiteMinder to set a user attribute in your custom directory. For example, you may need to change the last name of a user.

Syntax

```
int SM_EXTERN SmDirSetUserAttr (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUserDN,
   const char*              lpszAttrName,
   const char*              lpszAttrData
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.
lpszUserDN	I	Buffer containing the user DN whose user attribute has to be set.
lpszAttrName	I	Buffer containing the name of the user attribute.
lpszAttrData	I	Buffer containing the value of the user attribute.

Returns

Returns 0 if successful or -1 if not successful.

SmDirSetUserAttrMulti()

The SiteMinder Policy Server calls SmDirSetUserAttrMulti() so that you can set an array of values for a single attribute in your custom directory.

Syntax

```
int SM_EXTERN SmDirSetUserAttrMulti (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUserDN,
   const char*              lpszAttrName,
   const char**             lpszAttrData
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.
lpszUserDN	I	Buffer containing the user DN whose user attribute has to be set.
lpszAttrName	I	Buffer containing the name of the user attribute.
lpszAttrData	I	Buffer containing the values for the user attribute.

Returns

Returns 0 if successful or -1 if not successful.

SmDirSetUserDisabledState()

The SiteMinder Policy Server calls SmDirSetUserDisabledState() when an administrator uses the SiteMinder Policy Server User Interface to disable or enable a user, or Password Services disables a user.

This call gives you the opportunity to set the disabled flag in your custom directory to the disabled reason passed in through nDisabledReason.

When implementing SmDirSetUserDisabledState(), be sure that you have specified which field (or attribute) in the custom directory will hold the disabled reason. In the SiteMinder Policy Server User Interface, specify the attribute name in the Disabled Flag field on the User Attributes tab on the User Directory dialog box. This attribute is passed in through lpszDisabledAttr.

Syntax

```
int SM_EXTERN SmDirSetUserDisabledState (
   const Sm_Api_Context_t*        lpApiContext,
   void*                          pHandle,
   void*                          pInstanceHandle,
   const char*                    lpszUserDN,
   const char*                    lpszDisabledAttr,
   const Sm_Api_DisabledReason_t  nDisabledReason
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.
lpszUserDN	I	Buffer containing the Distinguished Name (DN) of the user whose disabled state has to be modified.
lpszDisabledAttr	I	The user directory attribute that holds a user's disabled state. The SiteMinder Policy Server checks this attribute to see if a user is enabled or disabled. If a user is disabled, this attribute also holds the specific reason. Use this attribute to change a user's disabled state.

Parameter	I/O	Description
nDisabledReason	I	Reason that the user was disabled or enabled. Possible reasons are enumerated in Sm_Api_DisabledReason_t. Store the user-disabled state in the user directory attribute that is specified in lpszDisabledAttr.

Note: A user account can be disabled for multiple reasons. Be sure to hold onto the disabled reason(s) and be sure that you don't overwrite those bits. For example, if the User must change password at next login checkbox is checked and the administrator clicks Disable, the nDisabledReason parameter holds both the Sm_Api_Disabled_PWMustChange bit and the Sm_Api_Disabled_AdminDisabled bit. When the user account is enabled, be sure to clear all the disabled bits.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

To enable a user's account in the Policy Server User Interface:

1. Navigate to the User Management dialog box.
2. Select the user whose account you are enabling.
3. Click Enable.

To disable a user's account in the Policy Server User Interface:

1. Navigate to the User Management dialog box.
2. Select the user whose account you are disabling.
3. Click Disable.

Enabled user accounts can also be disabled by using Password Services. For example, you can configure Password Services to disable a user account under the following conditions:

- Inactivity—A specified period of time has elapsed since the user has logged in.
- Failed login attempts—The user has reached a specified number of consecutive failed log in attempts.
- Expired password—The user has failed to make a required password change.
- Forced password change—The user is being forced to change his or her password. On the User Management dialog box, the administrator has checked User must change password at next login.

Sample Code Information

When using the sample, specify the disabled attribute as Disabled. This attribute will be passed in through the parameter lpszDisabledAttr. You can specify the attribute by entering Disabled in the Disabled Flag field on the User Attributes tab of the User Directory dialog box.

In the SiteMinder Policy Server User Interface, in the User Management dialog box, suppose an administrator selects the enabled user Mikel and clicks Disable. Using the sample code, the User DN (lpszUserDN) is Mikel, and the Disabled Attribute (lpszDisabledAttr) is Disabled. The Disabled Reason (nDisabledReason) is Sm_Api_Disabled_AdminDisabled. In the SiteMinder Policy Server User Interface, the User Management dialog box shows that the Current Settings for Mikel have changed from User is enabled to Disabled - administrative.

SmDirValidateInstance()

SiteMinder calls SmDirValidateInstance() so that the Directory API can validate the instance for which the handle is passed.

Either the directory instance handle or user instance handle can be passed through pInstanceHandle. Your code must determine which handle is passed. Check to see if the handle is valid and return the corresponding response.

Syntax

```
int SM_EXTERN SmDirValidateInstance (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle or directory instance handle.

Returns

Returns 0 if successful or -1 if not successful.

SmDirValidateUserDN()

Use this function to perform any needed validation on the user ID passed in through lpszUserDN.

If you do not need to implement SmDirValidateUserDN(), return 0, as shown in the sample code.

Syntax

```
int SM_EXTERN SmDirValidateUserDN (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUserDN
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.
lpszUserDN	I	Buffer containing the user DN that has to be validated.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

This function is called when you use the SiteMinder Test Tool to run IsAuthenticated for a user in the custom directory.

SmDirValidateUsername()

SiteMinder calls this function to validate the user name passed in through lpszUsername.

You can use this function to resolve the user name and copy the user ID into the lpszNewUsername buffer. If you choose not to implement SmDirValidateUsername(), set lpszNewUsername to NULL, as shown in the sample.

SiteMinder calls SmDirFreeString() to free the lpszNewUsername buffer.

Syntax

```
int SM_EXTERN SmDirValidateUsername (
   const Sm_Api_Context_t*  lpApiContext,
   void*                    pHandle,
   void*                    pInstanceHandle,
   const char*              lpszUsername,
   char**                   lpszNewUsername
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the directory instance handle.
lpszUsername	I	Buffer containing the user name that has to be validated.
lpszNewUsername	O	Buffer containing the validated user name.

Returns

Returns 0 if successful or -1 if not successful.

Remarks

This function is called when you use the SiteMinder Test Tool to run IsAuthenticated for a user in the custom directory.

SmDirValidateUserPolicyRelationship()

Use SmDirValidateUserPolicyRelationship() to validate the relationship between policy objects. Determine whether the user distinguished name has the specified relationship to the policy distinguished name. The relationship is passed in through nPolicyResolution.

For example:

- If nPolicyResolution is Sm_PolicyResolution_User, determine whether the user passed in through lpszUserDN is the same user that is passed in through lpszPolicyDN.
- If nPolicyResolution is Sm_PolicyResolution_UserGroup, determine whether the user passed in through lpszUserDN is a member of the group passed in through lpszPolicyDN.

Syntax

```
int SM_EXTERN SmDirValidateUserPolicyRelationship (
   const Sm_Api_Context_t*      lpApiContext,
   void*                        pHandle,
   void*                        pInstanceHandle,
   const char*                  lpszUserDN,
   const Sm_PolicyResolution_t  nPolicyResolution,
   const int                    bRecursive,
   const char*                  lpszPolicyDN,
   const char*                  lpszPolicyClass
);
```

Parameter	I/O	Description
lpApiContext	I	Pointer to the API context structure.
pHandle	I	The address of the pointer returned by SmDirInit().
pInstanceHandle	I	The address of a pointer to the user instance handle.
lpszUserDN	I	Buffer containing the user DN for which you must validate the relationship.

Parameter	I/O	Description
nPolicyResolution	I	The relationship between the user distinguished name and the policy distinguished name should match what is specified in nPolicyResolution. Specific policy resolutions are enumerated in Sm_PolicyResolution_t. For more information on Sm_PolicyResolution_t see Sm_PolicyResolution_t.
bRecursive	I	Whether the directory supports recursion (for example, nested groups). Suppose that the value of nPolicyResolution is Sm_PolicyResolution_UserGroup, the User DN is Bill Collector, and the Policy DN is Accounting. If your custom directory supports recursion, you will need to search down any hierarchy of groups to find the user. Bill Collector may be in a group called AR, which may be in the group Accounting.
lpszPolicyDN	I	Distinguished names of the object, such as users, groups or roles, bound to the policy.
lpszPolicyClass	I	Class of the object named in lpszPolicyDN. For example, the class could be Group.

Returns

Returns 0 if successful or -1 if not successful.

Structures Used in the Sample Directory Application

The sample Directory API application resides in:

```
<install_path>\sdk\samples\smdirapi\smdirapi.cpp
```

The Directory API includes a directory instance handle, a directory provider handle, and a directory entry (user) instance handle. These handles are returned from the initialization functions listed in section Initialization and Release Functions.

The sample code uses the following structures to manage these handles:

Handle Type	**Data Structure**
Directory instance	DirHandle_t
Directory provider	ProviderHandle_t
Directory entry (user) instance	UserHandle_t

Directory Instance Handle

The sample instantiates DirHandle_t when SmDirInitDirInstance() is called. The handle is then passed to the directory operations functions.

The same value need not be carried through the entire process. You are permitted to change the value.

The definition of DirHandle_t is as follows:

```
typedef struct DirHandle_s
{
   char nTag;
   bool bValid;
   char szErrMsg[ERRMSG_SIZE];
   char* pszUniqueKey;
   char* pszParameter;
   char* pszUsername;
   char* pszPassword;
   int bRequireCredentials;
   int bSecureConnection;
   int nSearchResults;
   int nSearchTimeout;
} DirHandle_t;
```

The sample releases DirHandle_t when SmDirReleaseInstance() is called. The value of *nTag* is used to distinguish between the directory instance handle and the user instance handle.

More Information:

How To Distinguish between Handle Types (see page 751)

Directory Provider Handle

The sample defines the provider handle structure ProviderHandle_t to serve as a bridge between the SiteMinder Policy Server and the Directory API. The provider handle can be used to store data from the time SiteMinder loads the library until the SiteMinder Policy Server shuts down.

The sample instantiates ProviderHandle_t when SmDirInit() is called. The handle is then passed to almost all subsequent functions. The same value need not be carried through the entire process. You are permitted to change the value.

The sample releases ProviderHandle_t when SmDirRelease() is called.

See the function SmDirInit() in the sample code for an example of ProviderHandle_t. You can follow this example, but you aren't required to. ProviderHandle_t can contain any information that you would like to set at the beginning of the process and carry through, such as a user's password.

Directory Entry (User) Instance Handle

The sample instantiates UserHandle_t when SmDirInitUserInstance() is called. This handle is then passed to the directory entry (user) operations functions. For a list of these functions, see Operations on a Directory Entry (User).

The same value need not be carried through the entire process. You are permitted to change the value.

The definition of UserHandle_t is as follows:

```
typedef struct UserHandle_s
{
   char nTag;
   bool bValid;
   char szErrMsg[ERRMSG_SIZE];
   DirHandle_t* phDir;
   char* pszUserDn;
} UserHandle_t;
```

The sample releases UserHandle_t when SmDirReleaseInstance() is called.

How To Distinguish between Handle Types

Some functions, such as SmDirReleaseInstance(), may be passed either the directory instance handle or the directory entry (user) instance handle. The sample code provides a way you can distinguish the directory instance handle from the directory entry (user) instance handle.

Notice that *nTag* is the first field of both DirHandle_t and UserHandle_t. When SmDirInitDirInstance() is called, *nTag* is set to 0 in DirHandle_t. When SmDirInitUserInstance is called, *nTag* is set to 1 in UserHandle_t.

When a function that accepts either type of handle is called, the value of *nTag* is checked to see which type of handle is being passed.

Chapter 13: Common Data Types and Structure

This section contains the following topics:

Exported Enumerations

The following enumerations in SmApi.h are used by more than one SiteMinder API:

- Sm_Api_DisabledReason_t (see page 753)
- Sm_Api_Reason_t (see page 755)
- Sm_PolicyResolution_t (see page 757)

Sm_Api_DisabledReason_t

Enumerates the reasons that a user account can be disabled.

The following APIs use Sm_Api_DisabledReason_t:

- Policy Management API
- Directory API

Note: A user account can be disabled for multiple reasons. For example, if the User must change password at next login checkbox is checked and the administrator clicks Disable, an *nDisabledReason* parameter holds both the Sm_Api_Disabled_PWMustChange bit and the Sm_Api_Disabled_AdminDisabled bit.

For examples of using Sm_Api_DisabledReason_t, see the examples under Sm_PolicyApi_SetDisabledUserState().

The following table shows the bits that can be set for disabled reason. A brief explanation of each reason, organized by reason type, follows the table.

Disabled Reason	Type	Value
Sm_Api_Disabled_DisabledMask	Mask	0x00ffffff
Sm_Api_Disabled_Enabled	Mask	0

Disabled Reason	Type	Value
Sm_Api_Disabled_AdminDisabled	Bits	0x00000001
Sm_Api_Disabled_MaxLoginFail	Bits	0x00000002
Sm_Api_Disabled_Inactivity	Bits	0x00000004
Sm_Api_Disabled_PWExpired	Bits	0x00000008
Sm_Api_Disabled_DirNativeDisabled	Bits	0x00000010
Sm_Api_Disabled_PWMustChange	Qualifier	0x01000000

Disabled Mask

- Sm_Api_Disabled_DisabledMask

  The disable bits mask is used to distinguish between two cases:

  - When disabled bits are set, the resulting hexadecimal value begins 0x00.
  - When users are forced to change their passwords, the resulting hexadecimal value begins 0x01.

- Sm_Api_Disabled_Enabled

  When a user account is enabled, the value is 0, as opposed to the other cases that have non-zero values.

Disabled Bits

A user account can be disabled for one or more of the following reasons:

- Sm_Api_Disabled_AdminDisabled

  Disabled by administrator.

- Sm_Api_Disabled_MaxLoginFail

  Disabled for maximum login failures.

- Sm_Api_Disabled_Inactivity

  Disabled for inactivity over a period of time.

- Sm_Api_Disabled_PWExpired

  Disabled for password expiration.

- Sm_Api_Disabled_DirNativeDisabled

  Disabled by the user repository and cannot be enabled in SiteMinder. For example, SiteMinder returns this disabled reason if an Active Directory user object expires because its accountExpires time elapsed.

Qualifier

- Sm_Api_Disabled_PWMustChange

  Forces a user to change his or her password during the next login. When users are forced to change their passwords, the resulting hexadecimal value begins 0x01, as opposed to 0x00.

Sm_Api_Reason_t

Enumerates the reasons for an access event, such as an authentication failure or session logout.

When an authentication scheme fails to authenticate, it may send back a reason status code. The status code returned from the authentication function is constructed using the SM_MAKEAUTH_STATUSVALUE macro (see Returns on page 10).

The Policy server sends the reason to the agent. The SiteMinder Web Agent exposes the reason so that Web applications can use it in their logic.

The following APIs use Sm_Api_Reason_t:

- Authentication API
- Event API
- Directory API

The following table shows the bits that can be set for access events.

Note: Values 0 through 31999 are reserved for use by SiteMinder. Values 32000 through 32767 are available for user-defined reasons.

Authentication Reason	**Value**
Sm_Api_Reason_None	0
Sm_Api_Reason_PwMustChange	1
Sm_Api_Reason_InvalidSession	2
Sm_Api_Reason_RevokedSession	3
Sm_Api_Reason_ExpiredSession	4
Sm_Api_Reason_AuthLevelTooLow	5
Sm_Api_Reason_UnknownUser	6
Sm_Api_Reason_UserDisabled	7
Sm_Api_Reason_InvalidSessionId	8

Authentication Reason	**Value**
Sm_Api_Reason_InvalidSessionIp	9
Sm_Api_Reason_CertificateRevoked	10
Sm_Api_Reason_CRLOutOfDate	11
Sm_Api_Reason_CertRevokedKeyCompromised	12
Sm_Api_Reason_CertRevokedAffiliationChange	13
Sm_Api_Reason_CertOnHold	14
Sm_Api_Reason_TokenCardChallenge	15
Sm_Api_Reason_ImpersonatedUserNotInDir	16
Sm_Api_Reason_Anonymous	17
Sm_Api_Reason_PwWillExpire	18
Sm_Api_Reason_PwExpired	19
Sm_Api_Reason_ImmedPWChangeRequired	20
Sm_Api_Reason_PWChangeFailed	21
Sm_Api_Reason_BadPWChange	22
Sm_Api_Reason_PWChangeAccepted	23
Sm_Api_Reason_ExcessiveFailedLoginAttempts	24
Sm_Api_Reason_AccountInactivity	25
Sm_Api_Reason_NoRedirectConfigured	26
Sm_Api_Reason_ErrorMessageIsRedirect	27
Sm_Api_Reason_Next_Tokencode	28
Sm_Api_Reason_New_PIN_Select	29
Sm_Api_Reason_New_PIN_Sys_Tokencode	30
Sm_Api_Reason_New_User_PIN_Tokencode	31
Sm_Api_Reason_New_PIN_Accepted	32
Sm_Api_Reason_Guest	33
Sm_Api_Reason_PWSelfChange	34
Sm_Api_Reason_ServerException	35
Sm_Api_Reason_UnknownScheme	36
Sm_Api_Reason_UnsupportedScheme	37

Authentication Reason	**Value**
Sm_Api_Reason_Misconfigured	38
Sm_Api_Reason_BufferOverflow	39
Sm_Api_Reason_SetPersistentSessionFailed	40
Sm_Api_Reason_UserLogout	41
Sm_Api_Reason_IdleSession	42
Sm_Api_Reason_PolicyServerEnforcedTimeout	43
Sm_Api_Reason_PolicyServerEnforcedIdle	44
Sm_Api_Reason_ImpersonationNotAllowed	45
Sm_Api_Reason_ImpersonationNotAllowedUser	46
Sm_Api_Reason_FederationNoLoginID	47
Sm_Api_Reason_FederationUserNotInDir	48
Sm_Api_Reason_FederationInvalidMessage	49
Sm_Api_Reason_FederationUnacceptedMessage	50

Sm_PolicyResolution_t

Sm_PolicyResolution_t, defined in SmApi.h, enumerates values that describe how one policy object, such as a user, is related to another policy object, such as a group.

The following APIs use the enumeration Sm_PolicyResolution_t:

- Policy Management API
- Directory API

The following table lists the supported policy resolutions. A brief description of each resolution follows the table.

Policy Resolution	**Value**
Sm_PolicyResolution_Unknown	0
Sm_PolicyResolution_User	1
Sm_PolicyResolution_UserGroup	2
Sm_PolicyResolution_UserProp	3
Sm_PolicyResolution_UserRole	4

Policy Resolution	Value
Sm_PolicyResolution_Org	5
Sm_PolicyResolution_Query	6
Sm_PolicyResolution_All	7
Sm_PolicyResolution_GroupProp	8
Sm_PolicyResolution_OrgProp	9
Sm_PolicyResolution_DnProp	10

Note: Another policy resolution, Sm_PolicyResolution_Max, is for internal use only.

- Sm_PolicyResolution_Unknown. The policy object is unknown.
- Sm_PolicyResolution_User. The policy object is the specified user object.
- Sm_PolicyResolution_UserGroup. The policy object is a member (directly, or indirectly through another group) of the specified user group object.
- Sm_PolicyResolution_UserProp. The policy object matches the specified filter (user scope).
- Sm_PolicyResolution_UserRole. The policy object is the occupant of the specified role.
- Sm_PolicyResolution_Org. The policy object is a member (directly or indirectly through another organization) of the specified organization object (this supports organizations and organization units).
- Sm_PolicyResolution_Query. The policy object is contained in the result set of a directory-specific query.
- Sm_PolicyResolution_All. The policy object is located in the specified directory.
- Sm_PolicyResolution_GroupProp. The policy object is policy-related to a "group"-like DN matching the specified filter.
- Sm_PolicyResolution_OrgProp. The policy object is policy-related to a "org"-like DN matching the specified filter.
- Sm_PolicyResolution_DnProp. The policy object is policy-related to any DN matching the specified filter.

More Information:

SmDirValidateUserPolicyRelationship() (see page 746)

Common Structure

The Sm_Api_Context_t structure is used by multiple SiteMinder APIs.

The structure provides the function pointers for the SiteMinder logging utility, trace utility, and error utility.

The following APIs use Sm_Api_Context_t:

- Authentication API
- Authorization API
- Tunnel Service API
- DMS Workflow API
- Directory API

More Information:

Sm_Api_Context_t (see page 572)

Chapter 14: Event Log Formats

Access Events

Access events indicate user-related activities. They are called in the context of authentication, authorization, and administration activity.

The format for access events in a text log depends on the event category ID.

Authentication and Authorization Format

If the event category ID is authentication (SmLogAccessCat_Auth) or authorization (SmLogAccessCat_Az), the format is:

```
lpszEvent lpszHostName lpszTimeString "szClientIp szUserName" "szAgentName szAction
szResource" [szTransactionId] [nReason] szStatusMsg
```

Elements in the above format example are described as follows:

- lpszEvent. The name (type) of the access event:

```
SmLogAccessEvent_AuthAccept : lpszEvent = "AuthAccept"
SmLogAccessEvent_AuthReject : lpszEvent = "AuthReject"
SmLogAccessEvent_AuthAttempt : lpszEvent = "AuthAttempt"
SmLogAccessEvent_AuthChallenge : lpszEvent = "AuthChallenge"
SmLogAccessEvent_AzAccept : lpszEvent = "AzAccept"
SmLogAccessEvent_AzReject : lpszEvent = "AzReject"
SmLogAccessEvent_AdminLogin : lpszEvent = "AdminLogin"
SmLogAccessEvent_AdminLogout : lpszEvent = "AdminLogout"
SmLogAccessEvent_AdminReject : lpszEvent = "AdminReject"
SmLogAccessEvent_AuthLogout : lpszEvent = "AuthLogout"
SmLogAccessEvent_ValidateAccept : lpszEvent = "ValidateAccept"
SmLogAccessEvent_ValidateReject : lpszEvent = "ValidateReject"
```

- lpszHostName. The name of the host.
- lpszTimeString. The timestamp of the occurrence of the event, in the format: [<date>/<month>/<year>:<hour>:<minute>:<second> <difference from GMT>]. For example: [27/Jun/2000:11:27:29 -0500]
- szClientIp. The IP address of the client machine.
- szUserName. The name of the user.
- szAgentName. The name of the agent.
- szAction. The action associated with the resource.
- szResource. The accessed resource.
- [szTransactionId]. A string that contains: idletime=<value>.

- [nReason]. The reason associated with the event. Reasons are enumerated in Sm_Api_Reason_t, which is in SmApi.h.
- szStatusMsg. The message associated with the event. The message depends on the event type, as shown in in the following table:

Event	**Role of szStatusMsg**
SmLogAccessEvent_AdminLogin	Holds the UserMsg returned by the authentication scheme.
SmLogAccessEvent_AdminReject	Holds the ErrorMsg returned by the authentication Scheme
SmLogAccessEvent_AuthAccept	Holds the UserMsg.
SmLogAccessEvent_AuthReject	Holds a concatenated string of UserMsg and ErrorMsg.
SmLogAccessEvent_AuthAttempt	Holds a concatenated string of UserMsg and ErrorMsg.
SmLogAccessEvent_AuthChallenge	Holds the UserMsg.
SmLogAccessEvent_ValidateAccept	Is an empty string.
SmLogAccessEvent_ValidateReject	Holds an error message containing the reason for validate reject. Examples: "Invalid session token" "Invalid session id" "Invalid session ip" "Invalid user DN" "Session has expired" "Invalid key in use" "Invalid error status"
SmLogAccessEvent_AuthLogout	An empty string.

Event	Role of szStatusMsg
SmLogAccessEvent_AzAccept	An empty string.
SmLogAccessEvent_AzReject	Depending on the type of az reject, it is a string explaining the reason for the reject. Examples: "Invalid session type for affiliate agent" "Invalid session type" "Session not authorized for this security level"

For example:

```
AuthAccept testbox [27/Jun/2000:11:27:29 -0500] "190.158.4.90
uid=scarter,ou=people,o=airius.com" "testagent GET /test/index.html"
[idletime=3600;maxtime=7200;authlevel=5;] [0]
```

In this example,

- lpszEvent is AuthAccept
- lpszHostName is testbox
- lpszTimeString is [27/Jun/2000:11:27:29 -0500]
- szClientIp is 190.158.4.90
- szUserName is uid=scarter,ou=people,o=airius.com
- szAgentName is testagent
- szAction is GET
- szResource is /test/index.html
- [szTransactionId] is [idletime=3600;maxtime=7200;
  authlevel=5;]
- [nReason] is [0]
- szStatusMsg is not specified.

Administration Format

If the event category ID is SmLogAccessCat_Admin, the format is:

```
lpszEvent lpszHostName lpszTimeString "szClientIp szUserName"     szStatusMsg
```

Elements in the above format example are described as follows:

- lpszEvent is the name (type) of the access event:

  ```
  SmLogAccessEvent_AdminLogin:lpszEvent = "AdminLogin"
  SmLogAccessEvent_AdminLogout:lpszEvent = "AdminLogout"
  SmLogAccessEvent_AdminReject:lpszEvent = "AdminReject"
  ```

- lpszHostName is the name of the host.
- lpszTimeString is the timestamp of the occurrence of the event, in the format: [<date>/<month>/<year>:<hour>:<minute>:<second> <difference from GMT>]. For example:

  ```
  [27/Jun/2000:11:27:29 -0500]
  ```

- szClientIp is the IP address of the client machine.
- szUserName is the name of the user.
- szStatusMsg is the message associated with the event. The message depends on the event type, as shown in the following table:

Event	Role of szStatusMsg
SmLogAccessEvent_AdminLogin	Holds the UserMsg returned by the authentication scheme.
SmLogAccessEvent_AdminReject	Holds the ErrorMsg returned by the authentication scheme

For example:

```
AdminLogin testbox [27/Jun/2000:11:26:50 -0500]
   "190.158.4.90 siteminder"
```

In this example,

- lpszEvent is AdminLogin
- lpszHostName is testbox
- lpszTimeString is [27/Jun/2000:11:26:50 -0500]
- szClientIp is 190.158.4.90
- szUserName is siteminder
- szStatusMsg is not specified.

Object Events

Object events are called when

- SiteMinder objects are created, updated, or deleted.
- An application or a user logs in to the object store.

Object events are called in the context of authentication, SiteMinder object changes, and management activity.

The format for object events in a text log depends upon the object event category ID.

AdminChange Format

AdminChange events occur when an administrator adds, updates, or deletes an object. The format is:

```
AdminChange <Hostname> <Time String> "- <Username> " <Event> <Category>
'<ObjectName>'
```

Elements in the above format example are described as follows:

- <HostName> is the name of the host.
- <Time String> is the timestamp of the occurrence of the event, in the format: [<date>/<month>/<year>:<hour>:<minute>:<second> <difference from GMT>]. For example:

  ```
  [27/Jun/2000:11:27:29 -0500]
  ```
- <UserName> is the name of the user who generated the event.
- <Event> is the name of the object event—namely:

  ```
  Create
  Delete
  Update
  UpdateField
  ```
- <Category> is the object that is the target of the event—for example, Rule or UserDirectory.
- <ObjectName> is the user-defined name for the object. Some object categories (such as RootConfig) have no ObjectName associated with them.

Here is an example of an AdminChange event format that was logged when administrator JLewis created the rule MyNewRule:

```
AdminChange MyHost [20/Jul/2001:10:26:15 -0500] "- JLewis" Create Rule 'MyNewRule'
```

Management Command Format

If the object category ID is management command (SmLogObjCat_ManagementCommand), the format is:

```
ManagementCommand <Hostname> <Time String> "- <Username> " <Event> '<Description>'
```

Elements in the above format example are described as follows:

- <HostName> is the name of the host.
- <Time String> is the timestamp of the occurrence of the event, in the format: [<date>/<month>/<year>:<hour>:<minute>:<second> <difference from GMT>]. For example:

  ```
  [27/Jun/2000:11:27:29 -0500]
  ```

- <UserName> is the name of the user who generated the event.
- <Event> is the name of the management command event—namely:

  ```
  FlushAll            ChangeDynamicKeys
  FlushUser           ChangePersistentKey
  FlushAllUsers       ChangeDisabledUserState
  FlushAllRealms      ChangeUserPassword
  ```

- <Description> is the user DN for management commands that involve a user, such as flush user command, a change disabled user state, and a change password.

Here is an example of a management command event format that was logged when administrator JLewis flushed the user cache for BRoy:

```
ManagementCommand MyHost [20/Jul/2001:13:26:23 -0500]
"- JLewis" FlushUser 'uid=BRoy,ou=HR,o=security.com'
```

EMS Events

EMS events include the following:

- Creating, deleting or modifying any of the following objects:
    - User
    - Organization
    - User role
    - Admin role
    - Resource
    - Generic object in the Top object class
- Enabling or disabling a user
- Assigning or removing a user role
- Modifying a user's password
- Logging in or logging out as an administrator
- Failing to authenticate
- EMS server session timeout

EMS Log Format

SiteMinder logs EMS events to a text file using the following format. In the format example, literal strings are shown in **bold** type:

```
lpszTimeString: Category lpszCat (nCategoryId),
      Event lpszEvent (nEventId)
   Username szUserName, SessionId szSessionId
   DirectoryName szDirName
   ObjectName szObjName, ObjectClass szObjClass,
      ObjectPath szObjPath
   Organization szOrgName, Role szRoleName
   Description: szFieldDesc
   Status: szStatusMsg
```

Elements in the preceding format example are described as follows:

- lpszTimeString is the timestamp of the occurrence of the event, in the format: [<date>/<month>/<year>:<hour>:<minute>:<second> <difference from GMT>]. For example:

  ```
  [27/Jun/2000:11:27:29 -0500]
  ```

- nCategoryId contains the category ID, and lpszCat contains the corresponding category name.
- nEventId contains the EMS event ID, and lpszEvent contains the corresponding event name.

The remaining fields, shown in *italics*, are members of the structure SmLog_EMS_t.

Category ID (nCategoryId)	**Category (lpszCat)**
SmLogEmsCat_DirectoryUser	"User"
SmLogEmsCat_DirectoryAdmin	"Admin"
SmLogEmsCat_DirectorySession	"Session"

Event ID (nEventId)	**Event (lpszEvent)**
SmLogEmsEvent_CreateUser	"CreateUser"
SmLogEmsEvent_DeleteUser	"DeleteUser"
SmLogEmsEvent_ModifyUser	"ModifyUser"
SmLogEmsEvent_AssignUserRole	"AssignUserRol" (sic)
SmLogEmsEvent_RemoveUserRole	"RemoveUserRole"
SmLogEmsEvent_EnableUser	"EnableUser"
SmLogEmsEvent_DisableUser	"DisableUser"
SmLogEmsEvent_CreateOrg	"CreateOrganization"
SmLogEmsEvent_DeleteOrg	"DeleteOrganization"
SmLogEmsEvent_ModifyOrg	"ModifyOrganization"
SmLogEmsEvent_CreateRole	"CreateRole"
SmLogEmsEvent_DeleteRole	"DeleteRole"
SmLogEmsEvent_ModifyRole	"ModifyRole"
SmLogEmsEvent_CreateResource	"CreateResource"

Event ID (nEventId)	Event (lpszEvent)
SmLogEmsEvent_DeleteResource	"DeleteResource"
SmLogEmsEvent_ModifyResource	"ModifyResource"
SmLogEmsEvent_AssignResourceRole	"AssignResourceRole"
SmLogEmsEvent_RemoveResourceRole	"RemoveResourceRole"
SmLogEmsEvent_Login	"Login"
SmLogEmsEvent_Logout	"Logout"
SmLogEmsEvent_LoginFail	"LoginFail"
SmLogEmsEvent_SessionTimeout	"SessionTimeout"
SmLogEmsEvent_AuthFail	"AuthFail"
SmLogEmsEvent_PasswordModify	"ChangePassword"
SmLogEmsEvent_CreateAdminRole	"CreateAdminRole"
SmLogEmsEvent_DeleteAdminRole	"DeleteAdminRole"
SmLogEmsEvent_ModifyAdminRole	"ModifyAdminRole"
SmLogEmsEvent_AssignAdminRole	"AssignAdminRole"
SmLogEmsEvent_RemoveAdminRole	"RemoveAdminRole"
SmLogEmsEvent_AddManagedOrg	"AddManagedOrg"
SmLogEmsEvent_RemoveManagedOrg	"RemoveManagedOrg"
SmLogEmsEvent_CreateObject	"CreateObject"
SmLogEmsEvent_DeleteObject	"DeleteObject"
SmLogEmsEvent_ModifyObject	"ModifyObject"

Appendix A: SAML 2.0 Property Reference

This section contains the following topics:

About the SAML 2.0 Properties

This reference includes the SAML 2.0 metadata properties associated with the structure Sm_PolicyApi_SAMLProviderProp_t. The properties are defined in the file SmPolicyApi45.h.

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 validate a principal based on the information in a SAML assertion. The assertion is supplied by the Identity Provider associated with the authentication scheme.

SAML 2.0 Property Reference

This reference includes the SAML 2.0 metadata properties associated with the structure Sm_PolicyApi_SAMLProviderProp_t. The properties are defined in the file SmPolicyApi45.h.

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 validate a principal based on the information in a SAML assertion. The assertion is supplied by the Identity Provider associated with the authentication scheme.

When reading the property references, note the following:

- Unless otherwise specified, properties of data type String have the following maximum lengths:
  - URIs and URLs must be less than 1,024 characters
  - All other Strings must not exceed 255 characters
- Specify Boolean values as either SAML_TRUE or SAML_FALSE.

SAML_AFFILIATION

Type	String
Applies to	SAML 2.0 Authentication, Service Provider
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. Identity Providers 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.

Note: An Identity Provider is assigned to an affiliation through its associated SAML 2.0 authentication scheme.

SAML_AUDIENCE

Type	String
Applies to	SAML 2.0 Authentication, Service Provider
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

Type	String
Applies to	SAML 2.0 Affiliation, SAML 2.0 Authentication, Service Provider
Required	No
Default	None

Description

A brief description of the affiliation, authentication scheme, or Service Provider object.

SAML_DISABLE_SIGNATURE_PROCESSING

Type	Boolean
Applies to	SAML 2.0 Authentication, Service Provider
Required	No
Default	SAML_FALSE

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 SAML_FALSE (signature processing enabled).

SAML_DSIG_ALGO

Type	String
Applies to	SAML 2.0 Authentication, Service Provider
Required	No
Default	1

Description

Specifes the XML Federation Signature algorithm with one of the following values:

- 1 = RSAwithSHA1 (default)
- 2 = RSAwithSHA256

SAML_DSIG_VERINFO_ISSUER_DN

Type	String
Applies to	SAML 2.0 Authentication, Service Provider

Required	With SAML 2.0 Authentication Schemes: Required only if SAML_DISABLE_SIGNATURE_PROCESSING is SAML_FALSE and one or both of the following are SAML_TRUE: SAML_SLO_REDIRECT_BINDING SAML_ENABLE_SSO_POST_BINDING ■ With Service Providers: Required only if SAML_DISABLE_SIGNATURE_PROCESSING is SAML_FALSE and one or both of the following are SAML_TRUE: ■ 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

Type	String
Applies to	SAML 2.0 Authentication, Service Provider
Required	With SAML 2.0 Authentication Schemes: Required only if SAML_DISABLE_SIGNATURE_PROCESSING is SAML_FALSE and one or both of the following are SAML_TRUE: ■ SAML_SLO_REDIRECT_BINDING ■ SAML_ENABLE_SSO_POST_BINDING With Service Providers: Required only if SAML_DISABLE_SIGNATURE_PROCESSING is SAML_FALSE and one or both of the following are SAML_TRUE: ■ 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

Type	Boolean
Applies to	SAML 2.0 Authentication, Service Provider
Required	No
Default	SAML_FALSE

Description

Specifies whether artifact binding is supported by the Service Provider and enabled by the Identity Provider.

SAML_ENABLE_SSO_POST_BINDING

Type	Boolean
Applies to	SAML 2.0 Authentication, Service Provider
Required	No
Default	SAML_FALSE

Description

Specifies whether HTTP POST binding is supported by the Service Provider and enabled by the Identity Provider.

See also SAML_DSIG_VERINFO_ISSUER_DN and SAML_DSIG_VERINFO_SERIAL_NUMBER.

SAML_ENABLED

Type	Boolean
Applies to	Service Provider
Required	No
Default	SAML_TRUE

Description

Specifies whether the Service Provider is activated.

SAML_IDP_AD_SEARCH_SPEC

Type	String
Applies to	SAML 2.0 Affiliation, SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Authentication
Required	Yes, if SAML_ENABLE_SSO_ARTIFACT_BINDING is SAML_TRUE
Default	None

Description

A URL specifying the default artifact resolution service for the Identity Provider.

SAML_IDP_BACKCHANNEL_AUTH_TYPE

Type	Integer
Applies to	SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Affiliation, SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Affiliation, SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Affiliation, SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Authentication
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

Type	Integer
Applies to	SAML 2.0 Authentication
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

Type	Integer
Applies to	SAML 2.0 Authentication
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

Type	Integer
Applies to	SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Authentication
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

Type	Boolean
Applies to	SAML 2.0 Authentication
Required	No
Default	SAML_FALSE

Description

Specifies whether the assertion selected for authentication must be encrypted. If this property is SAML_TRUE and the authentication scheme is passed an unencrypted assertion, the assertion cannot be authenticated.

SAML_IDP_REQUIRE_ENCRYPTED_NAMEID

Type	Boolean
Applies to	SAML 2.0 Authentication
Required	No
Default	SAML_FALSE

Description

Specifies whether the Name ID of the principal contained in the assertion must be encrypted. If this property is SAML_TRUE and the Name ID is not encrypted, the assertion cannot be authenticated.

SAML_IDP_SAMLREQ_ATTRIBUTE_SERVICE

Type	String
Applies to	SAML 2.0 Authentication
Required	No
Default	None

Description

The URL of the Attribute Service on the Attribute Authority.

SAML_IDP_SAMLREQ_ENABLE

Type	Boolean
Applies to	SAML 2.0 Authentication
Required	Yes
Default	False (0)

Description

Indicates whether the SAML Requester is enabled.

SAML_IDP_SAMLREQ_NAMEID_ALLOWED_NESTED

Type	Boolean
Applies to	SAML 2.0 Authentication
Required	No
Default	False (0)

Description

Indicated whether nested groups are allowed when selecting a DN attribute for the name identifier.

SAML_IDP_SAMLREQ_NAMEID_ATTR_NAME

Type	String

Applies to	SAML 2.0 Authentication
Required	No
Default	None

Description

The attribute name (User or DN) that holds the name when NameIdType is set to 1 or 2. If NameIdType is set to 1 or 2, NameIdAttrName must have a valid value.

SAML_IDP_SAMLREQ_NAMEID_DN_SPEC

Type	String
Applies to	SAML 2.0 Authentication
Required	No
Default	None

Description

The DN specification used when NameIdType is set to 2. When this is the case, NameIdDNSpec must have a valid value.

SAML_IDP_SAMLREQ_NAMEID_FORMAT

Type	Boolean
Applies to	SAML 2.0 Authentication
Required	No
Default	None

Description

The URI for a SAML 2.0 name identifier.

SAML_IDP_SAMLREQ_NAMEID_STATIC

Type	String
Applies to	SAML 2.0 Authentication
Required	No

Default	None

Description

The static text to be used as the name identifier when the NameIdType is set to 0. When this is the the case, a valid value must be specified for NameIdStatic.

SAML_IDP_SAMLREQ_NAMEID_TYPE

Type	Integer
Applies to	SAML 2.0 Authentication
Required	No
Default	1

Description

Represents the type of name identifier:

0 -- static text

1 -- user attribute

2 -- DN attribute

SAML_IDP_SAMELREQ_REQUIRE_SIGNED_ASSERTION

Type	Boolean
Applies to	SAML 2.0 Authentication
Required	No
Default	False (0)

Description

Indicates whether the assertion returned in response to an attribute query must be signed.

SAML_IDP_SAMELREQ_SIGN_ATTRIBUTE_QUERY

Type	Boolean
Applies to	SAML 2.0 Authentication

Required	No
Default	None

Description

Indicates whether the attribute query must be signed.

SAML_IDP_SIGN_AUTHNREQUESTS

Type	Boolean
Applies to	SAML 2.0 Authentication
Required	No
Default	False (0)

Description

Specifies whether authentication requests are signed.

SAML_IDP_SPID

Type	String
Applies to	SAML 2.0 Authentication
Required	Yes
Default	None

Description

The unique provider ID of the Service Provider being protected by this authentication scheme.

SAML_IDP_SPNAME

Type	String
Applies to	SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Authentication
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

Type	Boolean
Applies to	SAML 2.0 Authentication
Required	No
Default	SAML_TRUE

Description

Specifies whether to enforce a single-use policy for HTTP POST binding.

Setting this property to SAML_TRUE (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.

SAML_IDP_SSO_REDIRECT_MODE

Type	Integer
Applies to	SAML 2.0 Authentication

Required	No
Default	0

Description

Specifies the method by which response attribute information is passed when the user is redirected to the target resource. 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.
- 3. Persist Attributes — Attributes can be retrieved.

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

Type	String
Applies to	SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Affiliation, SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Affiliation, SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Affiliation
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

Type	String
Applies to	SAML 2.0 Authentication
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

Type	String
Applies to	SAML 2.0 Authentication
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

Type	String
Applies to	Service Provider
Required	Yes
Default	None

Description

The unique provider ID of this Service Provider.

SAML_MAJOR_VERSION

Type	Integer
Applies to	SAML 2.0 Affiliation, SAML 2.0 Authentication, Service Provider
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

Type	Integer
Applies to	SAML 2.0 Affiliation, SAML 2.0 Authentication, Service Provider
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

Type	String
Applies to	SAML 2.0 Affiliation, SAML 2.0 Authentication, Service Provider
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

Type	String
Applies to	SAML 2.0 Affiliation
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

Type	Integer
Applies to	SAML 2.0 Authentication, Service Provider
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 positive integer.

SAML_SLO_REDIRECT_BINDING

Type	Boolean
Applies to	SAML 2.0 Authentication, Service Provider
Required	No
Default	SAML_FALSE

Description

Specifies whether HTTP redirect binding is supported for single logout.

See also SAML_DSIG_VERINFO_ISSUER_DN and SAML_DSIG_VERINFO_SERIAL_NUMBER.

SAML_SLO_SERVICE_CONFIRM_URL

Type	String
Applies to	SAML 2.0 Authentication, Service Provider
Required	No
Default	None

Description

The URL where a user is redirected after single logout is completed.

SAML_SLO_SERVICE_RESPONSE_URL

Type	String
Applies to	SAML 2.0 Authentication, Service Provider
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

Type	String
Applies to	SAML 2.0 Authentication, Service Provider
Required	Yes, if SAML_SLO_REDIRECT_BINDING is SAML_TRUE
Default	None

Description

With HTTP-Redirect bindings, the Identity Provider URL where single logout requests are sent.

SAML_SLO_SERVICE_VALIDITY_DURATION

Type	Integer
Applies to	SAML 2.0 Authentication, Service Provider
Required	No
Default	60 (applies if a value is not provided and SAML_SLO_REDIRECT_BINDING is SAML_TRUE)

Description

The number of seconds for which a single logout request is valid.

The value provided must be a positive integer.

See also SAML_SKEWTIME.

SAML_SP_ARTIFACT_ENCODING

Type	String
Applies to	Service Provider
Required	No
Default	FORM (applies if a value is not provided and SAML_ENABLE_SSO_ARTIFACT_BINDING is SAML_TRUE)

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

Type	String
Applies to	Service Provider
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_ATTRSVC_AD_SEARCH_SPEC

Type	String
Applies to	Service Provider
Required	No
Default	None

Description

Search specification for an AD directory.

SAML_SP_ATTRSVC_CUSTOM_SEARCH_SPEC

Type	String
Applies to	Service Provider
Required	No
Default	None

Description

Search specification for a custom directory.

SAML_SP_ATTRSVC_ENABLE

Type	Boolean
Applies to	Service Provider
Required	No
Default	SAML_FALSE

Description

Indicates whether the Attribute Authority is enabled.

SAML_SP_ATTRSVC_LDAP_SEARCH_SPEC

Type	String
Applies to	Service Provider
Required	No
Default	None

Description

Search sepcification for an LDAP directory.

SAML_SP_ATTRSVC_ODBC_SEARCH_SPEC

Type	String
Applies to	Service Provider

Required	No
Default	None

Description

Search specification for an ODBC directory.

SAML_SP_ATTRSVC_REQUIRE_SIGNED_QUERY

Type	Boolean
Applies to	Service Provider
Required	No
Default	None

Description

Indicates whether the attribute query must be signed.

SAML_SP_ATTRSVC_SIGN_ASSERTION

Type	Boolean
Applies to	Service Provider
Required	No
Default	None

Description

Indicates whether the SAML Assertion should be signed.

SAML_SP_ATTRSVC_SIGN_RESPONSE

Type	Boolean
Applies to	Service Provider
Required	No
Default	None

Description

Indicates whether the SAML Response should be signed.

SAML_SP_ATTRSVC_VALIDITY_DURATION

Type	Integer
Applies to	Service Provider
Required	No
Default	60

Description

Specifies the number of seconds for which an assertion is valid.

SAML_SP_ATTRSVC_WINNT_SEARCH_SPEC

Type	String
Applies to	Service Provider
Required	No
Default	None

Description

Search specificatin for a WinNT directory.

SAML_SP_AUTHENTICATION_LEVEL

Type	Integer
Applies to	Service Provider
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_AUTHENTICATION_URL

Type	String
Applies to	Service Provider
Required	Yes
Default	None

Description

The protected URL for authenticating users of this Service Provider.

SAML_SP_AUTHN_CONTEXT_CLASS_REF

Type	String
Applies to	Service Provider
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

Type	String
Applies to	Service Provider
Required	Yes, if SAML_SP_ENABLE_IPD is SAML_TRUE
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

Type	String
Applies to	Service Provider
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

Type	String
Applies to	Service Provider
Required	Yes
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

Type	Boolean
Applies to	Service Provider
Required	No
Default	SAML_FALSE

Description

Specifies whether the Identity Provider Discovery profile is enabled.

SAML_SP_ENCRYPT_ASSERTION

Type	Boolean
Applies to	Service Provider
Required	No
Default	SAML_FALSE

Description

Specifies whether to encrypt the generated assertion at the Service Provider site. By default, the assertion is not encrypted.

SAML_SP_ENCRYPT_BLOCK_ALGO

Type	String
Applies to	Service Provider
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

Type	String
Applies to	Service Provider
Required	Yes, in either of the following circumstances: ■ If either of the following is SAML_TRUE: 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

Type	String
Applies to	Service Provider
Required	Yes, in either of the following circumstances: ■ If either of the following is SAML_TRUE: 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

Type	Boolean
Applies to	Service Provider
Required	No
Default	SAML_FALSE

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.

SAML_SP_ENCRYPT_KEY_ALGO

Type	String
Applies to	Service Provider
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

Type	Long (stored as decimal string)
Applies to	Service Provider
Required	No
Default	None

Description

The time by which an assertion must be generated.

This property is used with SAML_SP_STARTTIME and with the pszTimeGrid field of the Sm_PolicyApi_SAMLSP_t structure to define time restrictions for the generation of assertions.

This value is created from standard time_t values. However, it is stored as a decimal string. If you need to use SAML_SP_ENDTIME as data type long, be sure to convert it.

Set SAML_SP_ENDTIME to 0 to end the time restriction immediately.

SAML_SP_IDP_SOURCEID

Type	String
Applies to	Service Provider
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

Type	String
Applies to	Service Provider
Required	Yes
Default	None

Description

The provider ID of the Identity Provider that generates the assertions.

SAML_SP_IGNORE_REQ_AUTHNCONTEXT

Type	String
Applies to	Service Provider
Required	No
Default	0

Description

Specifies that the Identity Provider ignores "RequestedAuthnContext" in an AuthnRequest message (value of 1), or not (value of 0).

SAML_SP_IPD_SERVICE_URL

Type	String
Applies to	Service Provider
Required	Yes, if SAML_SP_ENABLE_IPD is SAML_TRUE
Default	None

Description

The host URL for the Identity Provider Discovery profile.

SAML_SP_NAMEID_ATTRNAME

Type	String
Applies to	SAML 2.0 Affiliation, Service Provider
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

Type	String
Applies to	SAML 2.0 Affiliation, Service Provider
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.

You can allow SiteMinder to search for attributes in nested groups. For information, see the description of the pszValue field of the structure Sm_PolicyApi_SAMLSPAttr_t.

SAML_SP_NAMEID_FORMAT

Type	String
Applies to	SAML 2.0 Affiliation, Service Provider
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

Type	String
Applies to	SAML 2.0 Affiliation, Service Provider
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

Type	Integer
Applies to	SAML 2.0 Affiliation, Service Provider
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

Type	Boolean
Applies to	Service Provider
Required	No

Default	SAML_FALSE

Description

Specifies whether an element indicating that the Assertion should be used only once is added to the Assertion.

SAML_SP_PASSWORD

Type	String
Applies to	Service Provider
Required	Yes, if SAML_ENABLE_SSO_ARTIFACT_BINDING is SAML_TRUE
Default	None

Description

The password to use for Service Provider access through the back channel.

SAML_SP_PERSISTENT_COOKIE

Type	Boolean
Applies to	Service Provider
Required	No
Default	SAML_FALSE

Description

Specifies whether an Identity Provider Discovery profile cookie should be persistent. Applies only if SAML_SP_ENABLE_IPD is SAML_TRUE.

SAML_SP_PLUGIN_CLASS

Type	String
Applies to	Service Provider
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 online SiteMinder Java API Documentation (Javadoc and Guide).

SAML_SP_PLUGIN_PARAMS

Type	String
Applies to	Service Provider
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

Type	Boolean
Applies to	Service Provider
Required	No
Default	SAML_FALSE

Description

Specifies whether authentication requests must be signed.

SAML_SP_REUSE_SESSION_INDEX

Type	Boolean
Applies to	Service Provider
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_STARTTIME

Type	Long (stored as decimal string)
Applies to	Service Provider
Required	No
Default	None

Description

The time when a time restriction for generating an assertion becomes effective. This value is stored in standard time_t format.

This property is used with SAML_SP_ENDTIME and with the pszTimeGrid field of the Sm_PolicyApi_SAMLSP_t structure to define time restrictions for the generation of assertions.

This value is created from standard time_t values. However, it is stored as a decimal string. If you need to use SAML_SP_STARTTIME as data type long, be sure to convert it.

Set SAML_SP_STARTTIME to 0 to start the time restriction immediately.

SAML_SP_VALIDITY_DURATION

Type	Integer
Applies to	Service Provider
Required	No

Default	60

Description

The number of seconds for which a generated assertion is valid.

The value provided must be a positive integer.

See also SAML_SKEWTIME.

SAML_SP_SESSION_NOTORAFTER_TYPE

Type	String
Applies to	Service Provider
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 your own value for the SessionNotOnOrAfter parameter in the assertion. If you select this option, enter a time in the SAML_SP_CUSTOM_TIME_OUT property.

SAML_SSOECPPROFILE

Type	Boolean
Applies to	SAML 2.0 Authentication, Service Provider
Required	No
Default	SAML_FALSE

Description

Specifies whether the Identity Provider or Service Provider supports SAML 2.0 Enhanced Client and Proxy profile requests.

SAML2_CUSTOM_ENABLE_INVALID_REQUEST_URL

Type	Boolean
Applies to	Custom error pages
Required	No
Default	None

Description

Specifies whether the custom error redirect process is enabled for an invalid request.

SAML2_CUSTOM_ENABLE_SERVER_ERROR_URL

Type	Boolean
Applies to	Custom error pages
Required	No
Default	None

Description

Specifies whether the custom error redirect process is enabled for a server error.

SAML2_CUSTOM_ENABLE_INVALID_REQUEST_URL

Type	Boolean

Applies to	Custom error pages
Required	No
Default	None

Description

Specifies whether the custom error redirect process is enabled for an invalid request.

SAML2_CUSTOM_INVALID_REQUEST_REDIRECT_MODE

Type	Boolean
Applies to	Custom error pages
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

Type	String
Applies to	Custom error pages
Required	No
Default	None

Description

- Specifies the redirect URL for an invalid request.

SAML2_CUSTOM_SERVER_ERROR_REDIRECT_MODE

Type	Boolean

Applies to	Custom error pages
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

Type	String
Applies to	Custom error pages
Required	No
Default	None

Description

Specifies the redirect URL for an internal server error .

SAML2_CUSTOM_UNAUTHORIZED_ACCESS_REDIRECT_MODE

Type	Boolean
Applies to	Custom error pages
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

Type	String
Applies to	Custom error pages
Required	No
Default	None

Description

Specifies the redirect URL for a forbidden access error.

Index