IBM Security Access Manager for Web
Version 8.0

Administration C API Developer Reference

IBM Security Access Manager for Web
Version 8.0

Administration C API Developer Reference

Note
Before using this information and the product it supports, read the information in "Notices" on page 319.

Edition notice

Note: This edition applies to version 8.0.0.2 of IBM Security Access Manager for Web (product number 5725-L52) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 12. Administration C API reference 55

Tables

About this publication

IBM Security Access Manager for Web, formerly called IBM Tivoli Access Manager for e-business, is a user authentication, authorization, and web single sign-on solution for enforcing security policies over a wide range of web and application resources.

This reference contains information about how to use Security Access Manager C administration API to enable an application to programmatically perform Security Access Manager administration tasks. This document describes the C implementation of the Security Access Manager administration API. See the *IBM Security Access Manager for Web: Administration Java Classes Developer Reference* for information regarding the Java™ implementation of these APIs.

Information on the `pdadmin` command-line interface (CLI) can be found in the *IBM Security Access Manager for Web: Command Reference*.

Access to publications and terminology

This section provides:

- A list of publications in the "IBM Security Access Manager for Web library."
- Links to "Online publications."
- A link to the "IBM Terminology website" on page x.

IBM Security Access Manager for Web library

The following documents are available online in the IBM Security Access Manager for Web library:

- *IBM Security Access Manager for Web Configuration Guide*, SC27-6205-00
- *IBM Security Access Manager for Web Administration Guide*, SC27-6207-00
- *IBM Security Access Manager Appliance Administration Guide*, SC27-6206-00
- *IBM Security Access Manager for Web Auditing Guide*, SC27-6208-00
- *IBM Security Access Manager for Web Troubleshooting Guide*, GC27-6209-00
- *IBM Security Access Manager for Web Error Message Reference*, GC27-6210-00

Online publications

IBM posts product publications when the product is released and when the publications are updated at the following locations:

IBM Security Access Manager for Web library
The product documentation site (http://pic.dhe.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.ammob.doc_8.0.0/welcome.html) displays the welcome page and navigation for the library.

IBM Security Systems Documentation Central
IBM Security Systems Documentation Central provides an alphabetical list of all IBM Security Systems product libraries and links to the online documentation for specific versions of each product.

IBM Publications Center
The IBM Publications Center site (http://www.ibm.com/e-business/

linkweb/publications/servlet/pbi.wss) offers customized search functions to help you find all the IBM publications you need.

IBM Terminology website

The IBM Terminology website consolidates terminology for product libraries in one location. You can access the Terminology website at http://www.ibm.com/software/globalization/terminology.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. You can use the keyboard instead of the mouse to operate all features of the graphical user interface.

For additional information, see the IBM Accessibility website at http://www.ibm.com/able/.

Technical training

For technical training information, see the following IBM Education website at http://www.ibm.com/software/tivoli/education.

Support information

IBM Support provides assistance with code-related problems and routine, short duration installation or usage questions. You can directly access the IBM Software Support site at http://www.ibm.com/software/support/probsub.html.

IBM Security Access Manager for Mobile Troubleshooting Guide provides details about:

- What information to collect before contacting IBM Support.
- The various methods for contacting IBM Support.
- How to use IBM Support Assistant.
- Instructions and problem-determination resources to isolate and fix the problem yourself.

Note: The **Community and Support** tab on the product information center can provide additional support resources.

Statement of Good Security Practices

IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM DOES NOT WARRANT THAT ANY SYSTEMS, PRODUCTS OR SERVICES ARE IMMUNE FROM, OR WILL MAKE YOUR ENTERPRISE IMMUNE FROM, THE MALICIOUS OR ILLEGAL CONDUCT OF ANY PARTY.

Chapter 1. Introduction to the administration API

You can use the Security Access Manager application developer kit (ADK) component to enable your application to programmatically administer Security Access Manager users and data.

The IBM Security Access Manager for Web (Security Access Manager) administration API component provides a set of functions for the administration of Security Access Manager users and data objects.

The API provides a way for applications to administer users, groups, protected objects, access control lists, protected object policies, and web resources.

This chapter contains the following topics:

Administration API overview

This section describes the types of objects that you can administer with the administration API.

You can use the administration API to administer the following types of objects:

- Policies
- Users
- Groups
- Access control lists (ACLs)
- Action groups
- Protected object policies (POPs)
- Protected objects
- Protected object spaces
- Authorization rules
- Domains
- Web resources
- Web resource groups
- Resource credentials

The administration API provides a set of functions for creating, modifying, examining, and deleting each of the preceding object types. The API also defines data types to represent each object type. The API includes the function calls necessary for manipulating each of the data types.

The administration API communicates directly with the Security Access Manager policy server component. The API establishes an authenticated, Secure Sockets

Layer (SSL) session with the Security Access Manager policy server process. When the SSL session is established, the API can send administration requests to the policy server.

The Security Access Manager policy server component services these requests in the same manner that it would service any other incoming requests.

System administrators also can use the **pdadmin** and `svrsslcfg` command-line interfaces to accomplish Security Access Manager administration tasks. The administration API functions map closely to these commands. Appendix C, "Administration API equivalents," on page 301 describes the commands that match administration API functions. Some administration API functions do not have a **pdadmin** or `svrsslcfg` command-line equivalent.

Security Access Manager APIs are thread-safe. Use caution when performing operations on objects with multiple threads; for example, if you want to create, modify and delete an ACL, and the delete is done before modifying, an error is returned.

Administration API components

This section identifies the components of the administration API.

The administration API consists of the following components:

- The administration API shared library
- The administration API header file
- The administration API library to link against (Microsoft Windows only)
- A demonstration application
- Makefiles for the demonstration application

The administration API shared libraries are distributed in the Security Access Manager runtime environment for each platform. The balance administration API components are distributed in the Security Access Manager ADK component.

The following sections provide more information about the shared libraries and ADK.

Administration API shared libraries

The administration API shared library is distributed in the Security Access Manager runtime environment component.

Table 1 lists the names of the shared libraries on each platform.

Table 1. Shared libraries

Platform	Shared Library Name
Solaris Operating Environment	`libpdadminapi.so`
IBM® AIX®	`libpdadminapi.a`
Microsoft Windows	`pdadminapi.dll`
Linux	`libpdadminapi.so`

Administration API application development kit

This section describes the administration API application development kit (ADK).

The ADK files are installed as part of the Security Access Manager ADK component package.

The ADK component contains files that can be placed anywhere on your system. Table 2 lists the files and suggests an installation directory (beneath the Security Access Manager installation directory) for each file.

Table 2. Administration API application developer kit files

Suggested Directory	File to Include	File Description
`include`	`ivadminapi.h`	The C header file containing the administration API function declarations.
`include`	`ivadmin_deprecated.h`	The C header file containing the prototypes and declarations for the functions, variables, and attributes that are deprecated in this version of Security Access Manager. Avoid including this header file as the symbols provided in it will be removed in a future release of the product.
`lib`	`pdadminapi.lib`	The library against which to link on the Microsoft Windows platform.

`admin_demo`	`pdadminapi_demo.c`
	`Makefile`
	`README.pdadminapi`
		This ADK provides a demonstration program and a sample makefile for each supported platform. You can place the demonstration program in any directory. The readme file explains how to build the demonstration program.

Building applications with the administration API

To develop applications that use the Security Access Manager administration API, you must install the required software and then link with appropriate libraries.

Software requirements

This section describes the software requirements for using the administration C API.

You must install and configure a Security Access Manager secure domain. If you do not have a Security Access Manager secure domain installed, install one before beginning application development. The minimum installation consists of a single system with the following Security Access Manager base components installed:

- Security Access Manager runtime environment
- Security Access Manager policy server
- Security Access Manager ADK

All systems in the Security Access Manager secure domains that have the runtime environment installed must have the IBM Global Security Kit (GSKit) component installed on them. If the policy server is using an LDAP server as the user registry, the Tivoli Directory Server client also must be installed on the system.

For detailed installation instructions, see the section of the *IBM Security Access Manager for Web: Installation Guide* relating to your operating system platform.

You might already have a Security Access Manager secure domain installed and want to add a development system to the domain. The minimum Security Access Manager installation consists of the following components:

- Security Access Manager runtime environment
- Security Access Manager ADK

Linking required libraries

This section provides tips for linking the required libraries.

To compile applications that use the administration API, you must install the Security Access Manager Application Developer Kit (ADK) component on the build computer.

When compiling your application on Windows systems, make sure that you add the `include` directory for the Windows library to the compiler command line.

When linking your application, specify the directory that contains the administration API shared library if it is not in the default location. You must explicitly link against the shared library.

Tested compilers

This section lists compilers that IBM tested for use with the Security Access Manager Application Developer Kit (ADK).

IBM tested the use of the Security Access Manager Application Developer Kit (ADK) component with the compilers listed in Table 3. Previous versions of the compilers listed are not supported. Compilers on other supported platforms were not tested.

Table 3. Compilers tested with Security Access Manager

Operating system platform tested	Tested compiler
IBM AIX 6.1	IBM XL C/C++ Version 10.1
Sun Solaris 11 Operating System	Oracle Solaris Studio Version 12.3
Red Hat Enterprise Linux Server release 5.6 64 bit x86 SUSE Linux Enterprise Server 10 SP3 on 64-bit System z®	GNU GCC 4.1.2
Microsoft Windows Server 2008 R2 Enterprise	Microsoft Visual Studio 2005 (using `vcvarsall.bat` AMD64)

Administration API example program

The Security Access Manager administration API ADK includes source for an example program that demonstrates use of the administration API.

The example program demonstrates how to do the following tasks:

- Initialize an administration API security context
- Display an error message
- Create a new Security Access Manager user
- Set a user account to be valid
- Change the password of the new user
- Create a new group
- Add the new user to the group
- Delete a group
- Delete a user
- Delete the administration API security context

See the sample makefile supplied with the sample program for build instructions specific to each supported operating system platform.

Deploying an administration API application

This section explains how to deploy an administration C API application.

Applications that have been developed with the Security Access Manager administration API must be run on systems that are configured as part of a Security Access Manager secure domain.

To run an administration API application, you must have installed the Security Access Manager runtime environment.

The Security Access Manager runtime environment requires that the Tivoli Directory Server client be installed on the application deployment system if an LDAP server is being used as the user registry.

Administration API applications use the SSL protocol to communicate with the Security Access Manager policy server. IBM Global Security Kit (GSKit) provides the necessary SSL support and is installed as part of the product installation.

Note: The Security Access Manager runtime environment installation enforces installation of the required software. For installation instructions, see the appropriate section in the *IBM Security Access Manager for Web: Installation Guide* for your operating system.

Gathering problem determination information

This section explains how to enable tracing for gathering troubleshooting information.

When developing an administration application, you might encounter a problem with Security Access Manager. To assist Tivoli® support personnel in diagnosing your problem, gather problem determination information relating to your error.

Security Access Manager components can be configured to log information to one or more trace files. You can enable tracing for the policy server, or any system with Security Access Manager runtime environment.

Enabling tracing on the policy server

This section explains how to enable tracing on the policy server.

To enable tracing on the policy server, edit the `/etc/routing` file, in the installation directory for the Security Access Manager policy server. Uncomment the last line.

Shut down and restart the policy server daemon, `pdmgrd`.

Enabling tracing on a system with the runtime component

This section explains how to enable tracing with the runtime component.

To enable tracing on the system where the error is occurring, edit the `/etc/routing` file, in the installation directory for the Security Access Manager runtime component. Uncomment the last line.

Restart the application that encountered the error, or re-enter the **pdadmin** command that failed. After the failure occurs again, gather the trace logs as outlined in the next section.

Gathering trace and message logs

This section explains how to gather trace logs and message logs for problem determination.

Trace and message log files for the policy server and Security Access Manager runtime environments are written to the `/log` directory. The `/log` is in the Security Access Manager installation directory. If the Tivoli Common Directory is used, the log files are located under the `HPD` directory in the Tivoli Common Directory.

To determine the names of the trace log files, you need to determine the process identifier (PID) of the Security Access Manager policy or authorization server. This information is recorded in files called `ivmgrd.pid` and `ivacld.pid`.

To determine the PID for the policy server, check the contents of the `ivmgrd.pid` file:

```
cat ivmgrd.pid
```

Similarly, check the `ivacld.pid` file for the PID of the authorization server.

After determining the PID, look in the *AM_BASE*`/log` directory for trace files with names of the form `trace__pdmgrd.`*PID*`_trace_utf8.log` for the policy server, or `trace__pdacld.`*PID*`_trace_utf8.log` for the authorization server. Also collect the following message files in the same directory:

```
msg__verbose.log
msg__notice.log
msg__fatal.log
msg__warning.log
msg__error.log
```

For additional information about logging and tracing, see the *IBM Security Access Manager for Web: Troubleshooting Guide*. For error messages, see the *IBM Security Access Manager for Web: Error Message Reference*.

Chapter 2. Administration API tasks

Each application that uses the administration API must do certain tasks necessary for API initialization, cleanup, memory management, and error handling.

The administration API provides functions for each of these tasks.

The following sections in this chapter describe the supported functions:

Establishing security contexts

To use the administration API, establish a Secure Sockets Layer (SSL) connection between the administration API application and the IBM Security Access Manager for Web (Security Access Manager) policy server.

The administration API terms this connection a *security context*.

The security context provides for the secure transfer of requests and data between the administration API application and the Security Access Manager policy server.

Call the function `ivadmin_context_createdefault2()` to create a context with the default SSL configuration. The default SSL configuration is the SSL configuration used by the Security Access Manager policy server.

The function `ivadmin_context_createdefault2()` automatically accesses the following Security Access Manager policy server configuration information:

- SSL key-ring file location
- SSL key-ring stash file location
- Security Access Manager policy server host name
- Security Access Manager policy server listening port

When `ivadmin_context_createdefault2()` is run on the same system as the Security Access Manager policy server, the preceding information is obtained from Security Access Manager configuration files.

The `ivadmin_context_createdefault2()` might be run on another system in the Security Access Manager secure domain that does not run the Security Access Manager policy server. The preceding information is obtained from stored information that was provided by the system administrator when the Security Access Manager runtime environment was configured.

There are two other functions that can be used for creating a security context. The `ivadmin_context_create3()` function creates a security context with SSL configuration information provided with the function call, instead of with same SSL configuration as the policy server. The `ivadmin_context_createlocal()` function creates a local context. Unlike other security contexts, a local context does not establish communication with any servers. A local context can be used only for manipulating configuration files with `ivadmin_cfg_*` functions.

A security context must be deleted with `ivadmin_context_delete()` function when no longer needed. Free any storage associated with the security context, including the context pointer, with `ivadmin_free()` function.

The following sections further describe how to create a security context.

Required input parameters

The administrator ID and password must be established before calling `ivadmin_context_createdefault2()`.

The administrator ID account and password are established during initial configuration of the Security Access Manager runtime environment.

You must provide the following information as input parameters when you call `ivadmin_context_createdefault2()` :

- The administrator ID to use when authenticating

  The administrator ID is the Security Access Manager administrator ID. Security Access Manager uses the underlying user registry to maintain this information.
- The password for the administrator
- The domain name

  The name of the domain to which the administrator ID belongs.

Handling of character data

Security Access Manager represents all internal character data in Unicode Transformation Format 8 (UTF-8).

The use of UTF-8 ensures that character data is handled the same regardless of what language or code page is used by the policy server. UTF-8 ensures that character data is handled the same by the authorization server, the user registry, or an application. The handling of character data is determined by the security context. The code set option on the `ivadmin_context_create3()` and `ivadmin_context_createlocal()` functions can be used to indicate how character data is handled.

The code set option of IVADMIN_CODESET_LOCAL indicates that input character data is encoded with current code page. Input character data must be converted to UTF-8 by Security Access Manager before use. Output character data is converted back into the local code page before being returned. This default handling of character data occurs if the security context was created with a function other than `ivadmin_context_create3()` and `ivadmin_context_createlocal()` and the local code page is not UTF-8.

The code set option of IVADMIN_CODESET_UTF8 indicates that the character data is already encoded in UTF-8. No input or output translation needs to be done.

If the local code page is UTF-8, no conversion of character data is done by Security Access Manager, regardless of how the security context was created.

Returned objects

The function `ivadmin_context_createdefault2()` returns the following data:

- A pointer to a context object of type `ivadmin_context`

  The context object contains all the information necessary to establish an SSL connection with the Security Access Manager policy server.
- A pointer to a response object of type `ivadmin_response`

  The response object contains information about any errors that are generated by administration API function calls.

Example code

The following code fragment shows an example call of `ivadmin_context_createdefault2()` with the administrator ID `sec_master`:

```
ivadmin_context ctx;
ivadmin_response rsp;
unsigned long status;

status = ivadmin_context_createdefault2("sec_master", sec_masterpwd, domain_id,
&ctx, &rsp);
if (status!= IVADMIN_TRUE) {
/* The context create call failed so we should just exit.
* Optionally, you can insert error handling code here
*/
return 0
}
```

Compatibility with earlier versions

The administration API provides a function that can create a context: `ivadmin_context_create3()`.

This function provides compatibility with earlier versions with applications developed with older versions of IBM Tivoli Security Access Manager. Applications must use the `ivadmin_context_createdefault2()` function to create a security context.

The function `ivadmin_context_create3()` provides only a subset of the functions available in `ivadmin_context_createdefault2()`. It does not automatically determine the SSL configuration for the Security Access Manager policy server. You must manually supply the necessary SSL configuration information.

Delegating user credentials

Each security context has a set of user credentials.

The Security Access Manager policy server examines these credentials. The policy server then decides whether to allow or deny a request for access to Security Access Manager data. The credentials associated with a security context are of the user specified to the `ivadmin_context_create3()` or `ivadmin_context_createdefault2()` function.

Use the administration API function `ivadmin_context_setdelcred()` to specify an alternative user credential used by the Security Access Manager policy server to make access decisions. The specified credentials accompany all access requests in the secure context, until the credentials are cleared and set again.

The user must previously have authenticated and established credentials before the credentials can be delegated.

To call `ivadmin_context_setdelcred()`, you must supply the following input parameters:

- Privilege Attribute Certificate (PAC) data
- PAC length

You can use the Security Access Manager authorization API function `azn_creds_get_pac()` to create PAC data from a credential. For more information about the establishment and use of user credentials, see the *IBM Security Access Manager for Web: Authorization C API Developer Reference*.

You can call the function `ivadmin_context_cleardelcred()` to clear the delegated credentials. The `ivadmin_context_hasdelcred()` function can be used to determine whether a context set a delegated credential.

See the following reference sections for information about these functions:

- "ivadmin_context_setdelcred()" on page 138
- "ivadmin_context_cleardelcred()" on page 117
- "ivadmin_context_hasdelcred()" on page 137

Creating objects

You can use the administration API to create the required Security Access Manager objects to complete administration tasks.

Before you can create an object, you must establish a security context. See "Establishing security contexts" on page 7.

For example, to create a user object, supply the following information:

- A security context
- Initialization values for data specific to the object, such as a user ID
- Any policies that apply to the object, such as password enforcement policies

To create a user in the user registry, supply the following parameters to `ivadmin_user_create3()`:

```
unsigned long
ivadmin_user_create3(
  ivadmin_context ctx,       // input - security context
  const char *userid,        // input - Security Access Manager user ID
  const char *dn,            // input - user registry distinguished name
  const char *cn,            // input - user registry common name
  const char *sn,            // input - user registry attribute surname
  const char *pwd,           // input - user registry  attribute password
  unsigned long group_count, // input - Number of user registry group memberships
  const char **groups,       // input - user registry group memberships
  unsigned long ssouser,     // input - SSO credentials policy
                             //           (true/false)
  unsigned long nopwdpolicy, // input - password policy enforced
                             //           at creation (true/false)
  ivadmin_response *rsp      // output - response object
);
```

Administration API functions that create objects return error conditions within an `ivadmin_response` object.

For example, the administration API provides functions to create the following objects in Table 4.

Table 4. Creating objects

Function	Description
`ivadmin_acl_create()`	Creates an access control list.
`ivadmin_action_create()`	Creates a Security Access Manager action.
`ivadmin_action_group_create()`	Creates a Security Access Manager action group.
`ivadmin_authzrule_create()`	Creates an authorization rule.
`ivadmin_domain_create()`	Creates a secure domain.
`ivadmin_group_create2()`	Creates a Security Access Manager group.
`ivadmin_objectspace_create()`	Creates a Security Access Manager protected object space.
`ivadmin_pop_create()`	Creates a protected object policy.
`ivadmin_protobj_create()`	Creates a protected object.
`ivadmin_ssocred_create()`	Creates a single sign-on credential.
`ivadmin_ssogroup_create()`	Creates a single sign-on group resource.
`ivadmin_ssoweb_create()`	Creates a single sign-on web resource.
`ivadmin_user_create3()`	Creates a Security Access Manager user.

Setting object values

You can use the administration API to set values in the data objects from the user registry.

Use the administration API set operations in the following situations:

- To modify values just after you create and initialize an object

  For example, after creating a user in the user registry, call `ivadmin_user_setaccexpdate()` to set an account expiration date for the user.
- To modify values for existing objects

  For example, to modify the maximum password age for all user accounts, call `ivadmin_context_setmaxpwdage()` .

To do a set operation, you must have a valid context established between the administration API application and the Security Access Manager policy server.

All set operations return the following data:

- An integer value (IVADMIN_TRUE or IVADMIN_FALSE) indicating whether the operation succeeded or failed.
- An `ivadmin_response` object. This object contains information about error conditions.

Table 5 lists examples of administration API set operations.

Table 5. Example set operations

Function	Description
`ivadmin_user_setdescription()`	Sets the description for the specified user

Table 5. Example set operations (continued)

Function	Description
`ivadmin_user_setaccexpdate()`	Sets the expiration date for the specified user account
`ivadmin_context_setminpwdlen()`	Sets the minimum password length for all user accounts
`ivadmin_acl_setuser()`	Sets the entry for the user in the specified access control list
`ivadmin_pop_setauditlevel()`	Sets the level of audit reporting for the specified protected object policy
`ivadmin_protobj_settype()`	Sets the protected object type

Getting objects

The administration API defines a number of data types to contain Security Access Manager data.

You can use the administration API to obtain objects of each of the defined data types. You can then use administration API functions to examine the values contained in each object.

The administration API get operations send a request to the Security Access Manager policy server to retrieve a reference or handle to the specified object. For example, the object might be user information contained in a user registry.

The Security Access Manager policy server returns data that describes the requested object to the client application through a secure communications channel. The application then constructs a copy of the object in local memory from the returned data. Free the local memory when the Security Access Manager object is no longer needed.

Table 6 lists examples of some administration API data types that are returned by API get functions.

Table 6. Example data types returned by get functions

Function	Data Type Returned	Object Description
`ivadmin_acl_get()`	`ivadmin_acl`	Access control list
`ivadmin_pop_get()`	`ivadmin_pop`	Protected object policy
`ivadmin_user_get()`	`ivadmin_ldapuser`	User information
`ivadmin_group_get()`	`ivadmin_ldapgroup`	Group information
`ivadmin_protobj_get2()`	`ivadmin_protobj`	Protected object
`ivadmin_domain_get()`	`ivadmin_domain`	Domain
`ivadmin_authzrule_get()`	`ivadmin_authzrule`	Authorization rule
`ivadmin_ssocred_get()`	`ivadmin_ssocred`	Resource credential
`ivadmin_ssogroup_get()`	`ivadmin_ssogroup`	Resource group
`ivadmin_ssoweb_get()`	`ivadmin_ssoweb`	Single sign-on web resource

Reading object values

When you establish a context and obtain an object through a **get** operation, use the administration API to do read operations on data in the object.

For example, when the application obtains an `ivadmin_ldapuser` object, the application can use API functions to read the uses distinguished name.

Character data obtained from the object is returned in the code set specified in the security context.

For performance reasons, the administration API does not send read requests directly to the Security Access Manager policy server. Performance is optimized by completing one get transaction through the security context to obtain the relevant object. Then, querying the contents of the object after it is stored on the local system.

Table 7 shows some example operations that read values from a returned object.

Table 7. Example read operations

Functions	Description
`ivadmin_user_getcn()`	Gets the common name from the specified `ivadmin_ldapuser` object.
`ivadmin_user_getdn()`	Gets the distinguished name from the specified `ivadmin_ldapuser` object.
`ivadmin_user_getsn()`	Gets the family name of the user from the specified `ivadmin_ldapuser` object.
`ivadmin_group_getdescription()`	Gets the description entry for the group from the `ivadmin_ldapgroup` object.
`ivadmin_acl_getuser()`	Gets the actions defined for a user from the `ivadmin_acl` object.
`ivadmin_pop_getauditlevel()`	Gets the audit level defined for the protected object policy (POP) from the `ivadmin_pop` object.
`ivadmin_protobj_getaclid()`	Gets the identifier for the access control list (ACL) that is attached to the protected object from the `ivadmin_protobj` object.
`ivadmin_domain_getid()`	Gets the name of the domain from the `ivadmin_domain` object.
`ivadmin_authzrule_getid()`	Gets the ID of the specified authorization rule from the `ivadmin_authzrule` object.
`ivadmin_ssocred_gettype()`	Gets the type of single sign-on resource associated with the credential from the `ivadmin_ssocred` object.

Listing object information

Some administrative tasks require the application to obtain a list of objects of one specific type.

For example, an administrator might need to review the list of existing users to decide whether a new user must be created.

You can use the administration API list operations to accomplish tasks of this type. These operations are identical to API get operations. Both types of operations take the following actions:

- Communicate with the policy server through the secure context
- Request Security Access Manager data from the policy server

Administration API list operations differ from get operations in one important way. List operations do *not* obtain a reference to an entire data object and place it in local memory. Instead, they obtain an *array of pointers* to the relevant data type, or to character data (which are names of listed items.)

This array of pointers enables list operations to extract only the important data from much larger data structures and return it to the client application. The client application must free all the data associated with the list with the `ivadmin_free()` function when it is no longer needed.

For example, the function `ivadmin_user_list()` returns a list of user IDs in the form of an array of pointers to character strings:

```
unsigned long
ivadmin_user_list(
   ivadmin_context ctx,     // input - Context to policy server
   const char *pattern,     // input - Search pattern
   unsigned long maxreturn, // input - Maximum number of returned items
   unsigned long *count,    // output - Count of returned item
   char ***userids,         // output - Array of pointers to userIDs
   ivadmin_response *rsp    // output - Response object
);
```

Use the `ivadmin_free()` function to free the memory used by the list when it is no longer needed. You must free the data associated with each character pointer and the array of pointers.

If the list operation encounters an error, the count is set to zero and the array of pointers is set to NULL.

Handling errors

The way an administration API call indicates that an error occurred depends on how the API returns information.

For the purposes of error handling, the administration APIs can be divided into three groups:

- APIs that return a numeric return code, output arguments, and a response object, such as `ivadmin_user_list()` and `ivadmin_pop_find()`.
- APIs that return a numeric return code and output arguments, such as `ivadmin_acl_attrget()` and `ivadmin_ssogroup_getresources()`.
- APIs that return a value only, such as `ivadmin_group_getdescription()` and `ivadmin_user_getsn()`.

If an administration API call returns a numeric return code, check the return code to determine whether the API was successful. If the API was unsuccessful and a response object is available, check the response object for additional information, as described in "Evaluating a response object" on page 15.

Whether a return code is provided or not, if an administration API call was not successful, any output or return values are set to indicate that no information was returned. Pointer arguments are set to NULL and counts and numeric values are set to zero.

Evaluating a response object

Many administration API calls return a pointer to an object of type `ivadmin_response`.

```
ivadmin_response *rsp;
```

Objects of type `ivadmin_response` are termed *response objects* and provide additional information about the operation.

The response objects are initialized by the administration API to `NULL`.

If a response object is returned, examine the contents to obtain further information about the error. Use the `ivadmin_response_getok()` function to examine a response object. This function returns an unsigned long integer. This return value corresponds to one of the following constants, which are defined in `ivadminapi.h`:

```
#define IVADMIN_FALSE           0
#define IVADMIN_TRUE            1
```

- If the call encountered an error, the response object contains the constant `IVADMIN_FALSE`.
- If the validation of input parameters fails, `IVADMIN_FALSE` is returned.
- If the call succeeded, the response object contains the constant `IVADMIN_TRUE`.

When `ivadmin_response_getok()` returns `IVADMIN_FALSE`, you can use additional administration API functions to obtain information about the error.

Obtaining error message text

You can view text messages that describe an error.

Procedure

1. Call `ivadmin_response_getcount()` to determine how many error messages were returned. **Note:** Most API calls return only one error message.
2. For each message returned, call `ivadmin_response_getmessage()`. Pass in, as an input parameter, an index value for each error message. The following sample code prints the response message (character string) from an administration API command:

```
void printResponse(ivadmin_response rsp, char *api_call) {
int i=0;

  if (rsp == NULL) {
    printf("  %s : failed\n", api_call);
  }

  if (ivadmin_response_getok(rsp)) {
    printf("   %s : succeeded\n", api_call);
  } else {
    for (i=0; i<ivadmin_response_getcount(rsp); i++) {
    printf("   %s : %s\n", api_call,
           ivadmin_response_getmessage(rsp, i));
    }
  }
}
```

In the preceding example, the response (rsp) can be NULL.

Results

See the following reference pages:

- "ivadmin_response_getcount()" on page 225
- "ivadmin_response_getmessage()" on page 225

Obtaining error codes

You can display a Security Access Manager value code that corresponds to each message with ivadmin_response_getmessage().

About this task

When you know the meaning of a particular value code, you can use this information to develop application logic specific to the particular error condition.

Procedure

1. Call ivadmin_response_getcount() to determine how many error messages were returned.

   Note: Most API calls return only one error message.
2. Call ivadmin_response_getcode() with an integer argument - input parameter, specifying the error message to examine.

   The response code is returned in the form of an unsigned integer:

```
void printErrorCode(ivadmin_response rsp, char *api_call) {
int i=0;

  if (rsp == NULL) {
   printf("   %s : failed\n", api_call);
  }

  if (ivadmin_response_getok(rsp)) {
    printf("   %s : succeeded\n", api_call);
  } else {
    for (i=0; i<ivadmin_response_getcount(rsp); i++) {
      printf("   %s : %ul\n", api_call,
             ivadmin_response_getcode(rsp, i));
    }
  }
}
```

Obtaining error message modifiers

Some administration API calls return a modifier that categorizes the returned message as an Information, Warning, or Error type.

The modifiers are defined as constants (unsigned longs):

```
#define IVADMIN_RESPONSE_INFO     0
#define IVADMIN_RESPONSE_WARNING  1
#define IVADMIN_RESPONSE_ERROR    2
```

- Call ivadmin_message_getcount() to determine how many information, warning, or error messages were returned.
- Call ivadmin_response_getmodifier() to determine the modifier for the specified message:

```
unsigned long = modifier;
modifier = ivadmin_response_getmodifier(ivadmin_response rsp,
unsigned long index);
```

Cleaning up and shutting down

Cleanup and shutdown of the administration API consists of freeing the memory and deleting the security contexts.

Freeing memory

The `ivadmin_free()` function in the administration API frees memory that is allocated by administration API calls.

All memory that is allocated by administration API calls must be freed with this function.

```
void ivadmin_free(void *p);
```

Be sure to free memory allocated when you create the following objects:

- An `ivadmin_context` object
  See "Establishing security contexts" on page 7.
- A local copy of a data object created by an administration API get function
  See "Getting objects" on page 12.
- An `ivadmin_response` object that contains error information
  See "Handling errors" on page 14.

You also must free character strings and array pointers that are created by an administration API list function. Use the `ivadmin_free` function to free this memory as well. See "Listing object information" on page 13.

Deleting a security context

The administration API application must close the connection, or security context, to the Security Access Manager policy server before exiting.

The context must be deleted so that the client system and the Security Access Manager policy server can free the SSL resources.

The administration API provides the function `ivadmin_context_delete()`. This function takes the following input parameters:

- A context object of type `ivadmin_context`
- A pointer to the response object of type `ivadmin_response`

When the context is deleted, the context memory is freed. Both the `ivadmin_context` object and `ivadmin_response` object must be freed.

The following code fragment shows a sample usage of `ivadmin_context_delete()`:

```
unsigned long status:
ivadmin_context ctx;
ivadmin_response rsp;
status = ivadmin_context_delete(ctx, &rsp);

if (status != IVADMIN_TRUE) {
```

```
/* Delete failed; insert appropriate error handling */
}
ivadmin_free(rsp);
ivadmin_free(ctx);
```

Chapter 3. Users and groups

The administration API provides a collection of functions for administering IBM Security Access Manager for Web (Security Access Manager) users and groups.

This chapter describes the tasks that those functions accomplish.

Information about Security Access Manager users and groups are stored in the user registry. You can use the administration API to both modify and access user and group settings in the user registry. The administration API provides functions to administer both individual user settings and global user settings.

Security Access Manager provides the **pdadmin** command-line interface (CLI) that accomplishes many of the same user and group administration tasks. Application developers who previously used the **pdadmin** command to manage a Security Access Manager secure domain find the administration API functions straightforward to implement.

This chapter displays the **pdadmin** command-line equivalent for each of the administration API function calls. You can review the output from the **pdadmin** command-line equivalents to better understand the types of information returned by the administration APIs. See the *IBM Security Access Manager for Web: Administration Guide* for detailed information about the **pdadmin** command.

This chapter contains the following topics:

Administering users

The administration API provides functions for creating, accessing, deleting, and listing Security Access Manager user information within the user registry.

The function `ivadmin_user_create3()` creates a user in the user registry used by the Security Access Manager policy server.

Note: When a user definition exists in the user registry, use the `ivadmin_user_import2()` function instead.

The `ivadmin_user_import2()` function imports an existing user definition from the user registry into Security Access Manager and allows the user definition to be managed by Security Access Manager.

Use the `ivadmin_user_delete2()` function to delete a user from Security Access Manager.

Table 8 on page 20 lists the user administration functions.

User registry difference: Leading and trailing blanks in a user name do not make the name unique when using an LDAP or Active Directory user registry. To keep name processing consistent regardless of what user registry is being used, do not define user names with leading or trailing blanks.

Table 8. Administering users

Functions	Description
"ivadmin_user_create3()" on page 249	Creates the specified user.
"ivadmin_user_delete2()" on page 251	Deletes the specified user.
"ivadmin_user_import2()" on page 270	Creates a Security Access Manager user by importing an existing user from the user registry.
"ivadmin_user_list()" on page 271	Lists Security Access Manager users.
"ivadmin_user_listbydn()" on page 272	Lists users by with distinguished name of the user registry.

Administering user accounts

After a user account is created in the user registry, you can set and get different pieces of information about the user.

You must create a security context between the calling application and the Security Access Manager policy server before you can access the user registry. You can obtain the user registry information for a user object by specifying either the user ID or the user distinguished name.

Call the `ivadmin_user_set*` group of API functions to establish security policies that apply to *one* specific Security Access Manager user. Call the `ivadmin_context_*` group of API functions to establish security policies that apply to *all* Security Access Manager users.

A policy might be set for a specific user account by an `ivadmin_user_set*` API call. The same policy might be set globally for *all* users by an `ivadmin_context_set*` API call. In this case, the policy for the specific user account takes priority and is used. This priority takes place regardless of whether the policy for the specific user is more or less restrictive than the global policy.

Note: When both an `ivadmin_user_set*` API and an `ivadmin_context_set*` API exist with similar functionality, they are combined under the `ivadmin_context_set*` API as shown in Table 9 on page 21.

This section describes the API calls that you can use to modify or access the following data:

- Account expiration date
- Account disablement time interval
- Maximum number of failed logins
- Time of day access
- User registry type
- User objects
- User account-valid status
- User names (distinguished names, common names, and surnames)

- User descriptions
- Group memberships
- Concurrent web sessions

Table 9. Administering user accounts

Functions	Description
"ivadmin_context_getaccexpdate()" on page 123 "ivadmin_user_getaccexpdate()" on page 253	Returns the account expiration date for user accounts.
"ivadmin_context_getdisabletimeint()" on page 125 "ivadmin_user_getdisabletimeint()" on page 256	Returns the time to disable user accounts when the maximum number of login failures is exceeded.
"ivadmin_context_getmaxconcurwebsess()" on page 126 "ivadmin_user_getmaxconcurwebsess()" on page 259	Returns the maximum number of concurrent web sessions.
"ivadmin_context_getmaxlgnfails()" on page 127 "ivadmin_user_getmaxlgnfails()" on page 260	Returns the maximum number of failed logins allowed for user accounts.
"ivadmin_context_gettodaccess()" on page 135 "ivadmin_user_gettodaccess()" on page 269	Returns the time of day access policy for user accounts.
"ivadmin_context_getuserreg()" on page 136	Determines which type of user registry is configured for the Security Access Manager policy server.
"ivadmin_context_setaccexpdate()" on page 138 "ivadmin_user_setaccexpdate()" on page 273	Sets the account expiration date for user accounts.
"ivadmin_context_setdisabletimeint()" on page 139 "ivadmin_user_setdisabletimeint()" on page 276	Sets the time to disable for user accounts when the maximum number of login failures is exceeded.
"ivadmin_context_setmaxconcurwebsess()" on page 140 "ivadmin_user_setmaxconcurwebsess()" on page 277	Sets the maximum number of concurrent sessions that are allowed for user accounts.
"ivadmin_context_setmaxlgnfails()" on page 142 "ivadmin_user_setmaxlgnfails()" on page 278	Sets the maximum number of failed login attempts that are allowed for user accounts.
"ivadmin_context_settodaccess()" on page 147 "ivadmin_user_settodaccess()" on page 287	Sets the time of day access for the account for user accounts.
"ivadmin_user_get()" on page 252	Gets the user object. Takes *userID* (character string) as an input parameter. Returns an object of type **ivadmin_ldapuser**. This object contains a number of user registry attributes for the specified user.
"ivadmin_user_getaccountvalid()" on page 254	Returns the account-valid indicator for the specified user object.

Table 9. Administering user accounts (continued)

Functions	Description
"ivadmin_user_getbydn()" on page 254	Gets the user object by with distinguished name in the user registry. Returns an object of type `ivadmin_ldapuser`.
"ivadmin_user_getcn()" on page 255	Returns the common name attribute from the specified user.
"ivadmin_user_getdescription()" on page 256	Returns the user description as a character string.
"ivadmin_user_getdn()" on page 257	Returns the distinguished name from the specified user.
"ivadmin_user_getmemberships()" on page 263	Lists the groups in which the specified user is a member.
"ivadmin_user_getsn()" on page 268	Returns the surname attribute for the specified user.
"ivadmin_user_getssouser()" on page 268	Returns a setting that indicates whether the user account has single sign-on capabilities.
"ivadmin_user_setaccountvalid()" on page 274	Enables or disables the specified user account.
"ivadmin_user_setdescription()" on page 275	Sets the user description.
"ivadmin_user_setssouser()" on page 286	Enables or disables the single sign-on capabilities of the Security Access Manager user.

Administering user passwords

You can manage user access by setting password attributes. You can specify policies that apply only to a single user or that apply to all users. This section describes the administration API calls to modify or access password data and policies.

Call the `ivadmin_user_*` group of API functions to establish security policies that apply to *one* specific Security Access Manager user. Call the `ivadmin_context_*` group of API functions to establish security policies that apply to *all* Security Access Manager users.

A policy might be set for a specific user password by an `ivadmin_user_*` API call. The same policy might be set globally for *all* users by an `ivadmin_context_*` API call. In this case, the policy for the specific user password takes priority, and is used. This priority takes place regardless of whether the policy for the specific user password is more or less restrictive than the global policy.

Note: When both an `ivadmin_user_*` command and an `ivadmin_context_*` command exist with similar functionality, they are combined and alphabetized under the `ivadmin_context_*` command in Table 10 on page 23.

Table 10. Administering user passwords

Functions	Description
“ivadmin_context_getmaxpwdage()” on page 128 “ivadmin_user_getmaxpwdage()” on page 261	Returns the maximum password age for user accounts.
“ivadmin_context_getmaxpwdrepchars()” on page 129 “ivadmin_user_getmaxpwdrepchars()” on page 262	Returns the maximum number of repeated characters allowed in a password for user accounts.
“ivadmin_context_getminpwdalphas()” on page 131 “ivadmin_user_getminpwdalphas()” on page 264	Gets the minimum number of alphabetic characters allowed in a password for user accounts.
“ivadmin_context_getminpwdlen()” on page 133 “ivadmin_user_getminpwdlen()” on page 265	Returns the minimum password length for user accounts.
“ivadmin_context_setminpwdnonalphas()” on page 145 “ivadmin_user_getminpwdnonalphas()” on page 265	Returns the minimum number of non-alphabetic characters allowed in a password for user accounts.
“ivadmin_context_getpwdspaces()” on page 134 “ivadmin_user_getpwdspaces()” on page 267	Returns policy for whether spaces are allowed in passwords for user accounts.
“ivadmin_context_setmaxpwdage()” on page 142 “ivadmin_user_setmaxpwdage()” on page 279	Sets the maximum password age for user accounts.
“ivadmin_context_setmaxpwdrepchars()” on page 143 “ivadmin_user_setmaxpwdrepchars()” on page 280	Sets the maximum number of repeated characters allowed in a password for user accounts.
“ivadmin_context_setminpwdalphas()” on page 144 “ivadmin_user_setminpwdalphas()” on page 281	Sets the minimum number of alphabetic characters allowed in a password for user accounts.
“ivadmin_context_setminpwdlen()” on page 146 “ivadmin_user_setminpwdlen()” on page 282	Sets the minimum password length for user accounts.
“ivadmin_context_setminpwdnonalphas()” on page 145 “ivadmin_user_setminpwdnonalphas()” on page 283	Sets the minimum number of non-alphabetic characters allowed in a password for user accounts.
“ivadmin_context_setpwdspaces()” on page 146 “ivadmin_user_setpwdspaces()” on page 285	Sets policy for whether spaces are allowed in passwords for user accounts.
“ivadmin_user_getlastpwdchange()” on page 155	Returns the time and date of when the password was last changed.
“ivadmin_user_getpasswordvalid()” on page 266	Returns the enabled indicator for the user password.
“ivadmin_user_setpassword()” on page 283	Sets the user password.
“ivadmin_user_setpasswordvalid()” on page 284	Enables or disables the Security Access Manager user password.

Administering groups

The administration API provides functions for creating, deleting, and listing the members of a group.

The name of a group is not case-sensitive. The values “group”, “GROUP”, “Group”, and “GrOuP” all refer to the same Security Access Manager group. Table 11 on page 24 lists the group administration functions.

User registry difference: Leading and trailing blanks in a group name do not make the name unique when using an LDAP or Active Directory user registry. To keep name processing consistent regardless of what user registry is being used, do not define group names with leading or trailing blanks.

Table 11. Administering groups

Functions	Description
"ivadmin_group_create2()" on page 156	Creates a group.
"ivadmin_group_import2()" on page 163	Creates a Security Access Manager group by importing an existing group from the user registry.
"ivadmin_group_delete2()" on page 157	Deletes the specified group.
"ivadmin_group_list()" on page 164	Lists group names that match the specified pattern. Group names can be Security Access Manager or user registry names.

Administering group attributes

Use the administration API to administer the attributes of a group.

Table 12 lists the group attribute administration functions.

Table 12. Administering group attributes

Functions	Description
"ivadmin_group_get()" on page 158	Returns the group object for the specified group name.
"ivadmin_group_getbydn()" on page 159	Returns the group object for the specified distinguished name.
"ivadmin_group_getcn()" on page 160	Returns the group common name attribute for the specified group.
"ivadmin_group_getdescription()" on page 160	Returns the group description.
"ivadmin_group_getdn()" on page 161	Returns the group distinguished name for the specified group.
"ivadmin_group_getid()" on page 162	Returns the group ID for the specified group.
"ivadmin_group_listbydn()" on page 165	Lists groups that match the specified pattern for distinguished names.
"ivadmin_group_setdescription()" on page 167	Sets the group description.
"ivadmin_group_getmembers()" on page 162	Lists the members of the group.
"ivadmin_group_addmembers()" on page 155	Adds the specified users to the specified group. **User registry difference:** Attempting to add a duplicate user to a group is handled differently depending on what user registry is being used. See Appendix B, "User registry differences," on page 293 for details.

Table 12. Administering group attributes (continued)

Functions	Description
"ivadmin_group_removemembers()" on page 166	Removes the specified users from the specified group.

Chapter 4. Administering protected objects and protected object spaces

You can use the administration API to create, modify, examine, list, and delete Security Access Manager protected objects. This chapter describes the administration API functions for administering protected object spaces and protected objects.

These protected objects represent resources that must be secured to enforce your security policy. You can specify the security policy by applying access control lists (ACLs), protected object policies (POPs), and authorization rules to the protected objects.

Security Access Manager protected objects exist within a virtual hierarchy known as a *protected object space*. Security Access Manager provides several protected object spaces by default. You can use the administration API to define new regions of the protected object space. You can also use the API to define and secure resources that are specific to a third-party application.

You must be familiar with protected objects before working with the administration API. For an introduction to protected objects, see the chapter about managing protected objects in the *IBM Security Access Manager for Web: Administration Guide*.

For an introduction to the use of ACLs, POPs, and authorization rules to secure protected objects, see the chapters about using access control policies, protected object policies, and authorization rules in the *IBM Security Access Manager for Web: Administration Guide*.

This chapter contains the following topics:

Administering protected object spaces

You can use the administration API to create and administer a user-defined protected object space.

You can use this protected object space to define a resource hierarchy that is specific to a third-party application that uses Security Access Manager authorization services to enforce a security policy.

User-defined object spaces created with the administration API are dynamic because they can be updated while Security Access Manager is running.

Table 13 on page 28 lists the methods available for administering protected object spaces.

Note: For an introduction to the creation of protected object spaces, see the protected object space information in the *IBM Security Access Manager for Web: Administration Guide*.

Table 13. Administering protected object spaces

Functions	Description
“ivadmin_objectspace_create()” on page 168	Creates a Security Access Manager protected object space.
“ivadmin_objectspace_delete()” on page 169	Deletes the specified Security Access Manager protected object space.
“ivadmin_objectspace_list()” on page 170	Lists the Security Access Manager protected object spaces.

Administering protected objects

Define protected objects that reflect the resources that your security policy protects.

Security Access Manager defines two types of protected objects: container objects and resource objects. Understand these concepts before creating and administering protected objects.

The name of a protected object can be of any length and contain any character. The forward slash (/) character is interpreted to be part of the object hierarchy. ACLs can be attached at the various points indicated by the forward slash character.

After creating a protected object, you can specify a security policy for it. You define and attach ACLs, POPs, authorization rules, or any combination of these entities.

For more information about Security Access Manager security concepts, see the *IBM Security Access Manager for Web: Administration Guide*.

Use caution when implementing protected objects programmatically. In many cases, the protected object hierarchy is manually designed, built, and tested by a security expert. Carefully review the hierarchy to ensure that the security policy is correctly enforced. If you choose to build protected object hierarchies programmatically, be sure to test and review the settings for each object before deploying the security environment.

Table 14 lists the functions available to administer protected objects.

Table 14. Administering protected objects

Functions	Description
“ivadmin_protobj_attachacl()” on page 198	Attaches the specified access control list to the specified protected object.
“ivadmin_protobj_attachauthzrule()” on page 199	Attaches an authorization rule to the specified protected object.
“ivadmin_protobj_create()” on page 204	Creates a Security Access Manager protected object.
“ivadmin_protobj_delete()” on page 205	Deletes the specified Security Access Manager protected object.
“ivadmin_protobj_detachacl()” on page 205	Detaches the access control list from the specified protected object.
“ivadmin_protobj_detachauthzrule()” on page 206	Detaches an authorization rule from the specified protected object.
“ivadmin_protobj_get3()” on page 209	Gets the specified protected object.

Table 14. Administering protected objects (continued)

Functions	Description
“ivadmin_protobj_getaclid()” on page 211	Gets the name of the ACL attached to the specified protected object.
“ivadmin_protobj_geteffaclid()” on page 213	Gets the name of the ACL in effect for the specified protected object.
“ivadmin_protobj_getpopid()” on page 216	Gets the name of the POP attached to the specified protected object.
“ivadmin_protobj_geteffpopid()” on page 214	Gets the name of the POP in effect for the specified protected object.
“ivadmin_protobj_getauthzruleid()” on page 211	Gets the name of the authorization rule object that is attached to the specified protected object.
“ivadmin_protobj_geteffauthzruleid()” on page 213	Gets the name of the authorization rule object that is in effect for the specified protected object.
“ivadmin_protobj_getdesc()” on page 212	Gets the description of the specified protected object.
“ivadmin_protobj_getid()” on page 215	Gets the name of the specified protected object.
“ivadmin_protobj_getpolicyattachable()” on page 215	Indicates whether a protected object policy or access control list can be attached to the specified protected object.
“ivadmin_protobj_exists()” on page 209	Indicates whether a protected object exists.
“ivadmin_protobj_access()” on page 196	Indicates whether a specific action to a specific object is permitted.
“ivadmin_protobj_multiaccess()” on page 220	Indicates whether the specified actions to the specified objects are permitted.
“ivadmin_protobj_getpopid()” on page 216	Gets the name of the protected object policy for the specified protected object.
“ivadmin_protobj_list3()” on page 217	Returns the protected objects contained under the specified directory.
“ivadmin_protobj_listbyacl()” on page 219	Returns a list of protected objects that have the specified access control list attached.
“ivadmin_protobj_setdesc()” on page 222	Sets the description field of the specified protected object.
“ivadmin_protobj_setpolicyattachable()” on page 223	Sets whether a protected object policy or access control list can be attached to the specified protected object.
“ivadmin_protobj_settype()” on page 223	Sets the type field of the specified protected object.
“ivadmin_protobj_listbyauthzrule()” on page 219	Lists the protected objects that have the specified authorization rule attached.

Administering extended attributes for a protected object

You can create, set, query, and delete the extended attributes for a protected object.

Protected objects without explicitly defined extended attributes inherit the first found set of extended attributes, which are defined at the parent object within the inheritance chain. The found set of extended attributes replaces the empty set of defined attributes. These inherited attributes are termed *effective extended attributes*.

Table 15 describes the methods for administering extended attributes and effective extended attributes for a protected object.

Table 15. Administering protected object attributes

Functions	Description
"ivadmin_protobj_attrdelkey()" on page 199	Deletes the specified extended attribute (name and values) from the specified protected object.
"ivadmin_protobj_attrdelval()" on page 200	Deletes the specified value from the specified extended attribute key in the specified protected object.
"ivadmin_protobj_effattrget()" on page 207	Displays a list of the values for the effective extended attribute that is associated with the specified protected object.
"ivadmin_protobj_effattrlist()" on page 208	Displays a list of all the effective extended attributes that are associated with the specified protected object.
"ivadmin_protobj_attrget()" on page 201	Returns the values that are associated with the specified extended attribute for the specified protected object.
"ivadmin_protobj_attrlist()" on page 202	Lists all the extended attributes that are associated with the specified protected object.
"ivadmin_protobj_attrput()" on page 203	Creates an extended attribute with the specified name and value, if it does not exist, and adds the attribute to the specified protected object. If the attribute specified exists, the specified value is added to the existing attribute.

Chapter 5. Administering access control

You can use the administration API to create, modify, examine, list, and delete Security Access Manager access control lists (ACLs).

You can also use the administration API to attach ACLs to Security Access Manager protected objects and to detach ACLs from protected objects.

Each ACL might contain entries for specific users and groups. You can use the administration API to set ACL entries for users and groups that exist in the Security Access Manager secure domain. You also can use the administration API to set ACL entries for the default user categories **any-other** and **unauthenticated**.

ACL entries consist of one or more permissions. These permissions specify actions that the owner of the entry is allowed to do. Security Access Manager provides a number of default permissions. You can use the administration API to define additional extended actions. You also can use the administration API to group the extended actions into action groups.

Understand the construction and use of ACLs before working with administration API ACL functions. The correct use of ACLs is key to successfully implementing a security policy. For more information, see the chapter about using access control lists in the *IBM Security Access Manager for Web: Administration Guide*.

This chapter contains the following topics:

Administering access control lists

Use ACLs to grant or restrict specific users and groups access to protected resources.

Use the administration API to:

- Create and delete ACLs
- Retrieve or change information associated with an ACL
- List the user, group, any-other, and unauthenticated entries that are included in the ACL
- List all defined ACLs

The name of an ACL can be of any length. The following characters are allowed in an ACL name:

- Alphanumeric characters defined in the locale
- The underscore (_) character
- The hyphen (-) character

You specify the user entries that belong in each ACL. You also specify the permissions or actions that each user is allowed to do.

You can specify permissions or actions based on group membership, rather than individual user identity, to expedite administration tasks.

The administration API defines the **ivadmin_acl** data type to contain a retrieved ACL. You can use administration API functions to extract information from the `ivadmin_aclPDAcl` object.

Be sure that you understand how to define an ACL policy before working with administration API ACL functions. For more information, see the section about ACL entry syntax in the *IBM Security Access Manager for Web: Administration Guide*.

Table 16 describes the methods for administering ACLs.

Table 16. Administering access control lists

Functions	Description
"ivadmin_acl_create()" on page 60	Creates an ACL.
"ivadmin_acl_delete()" on page 61	Deletes the specified ACL.
"ivadmin_acl_get()" on page 62	Returns the specified ACL.
"ivadmin_acl_getdescription()" on page 63	Returns the description of the specified ACL.
"ivadmin_acl_getid()" on page 65	Returns the name of the specified ACL.
"ivadmin_acl_list()" on page 67	Returns the names of all the defined ACLs.
"ivadmin_acl_listgroups()" on page 69	Returns a list of group names included in the specified ACL.
"ivadmin_acl_listusers()" on page 70	Returns a list of the user names included in the specified ACL.
"ivadmin_acl_setdescription()" on page 75	Sets or modifies the description for the specified ACL.

Administering access control list entries

You must create an ACL object before you can administer ACL entries for the object.

To create an ACL object, see "ivadmin_acl_create()" on page 60.

The administration API can be used to specify entries for each of the following ACL entry types:

- Users
- Groups
- User **any-other** (also known as **any-authenticated**)
- User **unauthenticated**

The type **any-other** applies to any user that is authenticated into the Security Access Manager secure domain but that does not have a separate entry in the ACL. The type **unauthenticated** applies to all user identities that are unknown to Security Access Manager. Unknown users cannot authenticate into the Security Access Manager secure domain.

Ensure that you understand ACL entry syntax, ACL entry types, ACL ID attributes, and ACL permission (action) attributes. Then use the administration API functions in this section.

Security Access Manager supports 18 default actions. For a list of the default Security Access Manager actions, see the section about default Security Access Manager permissions for actions in the *IBM Security Access Manager for Web: Administration Guide*.

For more information, see the section about ACL entry syntax in the *IBM Security Access Manager for Web: Administration Guide*.

Table 17 lists the methods for administering ACL entries.

Table 17. Administering access control list entries

Functions	Description
"ivadmin_acl_getanyother()" on page 63	Returns the actions defined in the entry for the user type **any-other** in the specified ACL.
"ivadmin_acl_getunauth()" on page 66	Returns the actions (permissions) defined in the entry for the user type `unauthenticated` in the specified ACL.
"ivadmin_acl_getuser()" on page 66	Returns the actions (permissions) defined in the entry for the specified user in the specified ACL.
"ivadmin_acl_setuser()" on page 78	Returns the actions (permissions) defined in the entry for the specified group in the specified ACL.
"ivadmin_acl_removeanyother()" on page 71	Removes the ACL entry for the **any-other** user from the specified ACL.
"ivadmin_acl_removegroup()" on page 72	Removes the ACL entry for the specified group from the specified ACL.
"ivadmin_acl_removeunauth()" on page 72	Removes the ACL entry for the **unauthenticated** user from the specified ACL.
"ivadmin_acl_removeuser()" on page 73	Removes the ACL entry for the specified user from the specified ACL.
"ivadmin_acl_setanyother()" on page 74	Sets or modifies the ACL entry for the **any-other** user in the ACL. Call this function to specify permissions for all authenticated users who do not have a separate user or group entry in the specified ACL.
"ivadmin_acl_setgroup()" on page 76	Sets or modifies the ACL entry for the specified group in the specified ACL.
"ivadmin_acl_setunauth()" on page 77	Sets the ACL entry for the **unauthenticated** user in the specified ACL. Call this function to specify permissions for those users who are not authenticated.
"ivadmin_acl_setuser()" on page 78	Sets the entry for the specified user in the specified ACL. Use this method to specify the actions that a user is permitted.

Administering access control list extended attributes

Extended attributes for an ACL can be obtained, set, and deleted.

Table 18 lists the methods available for administering ACL extended attributes.

Table 18. Administering access control list extended attributes

Functions	Description
"ivadmin_acl_attrdelkey()" on page 56	Deletes the specified extended attribute key from the specified ACL.
"ivadmin_acl_attrdelval()" on page 57	Deletes the specified value from the specified extended attribute key in the specified ACL.
"ivadmin_acl_attrget()" on page 58	Gets the extended attribute values for the specified extended attribute key from the specified ACL.
"ivadmin_acl_attrlist()" on page 59	Lists the extended attribute keys associated with the specified ACL.
"ivadmin_acl_attrput()" on page 60	Creates an extended attribute with the specified name and value, if it does not exist, and adds the attribute to the specified ACL. If the attribute specified exists, the specified value is added to the existing values for the attribute.

Administering action groups

You can use the administration API to create, examine, and delete new action groups.

Each action group can contain up to 32 actions. The default action group, referred to as the primary action group, contains the 18 predefined Security Access Manager actions, which means you can create up to 14 new actions to the primary group.

When you need to create more than 32 actions, you can use the administration API to define a new action group. Security Access Manager supports up to 32 action groups.

For more information about action groups, see the section about creating extended ACL actions and action groups in the *IBM Security Access Manager for Web: Administration Guide*. Table 19 lists the methods for administering action groups.

Table 19. Administering action groups

Functions	Description
"ivadmin_action_create_in_group()" on page 80	Defines a new action (permission) code in the specified action group. Call this function to add an action code to a user-defined extended action group.
"ivadmin_action_delete_from_group()" on page 82	Deletes an action (permission) code from the specified action group.

Table 19. Administering action groups (continued)

Functions	Description
"ivadmin_action_group_create()" on page 85	Creates a new action group with the specified name.
"ivadmin_action_group_delete()" on page 85	Deletes the specified action group and all the actions that belong to the specified group.
"ivadmin_action_group_list()" on page 86	Lists all the defined action group names.
"ivadmin_action_list_in_group()" on page 88	Lists all the defined action (permission) codes from the specified action group.

Administering extended actions

Security Access Manager provides a default set of actions (permissions) that belong to the primary action group that can be granted to users or groups. You can use the administration API to define new extended actions that supplement the set of default actions.

Each of the extended actions can belong to the primary action group or to a custom action group.

Extended actions are typically defined to support actions that are specific to a third-party application. For more information about extended actions, see the section about creating extended ACL actions and action groups in the *IBM Security Access Manager for Web: Administration Guide*.

Table 20 lists the methods for administering extended actions.

Table 20. Administering extended actions

Functions	Description
"ivadmin_action_create()" on page 79	Defines a new action (permission) code in the specified action group.
"ivadmin_action_delete()" on page 81	Deletes an action (permission) code from the specified action group.
"ivadmin_action_getdescription()" on page 83	Returns the description for the specified action.
"ivadmin_action_getid()" on page 84	Returns the code for the specified action.
"ivadmin_action_gettype()" on page 84	Returns the type for the specified action.
"ivadmin_action_list()" on page 87	Lists all the defined action (permission) codes for the specified action group.

Chapter 6. Administering protected object policies

You can use the administration API to create, modify, examine, and delete Security Access Manager protected object policies (POPs).

You can also use the Administration API to attach or detach POPs from protected objects.

You can use POPs to impose additional conditions on operations that are permitted by an access control list (ACL) policy. These additional conditions are enforced regardless of the user or group identities specified in the ACL entries.

Examples of additional conditions include:

- Specifying the quality of protection
- Writing a report record to the auditing service
- Requiring an authentication strength level
- Restricting access to a specific time period
- Enabling or disabling warning mode, which allows an administrator to validate security policy

Be sure that you understand Security Access Manager POPs before working with administration API to administer POPs. For more information, see the chapter about using POPs in the *IBM Security Access Manager for Web: Administration Guide*.

This chapter contains the following topics:

Administering protected object policy objects

POP objects are administered in a similar way to ACL policies. You can create and configure a POP, and then attach the POP to objects in the protected object space.

The administration API defines the `ivadmin_pop` data type to contain the retrieved POP. You can use administration API functions to extract data from the `ivadmin_pop` objects. You do not need to know the internal structure of the `ivadmin_pop` data type.

Table 21 lists the methods for administering protected object policy objects.

Table 21. Administering protected object policy objects

Function	Description
"ivadmin_pop_create()" on page 175	Creates a POP object with the default values.
"ivadmin_pop_delete()" on page 177	Deletes the specified POP.
"ivadmin_pop_detach()" on page 177	Detaches a POP from the specified protected object.
"ivadmin_pop_find()" on page 178	Finds and lists all protected objects that have the specified POP attached.

Table 21. Administering protected object policy objects (continued)

Function	Description
"ivadmin_pop_get()" on page 179	Returns the specified POP object. Call this function to get an object of type `ivadmin_pop`.
"ivadmin_pop_list()" on page 186	Lists all POP objects.

Administering protected object policy settings

You can use the administration API to set, modify, or remove attributes in a POP. You must create the POP object before specifying POP settings.

To create a POP object, see "ivadmin_pop_create()" on page 175. You can use administration API functions to specify the following POP attributes:

- Authentication levels
- Quality of Protection (QOP) requirements
- Auditing levels
- Time of day access restrictions
- Warning mode settings

Authentication levels specify whether additional or alternative authentication is required to access a protected object. The additional authentication is also called step-up authentication. This means that an additional authentication step is required to access resources that require more restrictive access policies. When using step-up authentication, you can either filter users based on IP address or you can specify step-up authentication for all users, regardless of IP address.

Call **ivadmin_pop_setanyothernw()** or `ivadmin_pop_setipauth()` to specify step-up authentication policy for objects that require authentication-sensitive authorization. When using step-up authentication, you can either filter users based on IP address or you can specify step-up authentication for all users, regardless of IP address.

Call `ivadmin_pop_setanyothernw()` or `ivadmin_pop_setipauth()` when you want to specify a POP that specifies step-up authentication policy for all users, regardless of IP address.

For more information about the use of the authentication level by WebSEAL, see the section about authentication strength POP policy (step-up) in the *IBM Security Access Manager for Web: Web Security Developer Reference*.

The quality of protection (QOP) level is not enforced internally by Security Access Manager. Applications that set the quality of protection can enforce it.

Audit levels specify what operations generate an audit record. This value is used internally by Security Access Manager and also can be used by applications to generate their audit records.

The time of day access setting is used to control access to a protected object based on the time when the access occurs.

The warning mode enables a security administrator to troubleshoot the authorization policy set on the protected object space.

When you set the warning attribute to **yes**, any action is possible by any user on the object where the POP is attached. Any access to an object is permitted even if the ACL policy attached to the object is set to deny this access.

Audit records are generated that capture the results of all ACL policies with warning mode set throughout the object space. The audit log shows the outcome of an authorization decision as it is made if the warning attribute is set to `no`.

Table 22 lists the methods for administering protected object policy settings.

Table 22. Administering protected object policy settings

Functions	Description
"ivadmin_pop_getanyothernw2()" on page 180	Returns the `anyothernw` or any other network, setting for the IP authentication level from the specified POP.
"ivadmin_pop_getauditlevel()" on page 180	Returns the audit level for the specified POP.
"ivadmin_pop_getdescription()" on page 181	Returns the description of the specified POP.
"ivadmin_pop_getipauth3()" on page 182	Returns the IP endpoint authentication setting in the specified POP.
"ivadmin_pop_getid()" on page 182	Returns the name of the specified POP.
"ivadmin_pop_getqop()" on page 184	Returns the quality of protection (QOP) level for the specified POP.
"ivadmin_pop_gettod()" on page 184	Returns the time of day range for the specified POP.
"ivadmin_pop_getwarnmode()" on page 185	Returns the warning mode value from the specified POP.
"ivadmin_pop_removeipauth2()" on page 187	Removes the `ipauth` access setting for authentication level from the specified POP.
"ivadmin_pop_setanyothernw2()" on page 188	Sets the `anyothernw` setting for authentication level from the specified POP.
"ivadmin_pop_setanyothernw_forbidden2()" on page 189	Sets the `anyothernw` access setting to `forbidden` for the specified POP.
"ivadmin_pop_setauditlevel()" on page 189	Sets the audit level for the specified POP.
"ivadmin_pop_setdescription()" on page 190	Sets the description of the specified POP.
"ivadmin_pop_getipauth3()" on page 182	Returns the specified IP endpoint authentication settings in the specified protected object policy (POP).
"ivadmin_pop_setipauth2()" on page 191	Sets the `ipauth` setting for authentication level in the specified POP.
"ivadmin_pop_setipauth_forbidden2()" on page 192	Sets the `ipauth` setting for authentication level to `forbidden` in the specified POP.
"ivadmin_pop_setqop()" on page 193	Sets the quality of protection level for the specified POP.
"ivadmin_pop_settod()" on page 194	Sets the time of day range for the specified POP.
"ivadmin_pop_setwarnmode()" on page 195	Sets the warning mode for the specified POP.

Administering protected object policy extended attributes

You can use the administration API to set, modify, or remove extended attributes in a POP.

Table 23 lists the methods for administering protected object policy extended attributes.

Table 23. Administering protected object policy extended attributes

Functions	Description
"ivadmin_pop_attrdelkey()" on page 171	Deletes the specified extended attribute from the specified POP.
"ivadmin_pop_attrdelval()" on page 172	Deletes the specified value from the specified extended attribute key in the specified POP.
"ivadmin_pop_attrget()" on page 173	Gets the values for the specified extended attribute from the specified POP.
"ivadmin_pop_attrlist()" on page 174	Lists the extended attributes associated with the specified POP.
"ivadmin_pop_attrput()" on page 175	Sets the value for the specified extended attribute in the specified POP.

Chapter 7. Administering authorization rules

Authorization rules are conditions or standards in an authorization policy. Authorization rules make access decisions based on attributes such as user, application, and environment context.

Authorization rules are defined to specify conditions that must be met before access to a protected object is permitted. A rule is created with a number of Boolean conditions. The conditions are based on data supplied to the authorization engine in the user credential, from the resource manager application, or from the encompassing business environment.

A Security Access Manager authorization rule is a policy type similar to an access control list (ACL) or a protected object policy (POP). The rule is stored as a text rule in a rule policy object. The rule is attached to a protected object in the same way and with the same constraints as ACLs and POPs.

The Security Access Manager administration API provides functions to create, delete, modify, list, and get authorization rules.

See the *IBM Security Access Manager for Web: Administration Guide*.

Use the functions shown in Table 24 to administer authorization rule objects.

Table 24. Administering authorization rules

Function	Description
"ivadmin_authzrule_create()" on page 89	Creates the specified authorization rule object.
"ivadmin_authzrule_delete()" on page 90	Deletes the specified authorization rule object.
"ivadmin_authzrule_get()" on page 90	Returns the specified authorization rule object.
"ivadmin_authzrule_getid()" on page 92	Returns the ID for the specified authorization rule.
"ivadmin_authzrule_getdescription()" on page 91	Returns the description for the specified authorization rule.
"ivadmin_authzrule_getfailreason()" on page 92	Returns the fail reason, if any, for the specified authorization rule.
"ivadmin_authzrule_getruletext()" on page 93	Returns the rule text for the specified authorization rule.
"ivadmin_authzrule_list()" on page 93	Lists all the registered authorization rules.
"ivadmin_authzrule_setdescription()" on page 94	Sets the description for the specified authorization rule.
"ivadmin_authzrule_setruletext()" on page 96	Sets the authorization rule text.
"ivadmin_authzrule_setfailreason()" on page 95	Sets the authorization rule fail reason.

Chapter 8. Administering single sign-on resources

You can use the administration API to administer resources that enable a Security Access Manager user to obtain single sign-on (SSO) capability across more than one web server.

This capability requires the use of Security Access Manager WebSEAL junctions.

You can use the administration API to create, modify, examine, and delete the following types of resources:

- Administering web resources
- Administering resource groups
- Administering resource credentials

Be sure that you understand Security Access Manager single sign-on support before you use the administration API to administer single sign-on resources. For more information about administering single sign-on capability across junctioned web server resources, see the section about user registry resource management commands in the *IBM Security Access Manager for Web: Administration Guide* and the section about using global sign-on (GSO) in the *IBM Security Access Manager for Web: Web Security Developer Reference*.

This chapter contains the following topics:

Administering web resources

A *web resource* is a web server that serves as the backend of a Security Access Manager WebSEAL junction.

An application on the joined web server can require users to authenticate specifically to the application.

The authentication information, such as user name and password, often differs from the authentication information used by Security Access Manager. Because of this difference, the junctioned web server requires an authenticated Security Access Manager user to log in again, with user name and password specific to the application on the joined web server.

You can use the administration API to configure Security Access Manager so that Security Access Manager users need to authenticate only one time. You must define a web resource (server) and then define a user-specific resource credential that contains user-specific authentication information for the web resource.

This section describes how to create, modify, and delete web resources. Administration of resource credentials is described in "Administering resource credentials" on page 45.

Note: The administration API does not do all WebSEAL junction configuration tasks through the API. Use the **pdadmin** commands to modify the junction definitions. For more information, see the *IBM Security Access Manager for Web: WebSEAL Administration Guide*.

Table 25 lists the methods for administering web resources.

Table 25. Administering web resources

Functions	Description
"ivadmin_ssoweb_create()" on page 245	Creates a single sign-on web resource.
"ivadmin_ssoweb_delete()" on page 246	Deletes the specified single sign-on web resource.
"ivadmin_ssoweb_get()" on page 247	Returns the specified single sign-on web resource.
"ivadmin_ssoweb_getdescription()" on page 247	Returns the description of the specified single sign-on web resource.
"ivadmin_ssoweb_getid()" on page 248	Returns the name (identifier) of the specified single sign-on web resource.
"ivadmin_ssoweb_list()" on page 249	Returns a list of all the single sign-on web resource names.

Administering resource groups

A *resource group* is a group of web servers. The servers are all junctioned to a Security Access Manager WebSEAL server and all use the same set of user IDs and passwords.

You can use the administration API to create resource groups. You can then create a single resource credential for all the resources in the resource group. Use a single resource credential to simplify the management of web resources by grouping similar web resources into resource groups.

You can also use the administration API to add more web resources, when necessary, to an existing resource group.

Table 26 lists the methods for administering resource groups.

Table 26. Administering resource groups

Functions	Description
"ivadmin_ssogroup_addres()" on page 238	Adds a single sign-on resource to a single sign-on resource group.
"ivadmin_ssogroup_create()" on page 239	Creates a single sign-on group resource.
"ivadmin_ssogroup_delete()" on page 240	Deletes a single sign-on group resource.
"ivadmin_ssogroup_get()" on page 241	Returns the specified single sign-on group resource.
"ivadmin_ssogroup_getdescription()" on page 241	Returns the description of the single sign-on group resource.
"ivadmin_ssogroup_getid()" on page 242	Returns the name of the single sign-on group resource.

Table 26. Administering resource groups (continued)

Functions	Description
"ivadmin_ssogroup_getresources()" on page 243	Returns a list of the member single sign-on resource names for the specified single sign-on group.
"ivadmin_ssogroup_list()" on page 243	Returns a list of all the single sign-on group resource names.
"ivadmin_ssogroup_removeres()" on page 244	Removes a single sign-on resource from the specified single sign-on resource group.

Note: Depending on the LDAP server in your environment, any attempt to remove a non-existing resource from a group might generate an error.

Administering resource credentials

A *resource credential* provides a user ID and password for a single sign-on user-specific resource, such as a web server or a group of web servers.

The web resource or group of web resources must exist before you can apply resource credentials to it. Resource credential information is stored in the Security Access Manager entry in the user registry.

You can use the administration API to create, modify, examine, and delete resource credentials. Table 27 lists the methods for administering credentials.

Table 27. Administering credentials

Functions	Description
"ivadmin_ssocred_create()" on page 230	Creates a single sign-on credential.
"ivadmin_ssocred_delete()" on page 231	Deletes a single sign-on credential.
"ivadmin_ssocred_get()" on page 232	Returns the specified single sign-on credential.
"ivadmin_ssocred_getid()" on page 233	Returns the name of the single sign-on resource associated with this credential.
"ivadmin_ssocred_getssopassword()" on page 234	Returns the password associated with the single sign-on credential.
"ivadmin_ssocred_getssouser()" on page 234	Returns the name of the resource user associated with the specified single sign-on credential.
"ivadmin_ssocred_gettype()" on page 235	Returns the type of the single sign-on resource associated with the specified single sign-on credential.
"ivadmin_ssocred_getuser()" on page 236	Returns the name of the Security Access Manager user associated with the single sign-on credential.
"ivadmin_ssocred_list()" on page 236	Returns the list of single sign-on credentials for the specified user.
"ivadmin_ssocred_set()" on page 237	Modifies a single sign-on credential.

Chapter 9. Administering domains

A Security Access Manager domain consists of all the physical resources that require protection. The domain also has the associated security policy that protects those resources.

The initial domain is the management domain and is created when the Policy Server is configured. Multiple domains can exist simultaneously in a Security Access Manager environment. Data is securely partitioned between domains. A user or process must authenticate to a specific domain to access data contained in it.

Each Security Access Manager environment contains a single management domain. A user must be authenticated to the management domain to create, delete, list, or modify additional domains.

The authorization API provides functions that can be used to manage domains.

For more information about the management of domains, see the *IBM Security Access Manager for Web: Administration Guide*. Table 28 lists the methods for administering domains.

Table 28. Administering domains

Functions	Description
"ivadmin_domain_create()" on page 148	Creates a new Security Access Manager domain.
"ivadmin_domain_delete()" on page 149	Deletes the specified Security Access Manager domain.
"ivadmin_domain_get()" on page 150	Gets the specified Security Access Manager domain object.
"ivadmin_domain_getdescription()" on page 151	Gets the description for the specified Security Access Manager domain.
"ivadmin_domain_getid()" on page 152	Gets the name of the specified Security Access Manager domain.
"ivadmin_domain_list()" on page 152	Lists the names of all the Security Access Manager domains, except for the management domain.
"ivadmin_domain_setdescription()" on page 153	Changes the description for the specified Security Access Manager domain.

Chapter 10. Configuring application servers

You can use the administration API to configure and unconfigure authorization and administration servers, modify configuration parameters, administer replicas, and do certificate maintenance.

These APIs are used by the `svrsslcfg` command-line utility instead of the **pdadmin** command-line utility.

The `svrsslcfg` command-line utility is used to do the necessary configuration steps that allow an application to use a secure sockets layer (SSL) connection for communicating with the policy server or the authorization server. It is not intended to do all the configuration that might be required to ensure a correctly functioning application.

For more information about the `svrsslcfg` utility, see the section about using `svrsslcfg` in the *IBM Security Access Manager for Web: Command Reference*.

Note: The local host name is used to build a unique name for the application. Depending on the TCP/IP configuration, the host name is not always consistent and might result in lookup failures. For example, the operating system might return the fully qualified host name while another computer might return the host name. If this problem happens in your network, use the following format to specify the server name to the command-line interface:

server_name/desired_host_name

For the API, these parameters are separate. The value of *desired_host_name* must be specified for the *host_name* parameter.

This chapter contains the following topics:

Configuration commands

Use configuration commands to enable an application and application server that use the authorization or administration API to communicate with the policy server or the authorization server.

An administrative user identity (for example, `sec_master`) and password must be specified for connecting to the policy server.

Table 29. Configuring application servers

Functions	Description
"ivadmin_cfg_configureserver3()" on page 99	Configures an application server by updating the configuration file and creating the keyring file.
"ivadmin_cfg_setlistening2()" on page 110	Sets or resets the enable-listening parameter in the configuration file.

Table 29. Configuring application servers (continued)

Functions	Description
"ivadmin_cfg_setport2()" on page 111	Changes the listening port number of the application and updates the port number in the configuration file.
"ivadmin_cfg_unconfigureserver()" on page 116	Unconfigures an application server.

Administering replicas

Use the configuration commands to add, change, or delete replica entries in the configuration file and return other configuration information.

Table 30. Administering replicas

Functions	Description
"ivadmin_cfg_addreplica2()" on page 97	Adds a replica entry to the configuration file.
"ivadmin_cfg_chgreplica2()" on page 98	Changes parameters of a replica entry in the configuration file.
"ivadmin_cfg_rmvreplica2()" on page 108	Removes a replica entry from the configuration file.

Certificate maintenance

Use the `ivadmin_cfg_renewservercert()` function only when the certificate is compromised.

Table 31. Certificate maintenance

Functions	Description
"ivadmin_cfg_renewservercert()" on page 107	Renews the server SSL certificate.

Chapter 11. Administering servers

You can use the administration API to get a list of tasks from the server.

You can also send a specific task to an authorization server and notify replica databases, either automatically or manually, when the master authorization database is updated.

This chapter contains the following topics:

Getting and doing administration tasks

You can send an administration task to a server. You also can request a list of all supported administration tasks from a server.

The caller must have credentials with sufficient permission to do the task. For more information, see the *IBM Security Access Manager for Web: Authorization C API Developer Reference*.

Notifying replica databases when the master authorization database is updated

When an administrator makes security policy changes, the policy server adjusts the master authorization database to reflect these changes.

To ensure that these changes also are dispersed to any authorization servers with replica databases, you can either:

- Configure a Security Access Manager application server, such as WebSEAL, to poll the master authorization database at regular intervals for updates. By default, polling is disabled. For more information about polling the master authorization database, see the `cache-refresh-interval` option described in the *IBM Security Access Manager for Web: Authorization C API Developer Reference*.
- Enable the policy server to notify authorization servers each time that the master authorization database is updated. Use this automatic process for environments where database changes are infrequent. For more information, see "Notifying replica databases automatically" on page 52.
- Notify authorization servers on demand, after you make updates to the master authorization database. Use this manual process for environments where database changes are frequent and involve substantial changes. For instructions, see "Notifying replica databases manually" on page 52.

Select the method that you want to use to update replica databases (automatic, manual, or both). Then fine-tune the settings in the `ivmgrd.conf` file on the policy

server. See “Setting the maximum number of notification threads” and “Setting the notification wait time.”

Notifying replica databases automatically

You can enable the policy server to send notifications to authorization servers each time that the master authorization database is updated.

In turn, the authorization servers automatically request a database update from the policy server.

To enable automatic database updates, edit the `ivmgrd.conf` file on the policy server and add the following *attribute=value* stanza entry pair:

```
[ivmgrd]
auto-database-update-notify = yes
```

You must restart the policy server for changes to take effect. Use this setting for environments where the master database is changed infrequently. To turn off automatic notification, specify no.

Notifying replica databases manually

When the master authorization database is updated, you can use the `ivadmin_server_replicate()` function to send notifications to application servers that are configured to receive database update notifications.

You can specify that a specific server receives update notifications, or specify NULL, which notifies all configured authorization servers in the secure domain.

If you specify a server name, you are notified whether the server was replicated successfully or if a failure occurred. If you do not specify a server name, return codes indicate whether the policy server started notifying authorization servers in your secure domain. Unless you specify the `server-name` option, you are not notified when an authorization server database was replicated successfully.

Setting the maximum number of notification threads

When the master authorization database is updated, this update is announced to replica databases through notification threads. Each replica then has the responsibility of downloading the new data from the master authorization database.

You can edit the `ivmgrd.conf` file to set a value for the maximum number of notification threads. This number is calculated based on the number of replica databases in your secure domain. For example, if you have 10 replica databases and want to notify them of master database changes simultaneously, specify a value of 10 for the max-notifier-threads stanza entry as shown:

```
[ivmgrd]
max-notifier-threads = 10
```

The default value is 10 threads.

Setting the notification wait time

There is a time delay in seconds between when the policy server updates the master authorization database and when notification is sent to database replicas.

If you added the `auto-database-update-notify = yes` stanza entry to the `ivmgrd.conf` file as described in "Notifying replica databases automatically" on page 52, you can set this period.

To do so, edit the `notifier-wait-time` stanza entry value in the `ivmgrd.conf` file. For example, you might make batch changes to the master authorization database. Wait until all changes are made before sending policy changes to database replicas. You might decide to increase the default value from 15 seconds to 25 seconds as shown:

```
[ivmgrd]
notifier-wait-time = 25
```

Editing the value for this attribute prevents the policy server from sending individual replica notifications for each of a series of database changes.

Administrating servers and database notification

The software provides several functions and methods to administer servers and database notification.

Table 32. Administering servers and database notification

Functions	Description
"ivadmin_server_gettasklist()" on page 227	Returns a list of tasks from the server.
"ivadmin_server_performtask()" on page 228	Sends a command to an authorization server.
"ivadmin_server_replicate()" on page 229	Notifies authorization servers to receive database updates.

Chapter 12. Administration C API reference

The APIs in this chapter are in alphabetical order by name.

ivadmin_accessOutdata_getAccessResult()

This API interprets the result from the `ivadmin_protobj_access()` and `ivadmin_protobj_multiaccess()` functions. The API returns the access result, which indicates whether a specified user can access the specified object.

Syntax

```
unsigned long ivadmin_accessOutdata_getAccessResult(
  ivadmin_accessOutdata outdata
);
```

Parameters

Input

outdata
A pointer to an `ivadmin_Outdata` structure previously returned from either the `ivadmin_protobj_access()` or `ivadmin_protobj_multiaccess()` function.

Description

Indicates whether the user has the specified access to the specified object.

Free this structure when it is no longer needed.

Return values

Returns the following values:

AZN_C_PERMITTED
Indicator of whether the user has the necessary access.

AZN_C_NOT_PERMITTED
Indicator of whether the user does not have the necessary access.

ivadmin_accessOutdata_getPermInfo()

This API interprets the result from the `ivadmin_protobj_access()` and `ivadmin_protobj_multiaccess()` functions. The API returns the permission information, if any, that is associated with an access request to an object.

Syntax

```
azn_attrlist_h_t ivadmin_accessOutdata_getPermInfo(
ivadmin_accessOutdata outdata
);
```

Parameters

Input

outdata

A pointer to an `ivadmin_Outdata` structure previously returned from either the `ivadmin_protobj_access()` or `ivadmin_protobj_multiaccess()` function.

Description

Returns the supplemental permission information (`azn_attrlist_h_t` structure) that is associated with the specified `ivadmin_Outdata` object.

Free this structure when it is no longer needed.

Return values

Returns the supplemental permission information (`azn_attrlist_h_t` structure) that is associated with the specified `ivadmin_Outdata` object.

ivadmin_accessOutdata_getResponseInfo()

This API returns the response information that is associated with an access request for an object.

Syntax

```
ivadmin_response ivadmin_accessOutdata_getResponseInfo(
ivadmin_accessOutdata outdata
);
```

Parameters

Input

outdata

A pointer to an `ivadmin_Outdata` structure previously returned from either the `ivadmin_protobj_access()` or `ivadmin_protobj_multiaccess()` function.

Description

Returns the response information that is associated with the specific access request to a specific object.

Free this structure when it is no longer needed.

Return values

Returns the following values:

IVADMIN_TRUE

Defined as 1. The function was successful.

IVADMIN_FALSE

Defined as 0. The function encountered an error.

ivadmin_acl_attrdelkey()

This API deletes the specified extended attribute key from the specified access control list.

Deletes the specified extended attribute key from the specified access control list.

Syntax

```
unsigned long ivadmin_acl_attrdelkey(
ivadmin_context ctx,
char *aclid,
char *attr_key,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The name of the access control list.

attr_key
The extended attribute to delete.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified extended attribute key from the specified access control list.

Command-line equivalent:

```
pdadmin acl modify ACL_name delete attribute attribute_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_attrdelval()

This API deletes the specified value from the specified extended attribute key in the specified access control list.

Syntax

```
unsigned long ivadmin_acl_attrdelval(
ivadmin_context ctx,
char *aclid,
char *attr_key,
char *attr_value,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The name of the access control list.

attr_key
The extended attribute key.

attr_value
The extended attribute value to delete from the extended attribute key.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified value from the specified extended attribute key in the specified access control list.

Command-line equivalent:

```
pdadmin acl modify ACL_name delete attribute attribute_name attribute_value
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_attrget()

This API returns the extended attribute value for the specified extended attribute key from the specified access control list.

Syntax

```
unsigned long ivadmin_acl_attrget(
ivadmin_acl acl,
char *attr_key,
unsigned long *count,
char ***attr_value
);
```

Parameters

Input

acl The `ivadmin_acl` object. This object contains the access control list.

attr_key
The attribute key to look up.

Output

count The number of extended attribute values returned.

attr_value
An array of pointers to the values returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

Description

Returns the extended attribute values for the specified extended attribute key from the specified access control list.

Command-line equivalent:

```
pdadmin acl show ACL_name attribute attribute_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_attrlist()

This API lists the extended attribute keys that are associated with the specified access control list.

Syntax

```
unsigned long ivadmin_acl_attrlist(
ivadmin_acl acl,
unsigned long *count,
char ***attr_list
);
```

Parameters

Input

acl The `ivadmin_acl` object. This object contains the access control list.

Output

count The number of extended attributes returned.

attr_list
An array of pointers to the extended attributes returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

Description

Lists the extended attribute keys that are associated with the specified access control list.

Command-line equivalent:

```
pdadmin acl list ACL_name attribute
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_attrput()

This API creates or changes the extended attribute value for the specified extended attribute key in the specified access control list.

Syntax

```
unsigned long ivadmin_acl_attrput(
ivadmin_context ctx,
char *aclid,
char *attr_key,
char *attr_value,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The name of the access control list.

attr_key
The extended attribute key for which you want to set a value.

attr_value
The value to set.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the extended attribute value for the specified extended attribute key in the specified access control list.

Command-line equivalent:

```
pdadmin acl modify ACL_name set attribute attribute_name attribute_value
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_create()

This API creates an access control list (ACL).

Syntax

```
unsigned long ivadmin_acl_create(
ivadmin_context ctx,
const char *aclid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The name of the ACL to create. The name can be of any length. The following characters are valid in the ACL name.
- Alphanumeric characters defined in the locale
- The underscore (_) character
- The hyphen (-) character

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates an ACL. This function creates an ACL policy in the policy database. It does not create the specific ACL entries.

Command-line equivalent:

```
pdadmin acl create ACL_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_delete()

This API deletes the specified access control list.

Syntax

```
unsigned long ivadmin_acl_delete(
ivadmin_context ctx,
const char *aclid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The name of the access control list.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified access control list.

Command-line equivalent:

```
pdadmin acl delete ACL_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_get()

This API returns the specified access control list.

Syntax

```
unsigned long ivadmin_acl_get(
ivadmin_context ctx,
const char *aclid,
ivadmin_acl *acl,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The name of the access control list.

Output

acl The returned access control list. Free this memory when it is no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the specified access control list.

Command-line equivalent:

```
pdadmin acl show ACL_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_getanyother()

This API returns the actions (permissions) that are defined in the entry for the user **any-other** in the specified access control list.

Syntax

```
const char* ivadmin_acl_getanyother(
ivadmin_acl acl
);
```

Parameters

Input

acl A pointer to the access control list.

Description

Returns the actions that are defined in the entry for the user **any-other** in the specified access control list. You must call the **ivadmin_acl_get()** function to obtain the `ivadmin_acl` object before using this function to obtain the actions defined for the **any-other** user type. Free this character string when it is no longer needed.

Each action is represented by a single alphabetic character. Default actions are provided in the primary action group by Security Access Manager. These default actions, such as `A` for add, or `v` for view, are listed in the *IBM Security Access Manager for Web: Administration Guide*.

Actions in the primary action group are always returned first, followed by the actions defined in other action groups. For example, if the entry contains the add and view actions from the primary action group, along with the P, D, and q actions from the AdminGroup action group, and the b and V actions from the Auditors action group, the returned string might be:

```
Av[AdminGroup]PDq[Auditors]bV
```

If no actions are defined in the entry, an empty string (`""`) is returned.

Command-line equivalent:

```
pdadmin acl show any-other
```

Return values

Returns the actions defined in the entry for the user **any-other** in the specified access control list.

ivadmin_acl_getdescription()

This API returns the description of the specified access control list.

Syntax

```
const char* ivadmin_acl_getdescription(
ivadmin_acl acl
);
```

Parameters

Input

acl A pointer to the access control list.

Description

Returns the description of the specified access control list. You must call the `ivadmin_acl_get()` function to obtain the `ivadmin_acl` object before using `ivadmin_acl_getdescription()`. Do not free this entry. This data is maintained in the access control list structure.

Command-line equivalent:

`pdadmin acl show ACL_name`

The description is part of the information returned by the `pdadmin acl show` command.

Return values

Returns the description of the specified access control list. The maximum length for a description is 1024 characters.

ivadmin_acl_getgroup()

This API returns the actions (permissions) defined in the entry for the specified group in the specified access control list.

Syntax

```
const char* ivadmin_acl_getgroup(
ivadmin_acl acl,
const char *groupid
);
```

Parameters

Input

acl A pointer to the access control list.

groupid
The name of the group for which you want the actions.

Description

Returns the actions (permissions) defined in the entry for the specified group in the specified access control list. You must call the `ivadmin_acl_get()` function to obtain the `ivadmin_acl` object before using the `ivadmin_acl_getgroup()` function to obtain the actions defined for the group. Free this entry when it is no longer needed.

Each action is represented by a single alphabetic character. Default actions are provided in the primary action group by Security Access Manager. These default actions, such as `A` for add, or `v` for view, are listed in the *IBM Security Access Manager for Web: Administration Guide*.

Actions in the primary action group are always returned first, followed by the actions defined in other action groups. For example, if the entry contains the add and view actions from the primary action group, along with the P, D, and q actions from the AdminGroup action group, and the b and V actions from the Auditors action group, the returned string might be:

```
Av[AdminGroup]PDq[Auditors]bV
```

If no actions are defined in the entry, an empty string ("") is returned.

Command-line equivalent:

```
pdadmin acl show ACL_name
```

Return values

Returns the actions (permissions) defined in the entry for the specified group in the specified access control list.

ivadmin_acl_getid()

This API returns the name of the specified access control list.

Syntax

```
const char* ivadmin_acl_getid(
ivadmin_acl acl
);
```

Parameters

Input

acl A pointer to the access control list.

Description

Returns the name of the specified access control list. You must call the `ivadmin_acl_get()` function to obtain the `ivadmin_acl` object before using this function. Do not free the returned name. This data is maintained in the `ivadmin_acl` structure.

Command-line equivalent:

```
pdadmin acl show ACL_name
```

The access control list name is part of the information returned by the **pdadmin** command.

Return values

Returns the name of the specified access control list. There is no limit to the length of the name.

ivadmin_acl_getunauth()

This API returns the actions (permissions) that are defined in the entry for the unauthenticated user in the specified access control list.

Syntax

```
const char* ivadmin_acl_getunauth(
ivadmin_acl acl
);
```

Parameters

Input

acl A pointer to the access control list.

Description

Returns the actions (permissions) that are defined in the entry for the user unauthenticated in the specified access control list. You must call the `ivadmin_acl_get()` function to obtain the `ivadmin_acl` object before using the `ivadmin_get_unauth()` function to obtain the actions defined for all unauthenticated users. Free the returned actions when they are no longer needed.

Each action is represented by a single alphabetic character. Default actions are provided in the primary action group by Security Access Manager. These default actions, such as `A` for add, or `v` for view, are listed in the *IBM Security Access Manager for Web: Administration Guide*.

Actions in the primary action group are always returned first, followed by the actions defined in other action groups. For example, if the entry contains the add and view actions from the primary action group, along with the P, D, and q actions from the AdminGroup action group, and the b and V actions from the Auditors action group, the returned string might be:

```
Av[AdminGroup]PDq[Auditors]bV
```

If no actions are defined in the entry, an empty string ("") is returned.

Command-line equivalent:

```
pdadmin acl show ACL_name
```

Return values

Returns the actions (permissions) that are defined in the entry for the user unauthenticated in the specified access control list.

ivadmin_acl_getuser()

This API returns the actions (permissions) that are defined in the entry for the specified user in the specified access control list.

Syntax

```
const char* ivadmin_acl_getuser(
ivadmin_acl acl,
const char *userid
);
```

Parameters

Input

acl A pointer to the access control list.

userid
The name of the user entry from which you want to get the list of defined actions.

Description

Returns the actions (permissions) defined in the entry for the specified user in the specified access control list. You must call the `ivadmin_acl_get()` function to obtain the `ivadmin_acl` object before using `ivadmin_acl_getuser()` to obtain the actions defined for the user. Free this character string when no longer needed.

Each action is represented by a single alphabetic character. Default actions are provided in the primary action group by Security Access Manager. These default actions, such as `A` for add, or `v` for view, are listed in the *IBM Security Access Manager for Web: Administration Guide*.

Actions in the primary action group are always returned first, followed by the actions defined in other action groups. For example, if the entry contains the add and view actions from the primary action group, along with the P, D, and q actions from the AdminGroup action group, and the b and V actions from the Auditors action group, the returned string might be:

```
Av[AdminGroup]PDq[Auditors]bV
```

If no actions are defined in the entry, an empty string (`""`) is returned.

Command-line equivalent:

```
pdadmin acl show ACL_name
```

Return values

Returns the actions (permissions) that are defined in the entry for the specified user in the specified access control list.

ivadmin_acl_list()

This API returns the names of all the defined access control lists.

Syntax

```
unsigned long ivadmin_acl_list(
ivadmin_context ctx,
unsigned long *count,
char ***aclids,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

count The number of access control list names returned.

aclids An array of pointers to the access control list names returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the names of all the defined access control lists. If no access control lists exist, or an error is encountered, NULL is returned.

Command-line equivalent:

```
pdadmin acl list
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_list2()

This API returns the names of all the defined access control lists that match the input pattern and maximum return count.

Syntax

```
unsigned long ivadmin_acl_list2(
ivadmin_context ctx,
const char *pattern,
unsigned long *maxreturn,
unsigned long *count,
char ***aclids,
ivadmin_response *rsp
);
```

Default value

None

Option descriptions

Input

ctx The context represents the login session in the Security Access Manager administrator API. The context value sends administration requests to the policy server.

pattern
The search term for access control lists. `IVADMIN_ALLPATTERN` is a search term that indicates all access control lists.

maxreturn
The maximum number of access control lists to return. 0 means there are no limit to the returned number of access control lists.

Usage notes

Use this API to view the names of all the defined access control lists that match the input pattern and maximum return count.

The command-line equivalent is:

```
pdadmin acl list <pattern> <max-return>
```

Return values

count The number of access control list names returned.

aclids An array of pointers to the access control list names returned. You must free the character data referenced by each pointer and the array of pointers when they are no longer needed.

rsp The response object indicates the success or failure of the function. It also contains error information.

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

Examples

This sample returns the names of all the defined access control lists.

```
unsigned long ivadmin_acl_list2(
ivadmin_context ctx,
const char *pattern,
unsigned long *maxreturn,
```

ivadmin_acl_listgroups()

This API returns the group names that are included in the specified access control list.

Syntax

```
unsigned long ivadmin_acl_listgroups(
ivadmin_acl acl,
unsigned long *count,
char ***groupids
);
```

Parameters

Input

acl A pointer to the access control list.

Output

count The number of group names returned.

groupids
An array of pointers to the group names is returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

Description

Returns the group names that are included in the specified access control list. You must call the `ivadmin_acl_get()` function to obtain the `ivadmin_acl` object before using this function.

Command-line equivalent:

```
pdadmin acl show ACL_name
```

The list of group names is part of the information returned by this **pdadmin** command.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_listusers()

This API returns the user names that are included in the specified access control list.

Syntax

```
unsigned long ivadmin_acl_listusers(
ivadmin_acl acl,
unsigned long *count,
char ***userids
);
```

Parameters

Input

acl A pointer to the access control list.

Output

count The number of user names returned.

userids
An array of pointers to the user names returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

Description

Returns the user names that are included in the specified access control list. You must call the `ivadmin_acl_get()` function to obtain the `ivadmin_acl` object before using this function.

Command-line equivalent:

```
pdadmin acl show ACL_name
```

The list of users is part of the information returned in the **pdadmin** command.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_removeanyother()

This API removes the access control list entry for the user **any-other** from the specified access control list.

Syntax

```
unsigned long ivadmin_acl_removeanyother(
ivadmin_context ctx,
const char *aclid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The name of the access control list.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Removes the access control list entry for the user **any-other** from the specified access control list.

Command-line equivalent:

```
pdadmin acl modify ACL_name remove any-other
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_removegroup()

This API removes the access control list entry for the specified group from the specified access control list.

Syntax

```
unsigned long ivadmin_acl_removegroup(
ivadmin_context ctx,
const char *aclid,
const char *groupid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The name of the access control list.

groupid
The name of the group entry to remove from the access control list.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Removes the access control list entry for the specified group from the specified access control list.

Command-line equivalent:

```
pdadmin acl modify ACL_name remove group group_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_removeunauth()

This API removes the access control list entry for the user **unauthenticated** from the specified access control list.

Syntax

```
unsigned long ivadmin_acl_removeunauth(
ivadmin_context ctx,
const char *aclid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The name of the access control list.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Removes the access control list entry for the user **unauthenticated** from the specified access control list.

Command-line equivalent:

```
pdadmin acl modify ACL_name remove unauthenticated
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_removeuser()

This API removes the access control list entry for the specified user from the specified access control list.

Syntax

```
unsigned long ivadmin_acl_removeuser(
ivadmin_context ctx,
const char *aclid,
const char *userid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The name of the access control list.

userid The name of the user entry to remove from the access control list.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Removes the access control list entry for the specified user from the specified access control list.

Command-line equivalent:

```
pdadmin acl modify ACL_name remove user user_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_setanyother()

This API creates or changes the access control list entry for the user **any-other** in the access control list.

Syntax

```
unsigned long ivadmin_acl_setanyother(
ivadmin_context ctx,
const char *aclid,
const char *actions,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The access control list name.

actions
The new permissions for this access control list entry. This string consists of single-letter permission codes.

Each action is represented by a single alphabetic character. Default actions are provided in the primary action group by Security Access Manager. These default actions, such as `A` for add, or `v` for view, are listed in the *IBM Security Access Manager for Web: Administration Guide*.

Actions in the primary action group can be specified first without the name of the action group. Otherwise, the action group name must precede them. Actions in other action groups must always be preceded with the action group name, which is enclosed in brackets ([]).
For example, to set an entry so that it contains the add and view actions from the primary action group, along with the P, B, and J actions from the Admin2 action group, and the b and C actions from the Auditors action group, any of the following strings can be used:

```
Av[Admin2]PBJ[Auditors]bC
[primary]Av[Admin2]PBJ[Auditors]bC
[Auditors]bC[Admin2]PBJ[primary]Av
[Admin2]PBJ[primary]Av[Auditors]bC
```

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the access control list entry for the user **any-other** in the access control list.

Command-line equivalent:

```
pdadmin acl modify ACL_name set any-other perms
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_setdescription()

This API creates or changes the description for the specified access control list.

Syntax

```
unsigned long ivadmin_acl_setdescription(
ivadmin_context ctx,
const char *aclid,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The access control list name.

description
The new description.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the description for the specified access control list.

Command-line equivalent:

```
pdadmin acl modify ACL_name description description
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_setgroup()

This API creates or changes the access control list (ACL) entry for the specified group in the specified access control list.

Syntax

```
unsigned long ivadmin_acl_setgroup(
ivadmin_context ctx,
const char *aclid,
const char *groupid,
const char *actions,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The access control list name.

groupid
The access control list entry for a specified group.

actions
The new permissions for this access control list entry. This string consists of single-letter permission codes.

Each action is represented by a single alphabetic character. Default actions are provided in the primary action group by Security Access Manager. These default actions, such as `A` for add, or `v` for view, are listed in the *IBM Security Access Manager for Web: Administration Guide*.

Actions in the primary action group can be specified first without the name of the action group. Otherwise, the action group name must precede them. Actions in other action groups must always be preceded with the action group name, which is enclosed in brackets ([]).
For example, to set an entry so that it contains the add and view actions from the primary action group, along with the P, B, and J actions from the Admin2 action group, and the b and C actions from the Auditors action group, any of the following strings can be used:

```
Av[Admin2]PBJ[Auditors]bC
[primary]Av[Admin2]PBJ[Auditors]bC
[Auditors]bC[Admin2]PBJ[primary]Av
[Admin2]PBJ[primary]Av[Auditors]bC
```

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the ACL entry for the specified group in the specified access control list. The Security Access Manager user registry must contain an entry for the specified group. Then, you can call this function to add an entry for the group to an ACL.

Command-line equivalent:

```
pdadmin acl modify ACL_name set group group_name perms
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_setunauth()

This API creates or changes the access control list entry for the user specified by **unauthenticated** in the specified access control list.

Syntax

```
unsigned long ivadmin_acl_setunauth(
ivadmin_context ctx,
const char *aclid,
const char *actions,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The access control list name.

actions
The new permissions for this access control list entry. This string consists of single-letter permission codes.

Each action is represented by a single alphabetic character. Default actions are provided in the primary action group by Security Access Manager. These default actions, such as `A` for add, or `v` for view, are listed in the *IBM Security Access Manager for Web: Administration Guide*.

Actions in the primary action group can be specified first without the name of the action group. Otherwise, the action group name must precede them. Actions in other action groups must always be preceded with the action group name, which is enclosed in brackets ([]).
For example, to set an entry so that it contains the add and view actions from the primary action group, along with the P, B, and J actions from the Admin2 action group, and the b and C actions from the Auditors action group, any of the following strings can be used:

```
Av[Admin2]PBJ[Auditors]bC
[primary]Av[Admin2]PBJ[Auditors]bC
[Auditors]bC[Admin2]PBJ[primary]Av
[Admin2]PBJ[primary]Av[Auditors]bC
```

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the access control list entry for the user specified by **unauthenticated** in the specified access control list.

Command-line equivalent:

`pdadmin acl modify` *`ACL_name`* `set unauthenticated` *`perms`*

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_acl_setuser()

This API creates or changes the entry for the specified user in the specified access control list.

Syntax

```
unsigned long ivadmin_acl_setuser(
ivadmin_context ctx,
const char *aclid,
const char *userid,
const char *actions,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The access control list name.

userid The access control list entry for the specified user.

actions
The new permissions for this access control list entry. This string consists of single-letter permission codes.

Each action is represented by a single alphabetic character. Default actions are provided in the primary action group by Security Access Manager. These default actions, such as `A` for add, or `v` for view, are listed in the *IBM Security Access Manager for Web: Administration Guide*.

Actions in the primary action group can be specified first without the name of the action group. Otherwise, the action group name must precede them. Actions in other action groups must always be preceded with the action group name, which is enclosed in brackets ([]).
For example, to set an entry so that it contains the add and view actions from the primary action group, along with the P, B, and J actions from the Admin2 action group, and the b and C actions from the Auditors action group, any of the following strings can be used:

```
Av[Admin2]PBJ[Auditors]bC
[primary]Av[Admin2]PBJ[Auditors]bC
[Auditors]bC[Admin2]PBJ[primary]Av
[Admin2]PBJ[primary]Av[Auditors]bC
```

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the entry for the specified user in the specified access control list.

Call this function to specify the permissions that the user is permitted to do. For a list of the default Security Access Manager actions, see the section about default Security Access Manager permissions for actions in the *IBM Security Access Manager for Web: Administration Guide*. The Security Access Manager user registry must contain an entry for the specified user. Then, you can use this function to add an entry for the user to an access control list.

Command-line equivalent:

```
pdadmin acl modify ACL_name set user user_name perms
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_action_create()

This API defines a new action (permission) code in the primary action group.

Syntax

```
unsigned long ivadmin_action_create(
ivadmin_context ctx,
const char *actionid,
const char *description,
const char *type,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

actionid
An action identifier. This identifier must be a single-letter code that does not conflict with existing permission codes. The input is left as a string for future expansion.

description
The description of a permission code. This description occurs in Web Portal Manager.

type The label for action category. This label occurs in Web Portal Manager.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Defines a new action (permission) code in the primary action group.

Each action group can contain 32 action codes. The default action group contains the 18 predefined Security Access Manager action codes. Therefore, you can call `ivadmin_action_create()` to add up to 14 new action codes to the primary group.

Action codes consist of one alphabetic character (a–z or A–Z). Action codes are case-sensitive. Each action code can be used only one time in an action group. Be sure that you do not attempt to redefine the default Security Access Manager action codes when adding new codes to the primary group.

Command-line equivalent:

```
pdadmin action create name description action_type
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_action_create_in_group()

This API defines a new action (permission) code in the specified action group.

Syntax

```
unsigned long ivadmin_action_create_in_group(
ivadmin_context ctx,
const char *actionid,
const char *description,
```

```
const char *type,
const char *groupname,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

actionid
The action identifier. This identifier must be a single-letter code that does not conflict with existing permission codes. The input is left as a string for future expansion.

description
The description of the permission code. This description occurs in Web Portal Manager.

type The label for the action category. This label occurs in Web Portal Manager.

groupname
The name of the action group in which to create the action.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Defines a new action (permission) code in the specified action group. Call this function to add an action code to a user-defined extended action group.

Action codes consist of one alphabetic character (a–z or A–Z). Action codes are case-sensitive. Each action code can be used only one time in an action group. Security Access Manager supports up to 32 actions in one action group.

Command-line equivalent:

```
pdadmin action create name description action_type action_group_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_action_delete()

This API deletes an action (permission) code from the primary action group.

Syntax

```
unsigned long ivadmin_action_delete(
ivadmin_context ctx,
const char *actionid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

actionid
The action identifier. This identifier must be a single-letter code that identifies the permission to delete.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes an action (permission) code from the primary action group.

Command-line equivalent:

```
pdadmin action delete name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_action_delete_from_group()

This API deletes an action (permission) code from the specified action group.

Syntax

```
unsigned long ivadmin_action_delete_from_group(
ivadmin_context ctx,
const char *actionid,
const char *groupname,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

actionid
The action identifier. This identifier must be a single-letter code that identifies the permission to delete.

groupname
The name of the action group from which to delete the action.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes an action (permission) code from the specified action group.

Command-line equivalent:

```
pdadmin action delete name action_group_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_action_getdescription()

This API returns the description for the specified action.

Syntax

```
const char* ivadmin_action_getdescription(
ivadmin_action action
);
```

Parameters

Input

action A pointer to the action.

Description

Returns the description for the specified action.

Do not free this description. This data is maintained in the `ivadmin_action` object.

Command-line equivalent:

```
pdadmin action list
```

The **pdadmin** command lists information about all the actions, including the description for each action.

Return values

Returns the description for the specified action. The maximum length for a description is 1024 characters.

ivadmin_action_getid()

This API returns the action identifier for the specified action.

Syntax

```
const char* ivadmin_action_getid(
ivadmin_action action
);
```

Parameters

Input

action A pointer to the action.

Description

Returns the single character action identifier for the specified action.

Do not free this action identifier. This data is maintained in the `ivadmin_action` structure.

Command-line equivalent:

```
pdadmin action list
```

This **pdadmin** command lists information about all the actions, including the code for each action.

Return values

Returns the single character action identifier for the specified action, or NULL if an error occurred.

ivadmin_action_gettype()

This API returns the type, or label, for the action category that is associated with the specified action.

Syntax

```
const char* ivadmin_action_gettype(
ivadmin_action action
);
```

Parameters

Input

action A pointer to the action.

Description

Returns the type, or label, of the action category that is associated with the specified action.

Do not free this type or label. This data is maintained in the `ivadmin_action` structure.

Command-line equivalent:

```
pdadmin action list
```

This **pdadmin** command lists information about all the actions, including the type for each action.

Return values

Returns the type, or label, of the action category that is associated with the specified action. There is no limit to the length of the label.

ivadmin_action_group_create()

This API creates an action group with the specified name.

Syntax

```
unsigned long ivadmin_action_group_create(
ivadmin_context ctx,
const char *groupname,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

groupname
The name of the new action group.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates an action group with the specified name. Security Access Manager supports a maximum of 32 action groups.

Command-line equivalent:

```
pdadmin action group create action_group_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_action_group_delete()

This API deletes the specified action group and all the actions that belong to the specified group.

Syntax

```
unsigned long ivadmin_action_group_delete(
ivadmin_context ctx,
const char *groupname,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

groupname
The name of the action group to delete.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified action group and all the actions that belong to the specified group.

Command-line equivalent:

```
pdadmin action group delete action_group_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_action_group_list()

This API lists all the defined action group names.

Syntax

```
unsigned long ivadmin_action_group_list(
ivadmin_context ctx,
unsigned long *count,
char ***names,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

count The number of action group names returned.

names An array of pointers to the action group names returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists all the defined action group names.

Command-line equivalent:

```
pdadmin action group list
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_action_list()

This API lists all the defined action (permission) codes from the primary action group.

Syntax

```
unsigned long ivadmin_action_list(
ivadmin_context ctx,
unsigned long *count,
ivadmin_action **actions,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

count The number of actions returned.

actions
An array of pointers to the actions returned. You must free the data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists all the defined action (permission) codes from the primary action group. Use this function to obtain an opaque list of actions. You can then use additional

functions to obtain information from each action (ivadmin_action). For example, you can use ivadmin_action_getdescription() to obtain a description for the specified ivadmin_action object.

Command-line equivalent:

```
pdadmin action list
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_action_list_in_group()

This API lists all the defined action (permission) codes from the specified action group.

Syntax

```
unsigned long ivadmin_action_list_in_group(
ivadmin_context ctx,
const char *actiongroup,
unsigned long *count,
ivadmin_action **actions,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

actiongroup
The name of the action group to list.

Output

count The number of actions returned.

actions
An array of pointers to the actions returned. You must free the data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists all the defined action (permission) codes from the specified action group.

Command-line equivalent:

```
pdadmin action list action_group_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_authzrule_create()

This API creates the specified authorization rule object.

Syntax

```
unsigned long ivadmin_authzrule_create(
ivadmin_context ctx,
const char *ruleid,
const char *ruledesc,
const char *ruletext,
const char *failreason,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ruleid The name of the authorization rule to create.

ruledesc
The description of the rule. Can be NULL.

ruletext
The rule text in XSL format.

failreason
A string that represents a fail reason code. Authorization might be denied as a result of evaluating this rule, but other authorization checks (such as a POP or an ACL) might be successful. In this case, this reason code is returned to the application that makes the authorization check. Can be NULL.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Creates an authorization rule. You can attach an authorization rule to a protected object. The rule attachment compares the user credential and application context attributes against the rule when authorizing access to the protected object.

Command-line equivalent:

```
pdadmin authzrule create rulename ruletext [-desc description]
 [-failreason failreason]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_authzrule_delete()

This API deletes the specified authorization rule object.

Syntax

```
unsigned long ivadmin_authzrule_delete(
ivadmin_context ctx,
const char *ruleid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ruleid The name of the authorization rule to delete.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Deletes an authorization rule.

Command-line equivalent:

```
pdadmin authzrule delete rulename
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_authzrule_get()

This API returns the specified authorization rule object.

Syntax

```
unsigned long ivadmin_authzrule_get(
ivadmin_context ctx,
const char *ruleid,
ivadmin_authzrule *rule,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ruleid The name of the authorization rule to return.

Output

rule The authorization rule object. Free this object when it is no longer needed.

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Returns the specified authorization rule object.

Command-line equivalent:

```
pdadmin authzrule show rulename
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_authzrule_getdescription()

This API returns the description for the specified authorization rule.

Syntax

```
const char* ivadmin_authzrule_getdescription(
ivadmin_authzrule rule
);
```

Parameters

Input

rule A pointer to the authorization rule object.

Description

Returns the description from the specified authorization rule object. You must call the `ivadmin_authzrule_get()` function to obtain an `ivadmin_authzrule` object

before using this function. Do not free the character string that is returned. This data is maintained in the ivadmin_authzrule object.

Command-line equivalent:

```
pdadmin authzrule show rulename
```

The description is part of the information returned by the **pdadmin** command.

Return values

Returns the description for the specified authorization rule.

ivadmin_authzrule_getfailreason()

This API returns the fail reason, if any, for the specified authorization rule.

Syntax

```
const char* ivadmin_authzrule_getfailreason(
ivadmin_authzrule rule
);
```

Parameters

Input

rule A pointer to the authorization rule object.

Description

Returns the fail reason from the specified authorization rule object. You must call the ivadmin_authzrule_get() function to obtain an ivadmin_authzrule object before using this function. Do not free the character string that is returned. This data is maintained in the ivadmin_authzrule object.

Command-line equivalent:

```
pdadmin authzrule show rulename
```

The fail reason is part of the information returned by the pdadmin command.

Return values

Returns the fail reason, if any, for the specified authorization rule. Returns an empty string when there is no fail reason.

ivadmin_authzrule_getid()

This API returns the name (ID) for the specified authorization rule.

Syntax

```
const char* ivadmin_authzrule_getid(
ivadmin_authzrule rule
);
```

Parameters

Input

rule A pointer to the authorization rule object.

Description

Returns the rule name (ID) from the specified authorization rule object. You must call the `ivadmin_authzrule_get()` function to obtain an `ivadmin_authzrule` object before using this function. Do not free the character string that is returned. This data is maintained in the `ivadmin_authzrule` object.

Command-line equivalent:

```
pdadmin authzrule show rulename
```

The rule name is part of the information returned by the **pdadmin** command.

Return values

Returns the name of the specified authorization rule.

ivadmin_authzrule_getruletext()

This API returns the text for the specified authorization rule.

Syntax

```
const char* ivadmin_authzrule_getruletext(
ivadmin_authzrule rule
);
```

Parameters

Input

rule A pointer to the authorization rule object.

Description

Returns the text from the specified authorization rule object. You must call the `ivadmin_authzrule_get()` function to obtain an `ivadmin_authzrule` object before using this function. Do not free the character string that is returned. This data is maintained in `ivadmin_authzrule` object.

Command-line equivalent:

```
pdadmin authzrule show rulename
```

The rule text is part of the information returned by the **pdadmin** command.

Return values

Returns the rule text, in XSL format, for the specified authorization rule.

ivadmin_authzrule_list()

This API lists the names of all the registered authorization rules.

Syntax

```
unsigned long ivadmin_authzrule_list(
ivadmin_context ctx,
unsigned long *count,
char ***ruleids,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

count The number of authorization rule strings returned.

ruleids
An array of pointers to the authorization rule strings returned. You must free the character data referenced by each pointer as well as the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists the names of all the registered authorization rules.

Command-line equivalent:

```
pdadmin authzrule list
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_authzrule_setdescription()

This API creates or changes the authorization rule description.

Syntax

```
unsigned long ivadmin_authzrule_setdescription(
ivadmin_context ctx,
const char *ruleid,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ruleid The name of the authorization rule.

description
The new description. Cannot be NULL. An empty string can be specified to clear the description.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Creates or changes the description for the specified authorization rule.

Command-line equivalent:

```
pdadmin authzrule modify rulename description description
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_authzrule_setfailreason()

This API creates or modifies the authorization rule fail reason.

Syntax

```
unsigned long ivadmin_authzrule_setruletext(
ivadmin_context ctx,
const char *ruleid,
const char *failreason,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ruleid The name of the authorization rule.

failreason
The new fail reason code. Authorization might be denied, but other authorization checks (such as a POP or an ACL) might be successful. In this case, this reason code is returned to the application that makes the authorization check. Cannot be NULL. An empty string can be specified to clear the fail reason code.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Creates or modifies the fail reason for the specified authorization rule

Command-line equivalent:

```
pdadmin authzrule modify rulename failreason failreason
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_authzrule_setruletext()

This API creates or modifies the authorization rule text.

Syntax

```
unsigned long ivadmin_authzrule_setruletext(
ivadmin_context ctx,
const char *ruleid,
const char *ruletext,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ruleid The name of the authorization rule.

ruletext
The new text for the rule, in XSL format.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Creates or modifies the rule text for the specified authorization rule.

Command-line equivalent:

```
pdadmin authzrule modify rulename ruletext ruletext
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_addreplica2()

This API adds a replica entry to the configuration file.

Syntax

```
unsigned long ivadmin_cfg_addreplica2(
ivadmin_context ctx,
const char *cfg_file_name,
const char *ivacld_host,
int ivacld_port,
int ivacld_rank,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The local context created by the `ivadmin_context_createlocal()` function.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, the specified parameter must be a fully qualified path name.

ivacld_host
The TCP host name of the `ivacld` server.

ivacld_port
The listening port number of the `ivacld` replica server. This value is the port number on which the `ivacld` server listens for requests.

ivacld_rank
The replica order of preference among other replicas.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Adds a replica entry to the configuration file. A replica entry is the host name, port number, and rank of an `ivacld` server with which the application server might communicate.

Command-line equivalent:

```
svrsslcfg -add_replica -f cfg_file -h host_name [-p port] [-k rank]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_chgreplica2()

This API changes parameters of a replica entry in the configuration file.

Syntax

```
unsigned long ivadmin_cfg_chgreplica2(
ivadmin_context ctx,
const char *cfg_file_name,
const char *ivacld_host,
int ivacld_port,
int ivacld_rank,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The local context created by the `ivadmin_context_createlocal()` function.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, the specified parameter must be a fully qualified path name.

ivacld_host
The TCP host name of the `ivacldv` server.

ivacld_port
The listening port number of the `ivacld` replica server. This value is the port number on which the `ivacld` server listens for requests.

ivacld_rank
The replica order of preference among other replicas.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Changes parameters of a replica entry in the configuration file. A replica entry is the host name, port number, and rank of an `ivacld` server with which the application server might communicate.

Command-line equivalent:

```
svrsslcfg -chg_replica -f cfg_file -h host_name [-p port] [-k rank]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_configureserver3()

This API configures an application API server.

Syntax

```
unsigned long ivadmin_cfg_configureserver3(
ivadmin_context ctx,
const char *cfg_file_name,
const char *kdb_dir_name,
const char *application_name,
const char *host_name,
ivadmin_cfg_servermode server_mode,
const char *server_pwd,
int enable_listening,
int listening_port,
int enable_refresh,
int kdb_pwd_life,
int ssl_timeout,
const char *appl_cert,
unsigned long group_count,
const char **groups,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server. The application server is defined in the domain to which this context is authenticated.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, the parameter must be a fully qualified path name.

kdb_dir_name
The keyring database directory.

application_name
The name for the application or server. The *application_name* and *host_name* combination must be unique, as this combination is used to uniquely identify the application.

host_name
The host name on which the application runs.

server_mode

The Security Access Manager server mode. The data type `ivadmin_cfg_servermode` is an enumerated data type. Enumerated values are (1) *local* and (2) *remote*.

server_pwd
The password for the application server. If NULL or an empty password is specified, a random password is automatically generated for the server account.

enable_listening
The listening-enabled setting in the configuration file. Specify `IVADMIN_TRUE` to enable listening and `IVADMIN_FALSE` to disable listening.

listening_port
The TCP/IP port on which the application listens.

enable_refresh
The certificate automatic refresh support setting. Specify `IVADMIN_TRUE` to enable or `IVADMIN_FALSE` to disable.

kdb_pwd_life
The keyring database password life, specified in days. If it is 0, a default of 183 days is used.

ssl_timeout
The Secure Sockets Layer (SSL) session timeout value in seconds. If it is 0, a default of 7200 is used.

appl_cert
The name of the file that contains a base-64 encoded SSL certificate. This parameter is optional. If specified, the certificate is stored in the keyring database with a label of `APPL_LDAP_CERT`. Typical use of this parameter is to store the certificate authority certificate that the application uses when it authenticates directly to the user registry.

Do not confuse this certificate with the certificate that is used to authenticate with the Security Access Manager policy server. The certificate specified by this parameter does not participate in authentication with the policy server. It is for application use only and allows the application to use a single keyring database for all SSL certificates.

group_count
The number of groups of which the application server must be made a member.

groups
A list of groups of which the application server must be made a member.

description
The description of the application server. Cannot be `NULL` but can be an empty string.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Configures an authorization API server by updating the configuration file and creating the keyring database. The combination of the `application_name` and `host_name` must be unique, as this combination is used to uniquely identify the application.

Command-line equivalent:

```
svrsslcfg –config -f cfg_file_name -d kdb_dir_name -n server_name \
 -s server_mode -r listening_port -P admin_pwd [-S server_pwd] \
 [-A admin_ID] [-t ssl_timeout] [-e kbd_pwd_life] [-l listening_mode]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_configureserver4()

This API configures an application API server.

Syntax

```
unsigned long ivadmin_cfg_configureserver4(
ivadmin_context ctx,
const char *cfg_file_name,
const char *kdb_dir_name,
const char *application_name,
const char *host_name,
ivadmin_cfg_servermode server_mode,
const char *server_pwd,
int enable_listening,
int listening_port,
int enable_refresh,
int kdb_pwd_life,
int ssl_timeout,
const char *appl_cert,
unsigned long group_count,
const char **groups,
const char *description,
ivadmin_response *rsp
int ldap_conf_server_config
);
```

Parameters

Input

ctx
The context to use when communicating with the policy server. The application server is defined in the domain to which this context is authenticated.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, the parameter must be a fully qualified path name.

kdb_dir_name
The keyring database directory.

application_name
The name for the application or server. The *application_name* and *host_name* combination must be unique, as this combination is used to uniquely identify the application.

host_name
The host name on which the application runs.

server_mode

The Security Access Manager server mode. The data type `ivadmin_cfg_servermode` is an enumerated data type. Enumerated values are (1) *local* and (2) *remote*.

server_pwd
The password for the application server. If NULL or an empty password is specified, a random password is automatically generated for the server account.

enable_listening
The listening-enabled setting in the configuration file. Specify `IVADMIN_TRUE` to enable listening and `IVADMIN_FALSE` to disable listening.

listening_port
The TCP/IP port on which the application listens.

enable_refresh
The certificate automatic refresh support setting. Specify `IVADMIN_TRUE` to enable or `IVADMIN_FALSE` to disable.

kdb_pwd_life
The keyring database password life, specified in days. If it is 0, a default of 183 days is used.

ssl_timeout
The Secure Sockets Layer (SSL) session timeout value in seconds. If it is 0, a default of 7200 is used.

appl_cert
The name of the file that contains a base-64 encoded SSL certificate. This parameter is optional. If specified, the certificate is stored in the keyring database with a label of `APPL_LDAP_CERT`. Typical use of this parameter is to store the certificate authority certificate that the application uses when it authenticates directly to the user registry.

Do not confuse this certificate with the certificate that is used to authenticate with the Security Access Manager policy server. The certificate specified by this parameter does not participate in authentication with the policy server. It is for application use only and allows the application to use a single keyring database for all SSL certificates.

group_count
The number of groups of which the application server must be made a member.

groups
A list of groups of which the application server must be made a member.

description
The description of the application server. The value can be an empty string, but cannot be `NULL`.

ldap_conf_server_config
When set to true, [ldap] `ldap-server-config` is set up to point `PolicyDirector/etc/ldap.conf` in the configuration file of the server. The LDAP host port and replica configuration are obtained from this common location.

When set to false, the host and port values are set directly in the configuration file of the server. The replica value is not updated automatically. You must manually configure the value.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Configures an authorization API server by updating the configuration file and creating the keyring database. The combination of the *application_name* and *host_name* must be unique, as this combination is used to uniquely identify the application.

Command-line equivalent:

```
svrsslcfg –config -f cfg_file_name -d kdb_dir_name -n server_name \
 -s server_mode -r listening_port -P admin_pwd [-S server_pwd] \
 [-A admin_ID] [-t ssl_timeout] [-e kbd_pwd_life] [-l listening_mode]
 [-ldap.conf mode]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_getvalue()

This API returns a value that is associated with a specific key in a specific stanza of the specified configuration file.

Syntax

```
unsigned long ivadmin_cfg_getvalue(
ivadmin_context ctx,
const char *cfg_file_name,
const char *stanza,
const char *key,
int *count,
char ***values
ivadmin_response *rsp
);
```

Parameters

Input

ctx The local context created by the `ivadmin_context_createlocal()` function.

cfg_file_name
The configuration file to examine. This value can be a fully qualified path name to a configuration file or a reference to a Security Access Manager configuration file. A reference to a Security Access Manager configuration file is indicated by using one of the following constants:

IVADMIN_CFG_CFGFILE_AMRTE
Security Access Manager runtime configuration file

IVADMIN_CFG_CFGFILE_AMMGRD
Security Access Manager policy server configuration file

IVADMIN_CFG_CFGFILE_AMACLD
Security Access Manager authorization server configuration file

IVADMIN_CFG_CFGFILE_AMPROXY
Security Access Manager proxy server configuration file

stanza The name of the stanza under which the input key is located. This value can be a user-defined stanza name, or a reference to a Security Access Manager stanza. Stanzas used by Security Access Manager components are listed in the appendixes of the *IBM Security Access Manager for Web: Administration Guide*. The following constants can be used:

IVADMIN_CFG_STANZA_SSL
SSL entries

IVADMIN_CFG_STANZA_POLICY_SVR
Security Access Manager policy server entries

key The name of the key whose value is to be returned. This value can be a user-defined key name or a reference to a Security Access Manager key. Keys used by Security Access Manager components are listed in the appendixes of the *IBM Security Access Manager for Web: Administration Guide*. The following constants can be used:

IVADMIN_CFG_KEY_POLICY_SVR
Security Access Manager policy server host name and port

IVADMIN_CFG_KEY_SSL_KEYFILE
SSL key file path

IVADMIN_CFG_KEY_SSL_STASHFILE
SSL stash file path

IVADMIN_CFG_KEY_SSL_PASSWORD
SSL password

Output

count The number of values returned. A zero is returned if there is no value associated with the key.

values An array of pointers to the values returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. The output might be empty or contain more than one error, informational, and warning messages. Free this object when it is no longer needed.

Description

Returns the value of a specific key from a specific stanza in a configuration file. All data is returned in an array of character strings, because some keys might be multi-valued. The caller must have the necessary operating system permissions to read the configuration file or database.

Command-line equivalent:

```
pdadmin config show config-file stanza key
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_removevalue()

This API removes a value or an array of values that are associated with the specified key in the specified stanza of the specified configuration file.

Syntax

```
unsigned long ivadmin_cfg_removevalue(
ivadmin_context ctx,
const char *cfg_file_name,
const char *stanza,
const char *key,
int count,
char **values
ivadmin_response *rsp
);
```

Parameters

Input

ctx The local context created by the `ivadmin_context_createlocal()` function.

cfg_file_name
The configuration file to examine. This value can be a fully qualified path name to a configuration file or a reference to a Security Access Manager configuration file. A reference to a Security Access Manager configuration file is indicated by using one of the following constants:

IVADMIN_CFG_CFGFILE_AMRTE
Security Access Manager runtime configuration file

IVADMIN_CFG_CFGFILE_AMMGRD
Security Access Manager policy server configuration file

IVADMIN_CFG_CFGFILE_AMACLD
Security Access Manager authorization server configuration file

IVADMIN_CFG_CFGFILE_AMPROXY
Security Access Manager proxy server configuration file

stanza The name of the stanza under which the input key is located. This value can be a user-defined stanza name, or a reference to a Security Access Manager stanza. Stanzas used by Security Access Manager components are listed in the appendixes of the *IBM Security Access Manager for Web: Administration Guide*. The following constants can be used:

IVADMIN_CFG_STANZA_SSL
SSL entries

IVADMIN_CFG_STANZA_POLICY_SVR
Security Access Manager policy server entries

key The name of the key whose value is to be modified. This value can be a

user-defined key name or a reference to a Security Access Manager key. Keys used by Security Access Manager components are listed in the appendixes of the *IBM Security Access Manager for Web: Administration Guide*.

Alternatively, specify a NULL *key* parameter and NULL array of *values* parameter to delete the stanza.

The following constants can be used:

IVADMIN_CFG_KEY_POLICY_SVR
Security Access Manager policy server host name and port

IVADMIN_CFG_KEY_SSL_KEYFILE
SSL key file path

IVADMIN_CFG_KEY_SSL_STASHFILE
SSL stash file path

IVADMIN_CFG_KEY_SSL_PASSWORD
SSL password

count The number of items in the *values* input array. The count must be zero if you want to delete *stanza* or *key*.

values An array of character pointers to configuration values to remove from the input key. Alternatively, specify a NULL array of *values* parameter to delete the specified *key* parameter.

Output

rsp The response object. Indicates the success or failure of the function. The output might be empty or contain more than one error, informational, and warning messages. Free this object when it is no longer needed.

Description

Removes the specified values from the specified key in the specified configuration file. If the *key* and *values* parameters are NULL, the entire stanza is removed. If only the values parameter is NULL, the specified key is removed. The caller must have the necessary operating system permissions to modify the configuration file.

Command-line equivalent:

To remove a value:

```
pdadmin config modify keyvalue remove config-file stanza key value
```

To remove a key:

```
pdadmin config modify keyvalue remove config-file stanza key
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

If all the values in the *values* array do not exist, `IVADMIN_FALSE` is returned and output *rsp* contains an error message. If at least one of the values specified in the array *values* exists, the function completes successfully.

ivadmin_cfg_renewservercert()

This API renews the server Secure Sockets Layer (SSL) certificate.

Syntax

```
unsigned long ivadmin_cfg_renewservercert(
ivadmin_context ctx,
const char *cfg_file_name,
const char *server_name,
const char *host_name,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, this parameter must be a fully qualified path name.

server_name
The unique server name.

host_name
The host name on which the application runs.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Use this API to refresh the certificate used to authenticate with the policy server if it expires or is compromised. The application must be stopped before using this API.

Command-line equivalent:

```
svrsslcfg -chgcert -f cfg_file -n server_name [-A admin_id] -P admin_pwd
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_rmvreplica2()

This API removes a replica entry from the configuration file. A replica entry is the host name, port number, and rank of an authorization server with which the application server might communicate.

Syntax

```
unsigned long ivadmin_cfg_rmvreplica2(
ivadmin_context ctx,
const char *cfg_file_name,
const char *ivacld_host,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The local context created by the `ivadmin_context_createlocal()` function.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, this parameter must be a fully qualified path name.

ivacld_host
The TCP host name of the `ivacld` server.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Removes a replica entry from the configuration file.

Command-line equivalent:

```
svrsslcfg -chg_replica -f cfg_file -h host_name [-p port] [-k rank]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_setapplicationcert2()

This API replaces the optional application certificate authority certificate and the optional Secure Sockets Layer (SSL) certificate in the keyring database.

Syntax

```
unsigned long ivadmin_cfg_setapplicationcert2(
ivadmin_context ctx,
const char *cfg_file_name,
const char *appl_cert,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The local context created by the `ivadmin_context_createlocal()` function.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, this parameter must be a fully qualified path name.

appl_cert
The name of the file that contains a base-64 encoded SSL certificate. This parameter is optional. If specified, the certificate is stored in the keyring database with a label of `APPL_LDAP_CERT`. Typical use of this parameter is to store the certificate authority certificate that the application uses when it authenticates directly to the user registry.

Do not confuse this certificate with the certificate that is used to authenticate with the Security Access Manager policy server. The certificate specified by this parameter does not participate in authentication with the policy server. It is for application use only and allows the application to use a single keyring database for all SSL certificates.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Replaces the optional application certificate authority certificate and the optional Secure Sockets Layer (SSL) certificate in the keyring database.

The application must be stopped before invoking this API.

Command-line equivalent:

```
svrsslcfg -modify -f cfg_file [-t timeout] [-C cert_file] [-l
listening_mode]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_setkeyringpwd2()

This API refreshes or changes the keyring database random password.

Syntax

```
unsigned long ivadmin_cfg_setkeyringpwd2(
ivadmin_context ctx,
const char *cfg_file_name,
int kdb_pwd_life,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The local context created by the `ivadmin_context_createlocal()` function.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, this parameter must be a fully qualified path name.

kdb_pwd_life
The keyring database password life in days. If 0, a default of 183 days is used.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Use this API to refresh or change the keyring database random password. A new random password is created in the stash file. The application must be stopped to execute this API.

Command-line equivalent:

```
svrsslcfg -chgcert -f <cfg_file_name> [-A admin_id] [-P admin_pwd]
```

where *cfg_file_name* is the configuration file to use. Unless the configuration file is in the current directory, this parameter must be a fully qualified path name.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_setlistening2()

This API sets or resets the enable-listening parameter in the configuration file.

Syntax

```
unsigned long ivadmin_cfg_setlistening(
ivadmin_context ctx,
const char *cfg_file_name,
int enable_listening,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The local context created by the `ivadmin_context_createlocal()` function.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, this parameter must be a fully qualified path name.

enable_listening
The listening-enabled setting in the configuration file. Specify `IVADMIN_TRUE` to enable listening and `IVADMIN_FALSE` to disable listening.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets or resets the enable-listening parameter in the configuration file. This flag determines whether the application server listens for communications from the policy server.

The listening port in the configuration file must be nonzero to enable listening. Otherwise, an error is returned. The application must be stopped and restarted after calling this API.

Command-line equivalent:

```
svrsslcfg -config -f cfg_file_name -l {yes | no | unset}
```

Where *cfg_file_name* is the configuration file to use. Unless the configuration file is in the current directory, this parameter must be a fully qualified path name.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_setport2()

This API changes the listening port number of the application and updates the port number in the configuration file.

Syntax

```
unsigned long ivadmin_cfg_setport2(
ivadmin_context ctx,
const char *cfg_file_name,
int listening_port,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The local context created by the `ivadmin_context_createlocal()` function.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, this parameter must be a fully qualified path name.

listening_port
The TCP/IP port on which the application listens.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Changes the listening port number of the application and updates the port number in the configuration file. The application server uses this port to communicate with the policy server.

The server must be stopped and restarted to activate this change. If the port value is set to zero, the listening-enabled flag are set to disable.

Command-line equivalent:

```
svrsslcfg –config -f cfg_file_name -d kdb_dir_name -n server_name \
-s server_type -r listening_port -P admin_pwd [-S server_pwd] \
 [-A admin_ID] [-t ssl_timeout] [-e kbd_pwd_life] [-l listening_mode]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_setssltimeout2()

This API changes the Secure Sockets Layer (SSL) timeout value in the configuration file.

Syntax

```
unsigned long ivadmin_cfg_setssltimeout2(
ivadmin_context ctx,
const char *cfg_file_name,
int ssl_timeout,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The local context created by the `ivadmin_context_createlocal()` function.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, this parameter must be a fully qualified path name.

ssl_timeout
The Secure Sockets Layer (SSL) session timeout value in seconds. If it is 0, a default of 7200 is used

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Changes the Secure Sockets Layer (SSL) timeout value in the configuration file.

The application must be stopped and restarted to activate this change.

Command-line equivalent:

```
svrsslcfg -modify -f cfg_file [-t timeout] [-C cert_file] [-l
listening_mode]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_setsvrpwd()

This API sets the password on the specified server.

Syntax

```
unsigned long ivadmin_cfg_setsvrpwd(
ivadmin_context ctx,
const char *cfg_file_name,
const char *newpassword,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

cfg_file_name
The configuration file to examine. This value can be a fully qualified path name to a configuration file or a reference to a configuration file. A reference to a configuration file is indicated by using one of the following constants:

IVADMIN_CFG_CFGFILE_AMRTE
Security Access Manager runtime configuration file

IVADMIN_CFG_CFGFILE_AMMGRD
Security Access Manager policy server configuration file

IVADMIN_CFG_CFGFILE_AMACLD
Security Access Manager authorization server configuration file

IVADMIN_CFG_CFGFILE_AMPROXY
Security Access Manager proxy server configuration file

newpassword
The new password to set.

Output

rsp The response object. Indicates the success or failure of the function. The output might be empty or contain more than one error, informational, and warning messages. Free this object when it is no longer needed.

Description

Updates the password of the server user account. If the update of the user registry is successful, the local configuration is updated. The caller must have the necessary Security Access Manager permissions to modify the password in the user registry. The caller must also have the necessary operating system permissions to modify the configuration file or database.

Command-line equivalent:

```
config modify svrpassword config-file password
```

Where *config-file* is the local configuration file and *password* is the password you want to set.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_setvalue()

This API creates or modifies a value that is associated with a specific key in a specific stanza of the specified configuration file.

Syntax

```
unsigned long ivadmin_cfg_setvalue(
ivadmin_context ctx,
const char *cfg_file_name,
const char *stanza,
const char *key,
int count,
char **values,
int append,
int obfuscated_db,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The local context created by the `ivadmin_context_createlocal()` function.

cfg_file_name
The configuration file to use. This value can be a fully qualified path name to a configuration file or a reference to a Security Access Manager configuration file. A reference to a Security Access Manager configuration file is indicated by using one of the following constants:

IVADMIN_CFG_CFGFILE_AMRTE
Security Access Manager runtime configuration file

IVADMIN_CFG_CFGFILE_AMMGRD
Security Access Manager policy server configuration file

IVADMIN_CFG_CFGFILE_AMACLD
Security Access Manager authorization server configuration file

IVADMIN_CFG_CFGFILE_AMPROXY
Security Access Manager proxy server configuration file

stanza The name of the stanza under which the input key is located. This value can be a user-defined stanza name, or a reference to a Security Access Manager stanza. Stanzas used by Security Access Manager components are listed in the appendixes of the *IBM Security Access Manager for Web: Administration Guide*. The following constants can be used:

IVADMIN_CFG_STANZA_SSL
SSL entries

IVADMIN_CFG_STANZA_POLICY_SVR
Security Access Manager policy server entries

key The name of the key whose value is to be set. This value can be a user-defined key name or a reference to a Security Access Manager key. Keys used by Security Access Manager components are listed in the appendixes of the *IBM Security Access Manager for Web: Administration Guide*. The following constants can be used:

IVADMIN_CFG_KEY_POLICY_SVR
Security Access Manager policy server host name and port

IVADMIN_CFG_KEY_SSL_KEYFILE
SSL key file path

IVADMIN_CFG_KEY_SSL_STASHFILE
SSL stash file path

IVADMIN_CFG_KEY_SSL_PASSWORD
SSL password

count The number of values to set.

values An array of character pointers to configuration values that are to be set for the specified key.

append
Specifies whether the values must be appended to the current values that are associated with the key. If true, values are appended to the current values that are associated with the key. Any duplicate values are ignored. Otherwise, the input values replace any existing values for the key.

obfuscated_db
Specifies whether the value must be obfuscated. If true, the key is placed in the obfuscated section of the configuration file. Otherwise, the key is placed in the non-obfuscated section.

Output

rsp The response object. Indicates the success or failure of the function. The output might be empty or contain more than one error, informational, and warning messages. Free this object when it is no longer needed.

Description

Creates or modifies the value of a specific key in a configuration file. If obfuscation is requested, the configuration file identified by the **file** key in the **configuration-database** stanza is updated. The caller must have the necessary operating system permissions to modify the configuration file, otherwise, `IVADMIN_FALSE` is returned.

Command-line equivalent:

```
pdadmin config modify keyvalue set [-obfuscate] config-file stanza key value

pdadmin config modify keyvalue append [-obfuscate] config-file stanza key value
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_cfg_unconfigureserver()

This API unconfigures an application server.

Syntax

```
unsigned long ivadmin_cfg_unconfigureserver(
ivadmin_context ctx,
const char *cfg_file_name,
const char *server_name,
const char *host_name,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

cfg_file_name
The configuration file to use. Unless the configuration file is in the current directory, this parameter must be a fully qualified path name.

server_name
The name for the application or server. The *server_name* and *host_name* combination is used to uniquely identify the application.

host_name
The host name on which the application runs.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Unconfigures an application server. This function reports success even if the server was not configured. This command destroys the keyring, any objects in the user registry, and the access control list database for the server.

The application must be stopped before calling this function.

Command-line equivalent:

```
svrsslcfg –unconfig -f cfg_file_name -n server_name \
[-P admin_password] [-A admin_ID]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_cleardelcred()

This API clears the delegated credential for the context.

Syntax

```
unsigned long ivadmin_context_cleardelcred(
ivadmin_context ctx,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Clears the delegated credential for the context.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_create3()

This API creates a security context for communicating with the Security Access Manager policy server.

Syntax

```
unsigned long ivadmin_context_create3(
const char *userid,
const char *pwd,
const char *domain,
const char *codeset,
const char *serverhost,
unsigned long port,
const char *keyringfile,
const char *keyringstashfile,
const char *configfile,
ivadmin_context *ctx,
ivadmin_response *rsp
);
```

Parameters

Input

userid The administrator user name used for authentication. This user must have the appropriate administration authority to do the required administrative operations. Cannot be NULL.

pwd The administrator password. Cannot be NULL.

domain
The name of the domain to which authentication occurs. If this argument is IVADMIN_DOMAIN_MANAGEMENT, then the management domain is used. If this argument is IVADMIN_DOMAIN_LOCAL, then the local domain is used. These constants are defined in the ivadminapi.h file.

If NULL is specified, the domain is obtained from the ssl-local-domain key in the ssl stanza of the specified *configfile.*

codeset

Character codeset. Indicates how the application encodes its character data. Cannot be NULL. The following constants are defined in the ivadminapi.h file:

IVADMIN_CODESET_UTF8
Character data is encoded in UTF-8.

IVADMIN_CODESET_LOCAL
Character data is encoded in the local code page

serverhost
The policy server host name or IP address. If NULL is specified, the host name is obtained from the master-host key in the manager stanza in the specified configfile.

port The policy server listening port number. If `NULL` is specified, the port number is obtained from the `master-port` key in the `manager` stanza in the specified `configfile`.

keyringfile
The fully qualified path name to the Secure Sockets Layer (SSL) keyring file that contains the public key of the Security Access Manager policy server. If NULL is specified, the keyring file name is obtained from the `ssl-keyfile` key in the `ssl` stanza in the specified `configfile`.

keyringstashfile
The fully qualified path name to the stash file that contains the password used to access the keyring file. If NULL is specified, the keyring stash file name is obtained from the `ssl-keyfile-stash` key in the `ssl` stanza in the specified `configfile`.

configfile
The fully qualified path name to a local configuration file. If NULL is specified, the `pd.conf` file is used. The content of this configuration file is used to determine the values for any arguments that were not explicitly specified as input.

Output

ctx The new context that is used to send administration requests to the policy server. Free this object when it is no longer needed.

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Creates a security context for communicating with the Security Access Manager policy server.

The context represents a connection to the Security Access Manager policy server. To successfully create a context, the Security Access Manager policy server must be available and the authentication must be successful.

Command-line equivalent:

```
pdadmin login [-a admin_id [-p password] [-d domain | -m]]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_createdefault2()

This API creates a security context for communicating with the Security Access Manager policy server with the default Secure Sockets Layer (SSL) configuration.

Syntax

```
unsigned long ivadmin_context_createdefault2(
const char *userid,
const char *pwd,
const char *domainid,
ivadmin_context *ctx,
ivadmin_response *rsp
);
```

Parameters

Input

userid The administrator user name for authentication. This user must have the appropriate administrative authority to do the required administrative operations.

pwd The administrator password.

domainid
The name of the domain to which authentication occurs. If this argument is `IVADMIN_DOMAIN_MANAGEMENT`, then the management domain is used. If this argument is `IVADMIN_DOMAIN_LOCAL`, then the local domain is used. These constants are defined in `ivadminapi.h`. If this argument is NULL, then the local domain is used.

Output

ctx The new context that is used to send administration requests to the Security Access Manager policy server. Free this object when it is no longer needed.

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

The context represents a connection to the Security Access Manager policy server. To successfully create a context, the Security Access Manager policy server must be available and the authentication must be successful.

The security context defined by this function treats character data as being encoded in the local code set. Character data supplied later to Security Access Manager with this security context is converted to UTF-8 from the local code set. Data returned is converted from UTF-8 back to the local code set. To have character data treated as UTF-8, create the security context with the `ivadmin_context_create3()` function and specify the appropriate value for the `codeset` parameter.

Command-line equivalent:

```
pdadmin login [-a admin_id [-p password] [-d domain | -m]]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_createlocal()

This API creates a context for local administration operations.

Syntax

```
unsigned long ivadmin_context_createlocal(
const char *userid,
const char *pwd,
const char *domain,
const char *codeset,
ivadmin_context *ctx,
ivadmin_response *rsp
);
```

Parameters

Input

userid This argument is ignored.

pwd This argument is ignored.

domain
This argument is ignored.

codeset

Character codeset. Indicates how the application encodes its character data. Cannot be NULL. The following constants are defined in the `ivadminapi.h` file:

IVADMIN_CODESET_UTF8
Character data is encoded in UTF-8.

IVADMIN_CODESET_LOCAL
Character data is encoded in the local code page

Output

ctx The new context that is used to send administration requests to the policy server. Free this object when it is no longer needed.

rsp The response object. Indicates the success or failure of the function. The empty or contains one or more informational, warning, and error messages. Free this object when it is no longer needed.

Description

Creates a context for local administration operations.

The context represents the authentication required to perform local Security Access Manager administrative tasks. Local tasks are those tasks that do not require communication with the Security Access Manager policy server.

Command-line equivalent:

```
pdadmin login -l
```

The **pdadmin** command always uses the local code page.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_delete()

This API deletes a security context and frees any resources that are associated with the context. Deletion includes any connections with the Security Access Manager policy server.

Syntax

```
unsigned long ivadmin_context_delete(
ivadmin_context ctx,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The security context to delete. This context can be used for communicating with the Security Access Manager policy server or a local context.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the security context and any resources that are associated with that context. This function must be called before exiting the application. Deletes the connection with the Security Access Manager policy server (if one exists) and frees Secure Sockets Layer (SSL) resources. The security context is not usable after this call. After deleting the context, free the context memory with the `ivadmin_free` function.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_domainismanagement()

This API indicates whether the specified context is authenticated to the management domain.

Syntax

```
unsigned long ivadmin_context_domainismanagement(
ivadmin_context ctx
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Description

Indicates whether the specified context is authenticated to the management domain. You must call the `ivadmin_context_create3()` or `ivadmin_context_createdefault2()` function to obtain an `ivadmin_context` object before using this function.

Command-line equivalent:

```
pdadmin context show
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The context is authenticated to the management domain.

IVADMIN_FALSE
Defined as 0. The context is not authenticated to the management domain.

ivadmin_context_getaccexpdate()

This API returns the account expiration date for all user accounts.

Syntax

```
unsigned long ivadmin_context_getaccexpdate(
ivadmin_context ctx,
unsigned long *seconds,
unsigned long *unlimited,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

seconds
The date and time of the expiration of all user accounts to return. This value is the number of seconds since 00:00:00 Universal time, 1 January 1970 (same as `time_t`).

unlimited
The `account-expiration-not-restricted` indicator to return.

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the account expiration date for all user accounts.

Command-line equivalent:

```
pdadmin policy get account-expiry-date
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_getcodeset()

This API returns the character code set that is associated with the specified security context.

Syntax

```
const char* ivadmin_context_getcodeset(
ivadmin_context ctx
);
```

Parameters

Input

ctx A pointer to the context whose code set is returned.

Description

Returns IVADMIN_CODESET_UTF8 if the character data is encoded in UTF-8, or IVADMIN_CODESET_LOCAL if the character data is encoded in the local code page. You must call the ivadmin_context_create3(), ivadmin_context_createdefault2(), or ivadmin_context_createlocal() function to obtain an ivadmin_context object before using this function.

Do not free the character data string that is returned. This data is maintained in the ivadmin_context object.

No command-line equivalent, as the **pdadmin** command always uses the local code page.

Return values

Returns the code set that is in effect for this context. Valid return values are IVADMIN_CODESET_LOCAL and IVADMIN_CODESET_UTF8. These constants are defined in the ivadminapi.h file.

ivadmin_context_getdisabletimeint()

This API returns the time to disable user accounts when the maximum number of login failures is exceeded. This setting applies to all user accounts.

Syntax

```
unsigned long ivadmin_context_getdisabletimeint(
ivadmin_context ctx,
unsigned long *seconds,
unsigned long *disable,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

seconds
The specified number of seconds that the user account is disabled if the maximum number of login failures is exceeded.

disable
The user account that is disabled when the maximum number of login failures is exceeded. Administrator action is required to enable the account.

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the time to disable user accounts if the maximum number of login failures is exceeded. This setting applies to all user accounts.

Command-line equivalent:

```
pdadmin policy get disable-time-interval
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_getdomainid()

This API returns the name of the domain that is associated with the specified context. This domain is the one selected for this context when the context was created.

Syntax

```
const char* ivadmin_context_getdomainid(
ivadmin_context ctx
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Description

Returns the domain name from the specified context object. You must call the `ivadmin_context_create3()` or `ivadmin_context_createdefault2()` function to obtain an `ivadmin_context` object before using this function. Do not free the character string that is returned. This data is maintained in the `ivadmin_context` object.

Command-line equivalent:

```
pdadmin context show
```

Return values

Returns the name of the domain to which the specified context is authenticated.

ivadmin_context_getmaxconcurwebsess()

This API returns the maximum number of concurrent web sessions that are allowed for each user account.

Syntax

```
unsigned long ivadmin_context_getmaxconcurwebsess(
ivadmin_context ctx,
unsigned long *sessions,
unsigned long *displace,
unsigned long *unlimited,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

sessions
A nonzero value that indicates the maximum number of concurrent web sessions.

displace
`IVADMIN_TRUE` indicates that the policy is set for session displacement.

unlimited
`IVADMIN_TRUE` indicates that the policy is set to allow unlimited concurrent sessions.

unset `IVADMIN_TRUE` indicates that the policy is not set; otherwise.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the setting for the maximum number of concurrent web sessions for each user account or `IVADMIN_TRUE` for the `displace`, `unlimited` or `unset` parameter.

This policy applies only to certain components of Security Access Manager. A *Web session is a user session that is maintained by the Security Access Manager systems: Security Access Manager WebSEAL and Security Access Manager Plug-ins for Web Servers.*

Command-line equivalent:

```
pdadmin policy get max-concurrent-web-sessions
```

Where the command returns one of the following values:

- An nonzero number
- `displace`
- `unlimited`
- `unset`

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_getmaxlgnfails()

This API returns the maximum number of login failures that are allowed for each user account.

Syntax

```
unsigned long ivadmin_context_getmaxlgnfails(
ivadmin_context ctx,
unsigned long *failures,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

failures
The maximum number of login failures allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the maximum number of login failures allowed for each user account.

Command-line equivalent:

```
pdadmin policy get max-login-failures
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_getmaxpwdage()

This API returns the maximum password age for all user accounts.

Syntax

```
unsigned long ivadmin_context_getmaxpwdage(
ivadmin_context ctx,
unsigned long *seconds,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

seconds
The returned maximum lifetime, in seconds, before expiration of the password. A value of zero means the password never expires.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the maximum password age for all user accounts.

Command-line equivalent:

```
pdadmin policy get max-password-age
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_getmaxpwdrepchars()

This API returns the maximum number of repeated characters allowed in a password for each user account.

Syntax

```
unsigned long ivadmin_context_getmaxpwdrepchars(
ivadmin_context ctx,
unsigned long *chars,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

chars The maximum number of repeated characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the maximum number of repeated characters allowed in a password for each user account.

Command-line equivalent:

```
pdadmin policy get max-password-repeated-chars
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_getmgmtdomainid()

This API returns the name of the management domain.

Syntax

```
const char* ivadmin_context_getmgmtdomainid(
ivadmin_context ctx
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Description

Returns the name of the management domain from the specified context object. You must call the `ivadmin_context_create3()` or `ivadmin_context_createdefault2()` function to obtain an `ivadmin_context` object before using this function. Do not free the character string that is returned. This data is maintained in the `ivadmin_context` object.

There is no explicit command-line equivalent. To log on to the management domain, use the `-m` flag on **pdadmin** invocation, or use the login subcommand and specify `–m`.

Return values

Returns the name of the management domain.

ivadmin_context_getmgmtsvrhost()

This API returns the name of the host system that runs the policy server with which this context has a communication session.

Syntax

```
const char* ivadmin_context_getmgmtsvrhost(
ivadmin_context ctx
);
```

Parameters

Input

ctx A pointer to the context whose policy server host name is returned.

Description

Returns the policy server host name from the specified context object. You must call the `ivadmin_context_create3()` or `ivadmin_context_createdefault2()` function to obtain an `ivadmin_context` object before using this function. Do not free the character string that is returned. This data is maintained in the `ivadmin_context` object.

There is no explicit command-line equivalent.

Return values

Returns the name of the host system that is running the policy server with which this context has a communication session.

ivadmin_context_getmgmtsvrport()

This API returns the TCP/IP port of the policy server with which this context has a communication session.

Syntax

```
unsigned long ivadmin_context_getmgmtsvrport(
ivadmin_context ctx
);
```

Parameters

Input

ctx A pointer to the context whose policy server port is returned.

Description

Returns the policy server port number from the specified context object. You must call the `ivadmin_context_create3()` or `ivadmin_context_createdefault2()` function to obtain an `ivadmin_context` object before using this function.

There is no explicit command-line equivalent.

Return values

Returns the TCP/IP port of the policy server with which this context has a communication session.

ivadmin_context_getminpwdalphas()

This API returns the minimum number of alphabetic characters that are allowed in a password for each user account.

Syntax

```
unsigned long ivadmin_context_getminpwdalphas(
ivadmin_context ctx,
unsigned long *chars,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

chars The minimum number of alphabetic characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the minimum number of alphabetic characters that are allowed in a password for each user account.

Command-line equivalent:

```
pdadmin policy get min-password-alphas
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_getminpwdnonalphas()

This API returns the minimum number of non-alphabetic characters that are allowed in a password for each user account.

Syntax

```
unsigned long ivadmin_context_getminpwdnonalphas(
ivadmin_context ctx,
unsigned long *chars,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

chars The minimum number of non-alphabetic characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the minimum number of non-alphabetic characters that are allowed in a password for each user account.

Command-line equivalent:

```
pdadmin policy get min-password-non-alphas
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_getminpwdlen()

This API returns the minimum password length that is allowed for all user accounts.

Syntax

```
unsigned long ivadmin_context_getminpwdlen(
ivadmin_context ctx,
unsigned long *length,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

length The minimum password length allowed to be set.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the minimum password length for all user accounts.

Command-line equivalent:

```
pdadmin policy get min-password-length
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_getpwdspaces()

This API returns whether spaces are allowed in passwords for all user accounts.

Syntax

```
unsigned long ivadmin_context_getpwdspaces(
ivadmin_context ctx,
unsigned long *allowed,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

allowed
The indicator of whether spaces are allowed in passwords.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns whether spaces are allowed in passwords for all user accounts.

Command-line equivalent:

```
pdadmin policy get password-spaces
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_gettodaccess()

This API returns the global time-of-day access policy.

Syntax

```
unsigned long ivadmin_context_gettodaccess(
ivadmin_context ctx,
unsigned long *days,
unsigned long *start,
unsigned long *end,
unsigned long *reference,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

days A bitmap of the days for the time-of-day access policy.

start The minutes after midnight for the start of the time range.

end The minutes after midnight for the end of the time range.

reference
The time zone: Either Coordinated Universal Time (UTC) or local.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the global time-of-day access policy.

Command-line equivalent:

```
pdadmin policy get todaccess
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_getuserid()

This API returns the name of the user for which the specified context is authenticated. This user ID is the one specified for this context when the context was created.

Syntax

```
const char* ivadmin_context_getuserid(
ivadmin_context ctx
);
```

Parameters

Input

ctx A pointer to the context whose userid is returned.

Description

Returns the user name from the specified context object. You must call the `ivadmin_context_create3()` or `ivadmin_context_createdefault2()` function to obtain an `ivadmin_context` object before using this function. Do not free the character string that is returned. This data is maintained in the `ivadmin_context` object.

Command-line equivalent:

```
pdadmin context show
```

Return values

Returns the name of the user for which the specified context is authenticated.

ivadmin_context_getuserreg()

This API returns the type of user registry that is configured for the policy server.

Syntax

```
unsigned long ivadmin_context_getuserreg(
ivadmin_context ctx,
unsigned long *registry,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

registry
A pointer to a registry type indicator (`IVADMIN_CONTEXT_DCEUSERREG` or `IVADMIN_CONTEXT_LDAPUSERREG`).

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the type of user registry that is configured for this Security Access Manager policy server. The following indicators are defined:

```
#define IVADMIN_CONTEXT_DCEUSERREG 0
#define IVADMIN_CONTEXT_LDAPUSERREG 1
#define IVADMIN_CONTEXT_ADUSERREG 2
#define IVADMIN_CONTEXT_DOMINOUSERREG 3
#define IVADMIN_CONTEXT_MULTIDOMAIN_ADUSERREG 4
```

Command-line equivalent:

```
pdadmin admin show configuration
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_hasdelcred()

This API indicates whether the specified context has a delegated credential set.

Syntax

```
unsigned long ivadmin_context_hasdelcred(
ivadmin_context ctx
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Description

Indicates whether the specified context has a delegated credential associated with it. A delegated credential is associated with a context with the `ivadmin_context_setdelcred()` function and is removed from a context with the `ivadmin_context_cleardelcred()` function.

There is no explicit command-line equivalent.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The context has a delegated credential.

IVADMIN_FALSE
Defined as 0. The context does not have a delegated credential.

ivadmin_context_setaccexpdate()

This API sets the account expiration date for all user accounts.

Syntax

```
unsigned long ivadmin_context_setaccexpdate(
ivadmin_context ctx,
unsigned long seconds,
unsigned long unlimited,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

seconds
The date and time of the expiration of all user accounts. This value is the number of seconds since 00:00:00 Universal time, 1 January 1970 (same as `time_t`).

unlimited
The specified user account is not expired and the *seconds* parameter is ignored if set to true.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the account expiration date for all user accounts.

Command-line equivalent:

```
pdadmin policy set account-expiry-date {unlimited | absolute_time | unset}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_setdelcred()

This API sets the delegated credential for the context based on the specified Privilege Attribute Certificate (PAC).

Syntax

```
unsigned long ivadmin_context_setdelcred(
ivadmin_context ctx,
const unsigned char *pacValue,
const unsigned long pacLength,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

pacValue
The credential PAC data.

pacLength
The credential PAC length.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the delegated credential for the context based on the specified PAC. Only one credential can be delegated at a time. If a delegated credential exists already for this context, it is overwritten.

There is no explicit command-line equivalent.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_setdisabletimeint()

This API sets the number of seconds before disabling each user account when the maximum number of login failures is exceeded.

Syntax

```
unsigned long ivadmin_context_setdisabletimeint(
ivadmin_context ctx,
unsigned long seconds,
unsigned long disable,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

seconds
The specified number of seconds that the user account is disabled if the maximum number of login failures is exceeded.

disable
The user account is disabled when the maximum number of login failures is exceeded. Administrator action is required to enable the account.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the number of seconds before disabling each user account when the maximum number of login failures is exceeded.

Command-line equivalent:

```
pdadmin policy set disable-time-interval {number | unset | disable}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_setmaxconcurwebsess()

This API sets the maximum number of concurrent web sessions that are allowed for each user account.

Syntax

```
unsigned long ivadmin_context_setmaxconcurwebsess(
ivadmin_context ctx,
unsigned long sessions,
unsigned long displace,
unsigned long unlimited,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

displace
The policy is to be set for session displacement if set to true. If set to false, the policy is set as specified. When set to true, this parameter overrides the `sessions` and `unlimited` parameters. This setting is overridden when the `unset` parameter is set to true.

Required value is `IVADMIN_TRUE` or `IVADMIN_FALSE`.

sessions
The maximum number of concurrent Web sessions that are allowed. This parameter is overridden when one of the following parameters is set to true: `unset`, `displace`, or `unlimited`. When specified, the value cannot be `0`.

unlimited
The policy is set to allow unlimited concurrent sessions if set to true. If set to false, the policy is set as specified. When set to true, this parameter overrides the `sessions` parameters. This parameter is overridden when the `unset` or `displace` parameter is set to true.

Required value is `IVADMIN_TRUE` or `IVADMIN_FALSE`.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified. When set to true this parameter overrides the `sessions`, `displace` and `unlimited` parameters.

Required value is `IVADMIN_TRUE` or `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the maximum number of concurrent web sessions that are allowed for each user account.

Sets concurrent web session policy to be limited to a maximum number of concurrent sessions. The setting can allow unlimited concurrent sessions or can be set for session displacement. When set to `displace`, a new web session for a user terminates any existing web session for that user. When set to *number*, the number represents the maximum number of web sessions that can be established. The number cannot be zero.

This policy applies only to certain components of Security Access Manager. A *web session is a user session that is maintained by the Security Access Manager systems: Security Access Manager WebSEAL and Security Access Manager Plug-ins for Web Servers.*

Command-line equivalent:

```
pdadmin policy set max-concurrent-web-sessions { number | displace | unlimited
|unset }
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_setmaxlgnfails()

This API sets the maximum number of login failures that are allowed for each user account.

Syntax

```
unsigned long ivadmin_context_setmaxlgnfails(
ivadmin_context ctx,
unsigned long failures,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

failures
The maximum number of login failures allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the maximum number of login failures that are allowed for each user account.

Command-line equivalent:

```
pdadmin policy set max-login-failures {number | unset}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_setmaxpwdage()

This API sets the maximum password age, in seconds, for all user accounts.

Syntax

```
unsigned long ivadmin_context_setmaxpwdage(
ivadmin_context ctx,
unsigned long seconds,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

seconds
The maximum lifetime, in seconds, before expiration of a password. Setting this value to zero means the password never expires.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the maximum password age, in seconds, for all user accounts.

Command-line equivalent:

```
pdadmin policy set max-password-age {unset | relative_time}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_setmaxpwdrepchars()

This API sets the maximum number of repeated characters that are allowed in a password for each user account.

Syntax

```
unsigned long ivadmin_context_setmaxpwdrepchars(
ivadmin_context ctx,
unsigned long chars,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

chars The maximum number of repeated characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the maximum number of repeated characters that are allowed in a password for each user account.

Command-line equivalent:

```
pdadmin policy set max-password-repeated-chars {number | unset}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_setminpwdalphas()

This API sets the minimum number of alphabetic characters that are allowed in a password for each user account.

Syntax

```
unsigned long ivadmin_context_setminpwdalphas(
ivadmin_context ctx,
unsigned long chars,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

chars The minimum number of alphabetic characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the minimum number of alphabetic characters that are allowed in a password for each user account.

Command-line equivalent:

```
pdadmin policy set min-password-alphas {unset | number}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_setminpwdnonalphas()

This API sets the minimum number of non-alphabetic characters that are allowed in a password for each user account.

Syntax

```
unsigned long ivadmin_context_setminpwdnonalphas(
ivadmin_context ctx,
unsigned long chars,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

chars The minimum number of non-alphabetic characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the minimum number of non-alphabetic characters that are allowed in a password for each user account.

Command-line equivalent:

```
pdadmin policy set min-password-non-alphas {number | unset}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_setminpwdlen()

This API sets the minimum password length for each user account.

Syntax

```
unsigned long ivadmin_context_setminpwdlen(
ivadmin_context ctx,
unsigned long length,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

length The minimum password length allowed to be set.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the minimum password length for each user account.

Command-line equivalent:

```
pdadmin policy set min-password-length {unset | number}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_setpwdspaces()

This API specifies whether spaces are allowed in passwords for all user accounts.

Syntax

```
unsigned long ivadmin_context_setpwdspaces(
ivadmin_context ctx,
unsigned long allowed,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

allowed
A setting that indicates whether spaces are allowed in passwords.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Indicates whether spaces are allowed in passwords for all user accounts.

The initial value of `setpwdspaces` is `unset`.

Command-line equivalent:

```
pdadmin policy set password-spaces {yes | no | unset}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_context_settodaccess()

This API sets the global time-of-day access policy.

Syntax

```
unsigned long ivadmin_context_settodaccess(
ivadmin_context ctx,
unsigned long days,
unsigned long start,
unsigned long end,
unsigned long reference,
```

```
unsigned long unset,
ivadmin_response *rsp
);
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

days A bitmap of the days for the time-of-day policy.

start The minutes after midnight for the start of the time range.

end The minutes after midnight for the end of the time range.

reference
The time zone: Coordinated Universal Time (UTC) or local.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Note: When setting a password policy, you provide a list of days, start time, and end time. The start time and end time apply to each day on the list. If the specified start time is later than the specified end time, then the access is allowed until the specified end time is reached on the next day.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the global time-of-day access policy.

Command-line equivalent:

```
pdadmin policy set todaccess todaccess_string
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_domain_create()

This API creates a domain.

Syntax

```
unsigned long ivadmin_domain_create(ivadmin_context ctx,
const char *domainid,
const char *adminid,
```

```
const char *password,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server. This context must be authenticated to the management domain.

domainid
The name of the domain to create.

adminid
The name of the domain administrator. Cannot be NULL or an empty string.

password
The initial password for the specified *adminid account in the new domain.*

description
The description of the new domain. Cannot be NULL.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Creates a domain with the specified description. This domain might be created earlier and then deleted without deleting the associated user and group data from the user registry. In this case, those users and groups will be available automatically after this API is started. In that case, a special response message is returned to indicate that is the case.

This command applies to LDAP registries only.

Command-line equivalent:

```
pdadmin domain create domain_name domain_admin domain_admin_password
 [-desc description]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_domain_delete()

This API deletes a domain.

Syntax

```
unsigned long ivadmin_domain_delete(ivadmin_context ctx,
const char *domainid,
unsigned long registry,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server. This context must be authenticated to the management domain.

domainid
The name of the domain to delete.

registry
The indicator of whether to delete the user and group data of the domain from the user registry and also delete the user and group data from Security Access Manager. Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`. The user and group data of the domain might not be deleted from the user registry. In this case, those users and groups are available automatically if the domain is re-created.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Deletes a domain. If the registry argument is `IVADMIN_TRUE`, the user and group data of the domain is removed from the user registry. A user and group data of the domain might not be deleted from the user registry. In this case, those users and group data are available automatically if the domain is created again.

This command applies to LDAP registries only.

Command-line equivalent:

```
pdadmin domain delete domain_name [-registry]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_domain_get()

This API returns the specified domain object.

Syntax

```
unsigned long ivadmin_domain_get(
ivadmin_context ctx,
const char *domainid,
ivadmin_domain *domain,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server. This context must be authenticated to the management domain.

domainid
The name of the domain to return.

Output

domain
The domain object. Free this object when it is no longer needed.

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Returns the specified domain object.

This command applies to LDAP registries only.

Command-line equivalent:

```
pdadmin domain show domain_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_domain_getdescription()

This API returns the description for the specified domain object.

Syntax

```
const char* ivadmin_domain_getdescription(
ivadmin_domain domain
);
```

Parameters

Input

domain
A pointer to the domain object.

Description

Returns the description for the specified domain object. You must call the `ivadmin_domain_get()` function to obtain an `ivadmin_domain` object before using this function. Do not free the character string that is returned. This data is maintained in the `ivadmin_domain` object.

Command-line equivalent:

```
pdadmin domain show domain_name
```

The description is part of the information returned by the **pdadmin** command.

Return values

Returns the description that is associated with the domain, or an empty string if the domain has no description.

ivadmin_domain_getid()

This API returns the name of the specified domain.

Syntax

```
const char* ivadmin_domain_getid(
ivadmin_domain domain
);
```

Parameters

Input

domain
A pointer to the domain object.

Description

Returns the name of the specified domain object. You must call the `ivadmin_domain_get()` function to obtain an `ivadmin_domain` object before using this function. Do not free the character string that is returned. This data is maintained in the `ivadmin_domain` object.

Command-line equivalent:

```
pdadmin domain show domain_name
```

The domain name is part of the information returned by the **pdadmin** command.

Return values

Returns the name of the specified domain.

ivadmin_domain_list()

This API lists the names of all the domains, except the management domain.

Syntax

```
unsigned long ivadmin_domain_list(
ivadmin_context ctx,
unsigned long *count,
char ***domainids,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server. This context must be authenticated to the management domain.

Output

count The number of domain identifiers returned. The number can be zero if no domains other than the management domain exist.

domainids
An array of pointers to domain names. Free each domain name and the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Lists the names of all the domains, except the management domain.

This command applies to LDAP registries only.

Command-line equivalent:

```
pdadmin domain list
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_domain_setdescription()

This API creates or changes the description for the specified domain.

Syntax

```
unsigned long ivadmin_domain_setdescription(
ivadmin_context ctx,
const char *domainid,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server. This context must be authenticated to the management domain.

domainid
The name of the domain. Cannot be NULL or an empty string.

description
The new description. Cannot be NULL.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Creates or changes the description for the specified domain.

This command applies to LDAP registries only.

Command-line equivalent:

```
pdadmin domain modify domain_name description description
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_free()

This API frees the memory that is allocated to the specified object.

Syntax

```
void ivadmin_free(
void p*
);
```

Parameters

Input

p A pointer to the object to be freed.

Description

Frees the memory that is allocated to the specified object. Use this function to free all memory that is allocated by the administration API functions.

There is no command-line equivalent for this function.

ivadmin_user_getlastpwdchange()

Returns a value that indicates the date and time of the user's last password change.

Syntax

```
unsigned long ivadmin_user_getlastpwdchange(ivadmin_ldapuser user
);
```

Parameters

Input

user Pointer to the user structure.

Description

Returns an attribute that indicates the date and time that the user last changed the password.

Command-line equivalent:

```
pdadmin user show user_name
```

The last password change status for the user is part of the information returned by the **pdadmin** command.

This value is only available if the following option is configured:

```
ivmgrd
provide-last-pwd-change = yes
```

Return values

Returns the following values:

- 0 is returned if the user is NULL.
- A long integer value that indicates the data and time of the user's last password change.

ivadmin_group_addmembers()

This API adds the specified users to the specified group.

Syntax

```
unsigned long ivadmin_group_addmembers(
ivadmin_context ctx,
const char *groupid,
unsigned long user_count,
const char **users,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

groupid
The group name.

user_count
The number of users to add to the group.

users The new member user names.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Adds the specified users to the specified group. Security Access Manager does not support a group as a group member.

Command-line equivalent:

```
pdadmin group modify group_name add user

pdadmin group modify group_name add (user1 user_2 ... user_n)
```

User registry difference: Attempting to add a duplicate user to a group is handled differently depending on what user registry is being used. See Appendix B, "User registry differences," on page 293 for details.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_group_create2()

This API creates a group.

Syntax

```
unsigned long ivadmin_group_create2(
ivadmin_context ctx,
const char *groupid,
const char *dn,
const char *cn,
const char *group_container,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

groupid
The group name.

dn The user registry distinguished name.

cn The common name attribute of the user registry.

group_container
The container object within the `/Management` object space. Can be NULL to indicate that it is at the root level.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates a Security Access Manager group in the user registry with the specified name, distinguished name, and common name.

Groups created in the Active Directory Lightweight Directory Service (AD LDS) user registry must be created in the same AD LDS partition where the Security Access Manager Management Domain information is stored.

User registry difference: Leading and trailing blanks in a group name do not make the name unique when using an LDAP or Active Directory user registry. To keep name processing consistent regardless of what user registry is being used, do not define group names with leading or trailing blanks.

Command-line equivalent:

```
pdadmin group create group_name dn cn
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_group_delete2()

This API deletes the specified group.

Syntax

```
unsigned long ivadmin_group_delete2(
ivadmin_context ctx,
const char *groupid,
unsigned long registry,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

groupid
The group name.

registry
The indicator of whether to delete the group from the user registry and also deleting all information about the group from Security Access Manager.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified group. Deletes all Security Access Manager information about the group and optionally deletes the user registry contents.

Command-line equivalent:

```
pdadmin group delete [–registry] group_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_group_get()

This API returns the specified group object.

Syntax

```
unsigned long ivadmin_group_get(
ivadmin_context ctx,
const char *groupid,
ivadmin_ldapgroup *group,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

groupid
The group name.

Output

group The returned group. Free the memory for this `ivadmin_ldapgroup` object when it is no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the group object for the specified group name. Free the memory for this `ivadmin_ldapgroup` object when it is no longer needed.

Command-line equivalent:

```
pdadmin group show group-name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_group_getbydn()

This API returns a group user with the user registry distinguished name for identification.

Syntax

```
unsigned long ivadmin_group_getbydn(
ivadmin_context ctx,
const char *dn,
ivadmin_ldapgroup *group,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

dn The distinguished name of group in the user registry.

Output

group The returned group. Free this memory when it is no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns a group user with the user registry DN for identification. Free the memory for this `ivadmin_ldapgroup` object when it is no longer needed.

User registry difference: The maximum length of the distinguished name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

Command-line equivalent:

```
pdadmin group show-dn dn
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_group_getcn()

This API returns the user registry common name attribute for the specified group.

Syntax

```
const char* ivadmin_group_getcn(
ivadmin_ldapgroup group
);
```

Parameters

Input

group A pointer to the group structure.

Description

Returns the user registry common name attribute for the specified group object.

Do not free this memory. This data is maintained in the `ivadmin_ldapgroup` structure.

User registry difference: The maximum length of the common name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

Command-line equivalent:

```
pdadmin group show group-name
```

The user registry common name is part of the information returned by the `pdadmin group show` command.

Return values

Returns the user registry common name attribute for the specified group.

ivadmin_group_getdescription()

This API returns the user registry description for the specified group.

Syntax

```
const char* ivadmin_group_getdescription(
ivadmin_ldapgroup group
);
```

Parameters

Input

group
A pointer to the group structure.

Description

Returns the user registry description for the specified group.

Do not free this memory. This data is maintained in the `ivadmin_ldapgroup` structure.

Command-line equivalent:

```
pdadmin group show group-name
```

The description is part of the information returned by the `pdadmin group show` command.

Return values

Returns the user registry description for the specified group. The maximum length of a description is 1024 characters.

ivadmin_group_getdn()

This API returns the user registry distinguished name for the specified group.

Syntax

```
const char* ivadmin_group_getdn(
ivadmin_ldapgroup group
);
```

Parameters

Input

group A pointer to the group structure.

Description

Returns the user registry distinguished name for the specified group.

Do not free this memory. This data is maintained in the `ivadmin_ldapgroup` structure.

User registry difference: The maximum length of the distinguished name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

Command-line equivalent:

```
pdadmin group show group-name
```

The user registry distinguished name is part of the information returned by the `pdadmin group show` command.

Return values

Returns the user registry distinguished name for the specified group.

ivadmin_group_getid()

This API returns the group name from the specified group object.

Syntax

```
const char* ivadmin_group_getid(
ivadmin_ldapgroup group
);
```

Parameters

Input

group A pointer to the group structure.

Description

Returns the group name from the specified group object.

Do not free this memory. This data is maintained in the `ivadmin_ldapgroup` structure.

Command-line equivalent:

```
pdadmin group show group-name
```

The group name is part of the information that is returned by the `pdadmin group show` command.

Return values

Returns the group name from the specified group object. The maximum length of a group name is 256 characters.

ivadmin_group_getmembers()

This API lists the user names of the members of the specified group.

Syntax

```
unsigned long ivadmin_group_getmembers(
ivadmin_context ctx,
const char *groupid,
unsigned long *count,
char ***userids,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

groupid
The group name.

Output

count The number of user names returned.

userids
An array of pointers to the user names returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists the user names of the members of the specified group.

Command-line equivalent:

```
pdadmin group show-members group_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_group_import2()

This API creates a Security Access Manager group by importing a group that exists in the user registry.

Syntax

```
unsigned long ivadmin_group_import2(
ivadmin_context ctx,
const char *groupid,
const char *dn,
const char *group_container,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

groupid
The group name.

dn The user registry distinguished name.

group_container
The container object within the `/Management` object space. Can be NULL to indicate that it is at the root level.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates a Security Access Manager group by importing a group that exists in the user registry.

Groups created in the Active Directory Lightweight Directory Service (AD LDS) user registry must be created in the same AD LDS partition where the Access Manager Management Domain information is stored.

Command-line equivalent:

```
pdadmin group import group_name dn
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_group_list()

This API lists the names of the Security Access Manager groups.

Syntax

```
unsigned long ivadmin_group_list(
ivadmin_context ctx,
const char *pattern,
unsigned long maxreturn,
unsigned long *count,
char ***groupids,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

pattern
The pattern match for group names. `IVADMIN_ALLPATTERN` indicates all groups.

maxreturn
The maximum number to return. `IVADMIN_MAXRETURN` indicates unlimited. This number can also be limited by the user registry server so the maximum returned is really the minimum of the server configuration and this value.

Output

count The number of group names returned.

groupids
An array of pointers to the group names returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists the names of the Security Access Manager groups. Returns a list of group names whose name matches the pattern specified.

The order returned is the order created.

Command-line equivalent:

```
pdadmin group list pattern max_return
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_group_listbydn()

This API returns a list of user registry distinguished names whose user registry common name attribute matches the pattern specified.

Syntax

```
unsigned long ivadmin_group_listbydn(
ivadmin_context ctx,
const char *pattern,
unsigned long maxreturn,
unsigned long *count,
char ***dns,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

pattern
The pattern match for common name attribute. `IVADMIN_ALLPATTERN` indicates all users.

maxreturn
The maximum number to return. `IVADMIN_MAXRETURN` indicates unlimited. This number can also be limited by the user registry server. The limit specifies that the maximum returned is really the minimum of the server configuration and this value.

Output

count The number of user registry distinguished names returned.

dns An array of pointers to the user registry distinguished names returned. You

must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns a list of user registry distinguished names whose user registry common name attributes match the pattern specified.

User registry difference: The maximum length of the distinguished name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

Command-line equivalent:

```
pdadmin group list-dn pattern max_return
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_group_removemembers()

This API removes the specified user or users from the specified group.

Syntax

```
unsigned long ivadmin_group_removemembers(
ivadmin_context ctx,
const char *groupid,
unsigned long user_count,
const char **users,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

groupid
The group name.

user_count
The number of user names to remove.

users The member user names to remove.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Removes the specified user or users from the specified group.

Command-line equivalent:

```
pdadmin group modify group_name remove user_name

pdadmin group modify group_name remove (user_1 user_2 ... user_n)
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_group_setdescription()

This API creates or changes the description for the specified group.

Syntax

```
unsigned long ivadmin_group_setdescription(
ivadmin_context ctx,
const char *groupid,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

groupid
The group name.

description
The new description.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the description for the specified group.

Command-line equivalent:

```
pdadmin group modify group_name description description
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_objectspace_create()

This API creates a Security Access Manager protected object space.

Syntax

```
unsigned long ivadmin_objectspace_create(
ivadmin_context ctx,
const char *objspaceid,
unsigned long type,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objspaceid
The name of the object space to create.

type The type of object space to create.

description
A description for the object space.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates a Security Access Manager protected object space.

You must specify the object type for each new object by with the *type* input parameter. The object type is used by Web Portal Manager to display an appropriate icon with the object.

Note: The root of the new protected object has the `ispolicyattachable` attribute set to true. See "ivadmin_protobj_setpolicyattachable()" on page 223.

The supported object types are in Table 33.

Table 33. Supported object types

Variable name	Value	Description
IVADMIN_PROTOBJ_TYPE_UNKNOWN	0	Unknown
IVADMIN_PROTOBJ_TYPE_DOMAIN	1	Secure domain
IVADMIN_PROTOBJ_TYPE_FILE	2	File
IVADMIN_PROTOBJ_TYPE_PROGRAM	3	Executable program
IVADMIN_PROTOBJ_TYPE_DIR	4	Directory

Table 33. Supported object types (continued)

Variable name	Value	Description
IVADMIN_PROTOBJ_TYPE_JNCT	5	Junction
IVADMIN_PROTOBJ_TYPE_WEBSEAL_SVR	6	WebSEAL server
IVADMIN_PROTOBJ_TYPE_NETSEAL_SVR	7	Unused
IVADMIN_PROTOBJ_TYPE_EXTERN_AUTH_SVR	8	Unused
IVADMIN_PROTOBJ_TYPE_HTTP_SVR	9	Unused
IVADMIN_PROTOBJ_TYPE_NON_EXIST_OBJ	10	Nonexistent object
IVADMIN_PROTOBJ_TYPE_CONTAINER	11	Container object
IVADMIN_PROTOBJ_TYPE_LEAF	12	Leaf object
IVADMIN_PROTOBJ_TYPE_PORT	13	Port
IVADMIN_PROTOBJ_TYPE_APP_CONTAINER	14	Application container object
IVADMIN_PROTOBJ_TYPE_APP_LEAF	15	Application leaf object
IVADMIN_PROTOBJ_TYPE_MGMT_OBJ	16	Management object
IVADMIN_PROTOBJ_TYPE_NETSEAL_NET	17	Unused

Command-line equivalent:

```
pdadmin objectspace create objectspace_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_objectspace_delete()

This API deletes the specified Security Access Manager protected object space.

Syntax

```
unsigned long ivadmin_objectspace_delete(
ivadmin_context ctx,
const char *objspaceid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objspaceid
The name of the object space to delete.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified Security Access Manager protected object space.

Command-line equivalent:

```
pdadmin objectspace delete objectspace_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_objectspace_list()

This API lists the names of all the Security Access Manager protected object spaces.

Syntax

```
unsigned long ivadmin_objectspace_list(
ivadmin_context ctx,
unsigned long *count,
char ***objspace_list,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

count The number of object space names returned.

objspace_list
An array of pointers to the names of the object spaces returned. You must free the character data referenced by each pointer as well as the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists the names of all the Security Access Manager protected object spaces.

Command-line equivalent:

```
pdadmin objectspace list
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_attach()

This API attaches a protected object policy to the specified protected object.

Syntax

```
unsigned long ivadmin_pop_attach(
ivadmin_context ctx,
char *popid,
char *objid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy to attach.

objid The name of the protected object.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Attaches a protected object policy to the specified protected object. Be sure that the protected object exists in the protect object space before attempting to attach a POP.

Command-line equivalent:

```
pdadmin attach object_name pop_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_attrdelkey()

This API deletes the specified extended attribute from the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_attrdelkey(
ivadmin_context ctx,
char *popid,
char *attr_key,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

attr_key
The extended attribute to delete.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified extended attribute from the specified protected object policy.

Command-line equivalent:

```
pdadmin pop modify pop_name delete attribute attribute_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_attrdelval()

This API deletes the specified value from the specified extended attribute key in the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_attrdelval(
ivadmin_context ctx,
char *popid,
char *attr_key,
char *attr_value,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

attr_key
The extended attribute that contains the value to delete.

attr_value
The value to delete from the extended attribute.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified value from the specified extended attribute key in the specified protected object policy.

Command-line equivalent:

```
pdadmin pop modify pop_name delete attribute attribute_name attribute_value
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_attrget()

This API returns the values for the specified extended attribute from the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_attrget(
ivadmin_pop pop,
char *attr_key,
unsigned long *count,
char ***attr_value
);
```

Parameters

Input

pop The protected object policy to access.

attr_key
The extended attribute to get.

Output

count The number of values returned.

attr_value
An array of pointers to the extended attribute values returned. You must free the character data referenced by each pointer as well as the array of pointers when they are no longer needed.

Description

Returns the values for the specified extended attribute from the specified protected object policy. The value returned is in the same format as when it was created with the `ivadmin_pop_attrput()` function. If an error occurs, NULL is returned.

Command-line equivalent:

```
pdadmin pop show pop_name attribute
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_attrlist()

This API lists the extended attributes that are associated with the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_attrlist(
ivadmin_pop pop,
unsigned long *count,
char ***attr_list
);
```

Parameters

Input

pop The protected object policy.

Output

count The number of extended attributes returned.

attr_list
An array of pointers to the extended attributes returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

Description

Lists the extended attributes that are associated with the specified protected object policy. If an error occurs, `NULL` is returned.

Command-line equivalent:

```
pdadmin pop list pop_name attribute
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_attrput()

This API creates or modifies the value for the specified extended attribute in the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_attrput(
ivadmin_context ctx,
char *popid,
char *attr_key,
char *attr_value,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

attr_key
The extended attribute for which a value must be set.

attr_value
The value to set.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or modifies the value for the specified extended attribute in the specified protected object policy.

Command-line equivalent:

```
pdadmin modify pop_name set attribute attribute_name attribute_value
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_create()

This API creates a protected object policy object.

Syntax

```
unsigned long ivadmin_pop_create(
ivadmin_context ctx,
const char *popid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy to create.

The value for `popid` can contain alphabets, numbers, and symbols such as '+' and '&'. The forward slash symbol (/) is not permitted.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates a protected object policy object with the default values described in Table 34.

Table 34. Protected object policy default values

Attribute Name	Default Value
Description	none
Warning mode	no
Audit level	none
Quality of protection	none
Time-of-day access	sun, mon, tue, wed, thu, fri, sat:anytime:local
IP endpoint authentication method policy	0
Any other network	0

For more information about creating POPs, see the section about creating and deleting protected object policies in the *IBM Security Access Manager for Web: Administration Guide*.

Command-line equivalent:

```
pdadmin pop create pop_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_delete()

This API deletes the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_delete(
ivadmin_context ctx,
const char *popid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy to delete.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified protected object policy.

Command-line equivalent:

```
pdadmin pop delete pop_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_detach()

This API detaches a protected object policy from the specified protected object.

Syntax

```
unsigned long ivadmin_pop_detach(
ivadmin_context ctx,
char *objid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object to detach from.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Detaches a protected object policy from the specified protected object.

Command-line equivalent:

```
pdadmin pop detach pop_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_find()

This API finds and lists all names of all protected objects that have the specified protected object policy attached.

Syntax

```
unsigned long ivadmin_pop_find(
ivadmin_context ctx,
char *popid,
unsigned long *count,
char ***obj_list,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy to find.

Output

count The number of protected objects returned.

obj_list
An array of pointers to the protected objects returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Finds and lists all protected objects that have the specified protected object policy attached.

Command-line equivalent:

```
pdadmin pop find pop_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_get()

This API returns the specified protected object policy object.

Syntax

```
unsigned long ivadmin_pop_get(
ivadmin_context ctx,
char *popid,
ivadmin_pop *pop,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy to get.

Output

pop The protected object policy that is returned.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the specified protected object policy object. Call this function to get an object of type `ivadmin_pop`.

You must free the `ivadmin_pop` object when it is no longer needed.

Command-line equivalent:

```
pdadmin pop show pop_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_getanyothernw2()

This API returns the setting for any other network (**anyothernw**) for the IP authentication level from the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_getanyothernw2(
  ivadmin_context ctx,
 const char *popid,
 unsigned long *level,
 ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy to get.

Output

level The returned authentication level that is associated with **anyothernw**.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the setting for any other network (**anyothernw**) for the authentication level from the specified protected object policy. If the authentication setting is `forbidden`, then the constant `IVADMIN_IPAUTH_FORBIDDEN` is returned.

Command-line equivalent:

```
pdadmin pop show pop_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_getauditlevel()

This API returns the audit level for the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_getauditlevel(
ivadmin_pop pop
);
```

Parameters

Input

pop The protected object policy.

Description

Returns the audit level for the specified protected object policy.

Command-line equivalent:

```
pdadmin show pop_name
```

The audit level is part of the information returned by the **pdadmin** command.

Return values

Returns the audit level. The audit level is specified as an unsigned long. The following audit levels are defined:

```
#define IVADMIN_AUDIT_NONE (0)
#define IVADMIN_AUDIT_PERMIT (1)
#define IVADMIN_AUDIT_DENY (2)
#define IVADMIN_AUDIT_ERROR (4)
#define IVADMIN_AUDIT_ADMIN (8)
#define IVADMIN_AUDIT_ALL (15)
```

Descriptions for the audit levels can be found in Table 35.

Table 35. Descriptions of audit levels

Audit Value	Description
none	Auditing is disabled.
permit	Audit all requests on a protected object that result in successful access.
deny	Audit all requests on a protected object that result in denial of access.
error	Audit all internally generated error messages when access to the protected object is denied.
admin	Not implemented.
all	Audit success, error, and failure for all events.

ivadmin_pop_getdescription()

This API returns the description of the specified protected object policy.

Syntax

```
const char* ivadmin_pop_getdescription(
ivadmin_pop pop
);
```

Parameters

Input

pop The protected object policy.

Description

Returns the description of the specified protected object policy. You must call `ivadmin_pop_get()` to obtain an `ivadmin_pop` object before calling this function.

Do not free this description. This data is maintained in the `ivadmin_pop` structure.

Command-line equivalent:

```
pdadmin show pop_name
```

The description is part of the information returned by the **pdadmin** command.

Return values

Returns the description of the specified protected object policy. There is no limit to the length of the description.

ivadmin_pop_getid()

This API returns the name of the specified protected object policy.

Syntax

```
const char* ivadmin_pop_getid(
ivadmin_pop pop
);
```

Parameters

Input

pop The protected object policy.

Description

Returns the name of the specified protected object policy. You must call `ivadmin_pop_get()` to obtain an `ivadmin_pop` object before calling this function.

Do not free this name. This data is maintained in the `ivadmin_pop` structure.

Command-line equivalent:

```
pdadmin show pop_name
```

The name is part of the information returned by the **pdadmin** command.

Return values

Returns the name of the specified protected object policy. There is no limit to the name of the policy.

ivadmin_pop_getipauth3()

This API returns the specified IP endpoint authentication settings in the specified protected object policy (POP).

Syntax

```
unsigned long ivadmin_pop_getipauth3(
ivadmin_context ctx,
const char *popid,
unsigned long *count,
unsigned char ***networks,
unsigned char ***netmasks,
unsigned long **levels,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

Output

count The number of settings retrieved.

networks
An array of network address.

netmask
Any array of netmask address.

levels An array of authentication levels that are associated with the POP.

Note: You can specify any integer value, except `1000`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

If the network is not already protected, adds the specified IP endpoint authentication settings in the specified protected object policy.

When specifying the *network* and *netmask* parameters, the C client code is dependent on the operating system-provided functions. Be aware of the following restrictions:

- IPv4 clients must provide addresses in IPv4 format even with IPv6 servers.
- IPv6 clients can provide addresses in IPv4 or IPv6 format to IPv6 servers.

For an IPv6 address to be accepted, the server must be IPv6. You cannot provide an IPv6 address to an IPv4 server.

Command-line equivalent:

`pdadmin pop modify pop_name set ipauth add network netmask level`

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_getqop()

This API returns the quality of protection level for the specified protected object policy.

Syntax

```
const char* ivadmin_pop_getqop(
ivadmin_pop pop
);
```

Parameters

Input

pop The protected object policy.

Description

Returns the quality of protection level for the specified protected object policy.

Do not free this quality of protection level. This data is maintained in the `ivadmin_pop` structure.

Command-line equivalent:

```
pdadmin show pop_name
```

The quality of protection level is part of the information returned by the **pdadmin** command.

Return values

Returns the quality of protection level for the specified protected object policy. The following levels are defined:

- `none`
- `integrity`
- `privacy`

ivadmin_pop_gettod()

This API Returns the time-of-day range for the specified protected object policy.

Syntax

```
unsigned long
ivadmin_pop_gettod(
ivadmin_pop pop,
unsigned long *days,
unsigned long *start,
unsigned long *end,
unsigned long *reference
);
```

Parameters

Input

pop The protected object policy.

Output

days A bitmap of the days.

start The minutes for the start of the range.

end The minutes for the end of the range.

reference
The time reference; either Universal Time Coordinated (UTC) or local.

Description

Returns the time-of-day range for the specified protected object policy.

Command-line equivalent:

```
pdadmin show pop_name
```

The time-of-day range is part of the information returned by the **pdadmin** command.

The following values are defined for the time-of-day settings:

```
#define IVADMIN_TIME_LOCAL (0)
#define IVADMIN_TIME_UTC (1)
#define IVADMIN_TOD_ANY (0)
#define IVADMIN_TOD_SUN (1)
#define IVADMIN_TOD_MON (2)
#define IVADMIN_TOD_TUE (4)
#define IVADMIN_TOD_WED (8)
#define IVADMIN_TOD_THU (16)
#define IVADMIN_TOD_FRI (32)
#define IVADMIN_TOD_SAT (64)
#define IVADMIN_TOD_ALL (127)
#define IVADMIN_TOD_WEEKDAY (62)
#define IVADMIN_TOD_WEEKEND (65)
#define IVADMIN_TOD_MINUTES (60)
#define IVADMIN_TOD_OCLOCK (3600)
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_getwarnmode()

This API returns the warning mode value from the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_getwarnmode(
ivadmin_pop pop
);
```

Parameters

Input

pop The protected object policy.

Description

Returns the warning mode value from the specified protected object policy.

Command-line equivalent:

```
pdadmin show pop_name
```

The warning mode value is part of the information returned by the **pdadmin** command.

Return values

Returns the warning mode set for this protected object policy.

ivadmin_pop_list()

This API lists the names of all protected object policy objects.

Syntax

```
unsigned long ivadmin_pop_list(
ivadmin_context ctx,
unsigned long *count,
char ***poplist,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

count The number of protected object policies returned.

poplist
An array of pointers to the protected object policies returned. You must free the character data referenced by each pointer as well as the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists the names of all protected object policy objects.

Command-line equivalent:

```
pdadmin pop list
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_removeipauth2()

This API removes IP endpoint authentication settings from the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_removeipauth2(
ivadmin_context ctx,
const char *popid,
const unsigned char *network,
const unsigned char *netmask,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

network
The network address to delete.

netmask
The netmask address.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

If present, removes IP endpoint authentication settings from the specified protected object policy that were set or added.

When specifying the *network* and *netmask* parameters, the C client code is dependent on the operating system-provided functions. Be aware of the following restrictions:

- IPv4 clients must provide addresses in IPv4 format even with IPv6 servers.
- IPv6 clients can provide addresses in IPv4 or IPv6 format to IPv6 servers.

For an IPv6 address to be accepted, the server must be IPv6. You cannot provide an IPv6 address to an IPv4 server.

Command-line equivalent:

```
pdadmin pop modify pop_name set ipauth remove network netmask
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_setanyothernw2()

This API modifies the setting for any other network (**anyothernw**) to the specified IP authentication level for the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_setanyothernw2(
ivadmin_context ctx,
const char *popid,
unsigned long level,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

level The authentication level, except for `forbidden`, to associate with **anyothernw**. To set the authentication level to `forbidden`, use the `ivadmin_pop_setanyothernw_forbidden2()` function.

Note: You can specify any integer value, except `1000`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Modifies the setting for any other network (**anyothernw**) for the authentication level from the specified protected object policy (POP). Use the **anyothernw** setting to set the authentication level for all IP addresses and IP address ranges not listed explicitly in the POP.

When specifying IP addresses, the C client code is dependent on the operating system-provided functions. Be aware of the following restrictions:

- IPv4 clients must provide addresses in IPv4 format even with IPv6 servers.
- IPv6 clients can provide addresses in IPv4 or IPv6 format to IPv6 servers.

For an IPv6 address to be accepted, the server must be IPv6. You cannot provide an IPv6 address to an IPv4 server.

Command-line equivalent:

```
pdadmin pop modify pop_name set ipauth anyothernw level
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_setanyothernw_forbidden2()

This API modifies the access setting for any other network (**anyothernw**) to `forbidden` for the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_setanyothernw_forbidden2(
ivadmin_context ctx,
const char *popid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Modifies the access setting for any other network (**anyothernw**) to `forbidden` for the specified protected object policy.

Command-line equivalent:

```
pdadmin pop modify pop_name set ipauth anyothernw forbidden
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_setauditlevel()

This API creates or modifies the audit level for the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_setauditlevel(
ivadmin_context ctx,
char *popid,
unsigned long audit_level,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid
The name of the protected object policy.

audit_level
The new audit level for the protected object policy.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or modifies the audit level for the specified protected object policy.

Command-line equivalent:

```
pdadmin pop modify pop_name set audit-level [all | none | audit_level_list]
```

The audit level is specified as an unsigned long. The following audit levels are defined:

```
#define IVADMIN_AUDIT_NONE    (0)
#define IVADMIN_AUDIT_PERMIT  (1)
#define IVADMIN_AUDIT_DENY    (2)
#define IVADMIN_AUDIT_ERROR   (4)
#define IVADMIN_AUDIT_ADMIN   (8)
#define IVADMIN_AUDIT_ALL    (15)
```

Table 35 on page 181 lists audit levels and their descriptions.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_setdescription()

This API creates or modifies the description of the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_setdescription(
ivadmin_context ctx,
char *popid,
char *desc,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

desc The new description for the protected object policy.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or modifies the description of the specified protected object policy.

Command-line equivalent:

```
pdadmin pop modify pop_name set description description
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_setipauth2()

This API adds the specified IP endpoint authentication setting in the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_setipauth2(
ivadmin_context ctx,
const char *popid,
const unsigned char *network,
const unsigned char *netmask,
unsigned long level,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

network
The network address.

netmask
The netmask address.

level The authentication level, except for `forbidden`, to set for the POP. To set the authentication level to `forbidden`, use the `ivadmin_pop_setipauth_forbidden2()` function.

Note: You can specify any integer value, except `1000`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

If the network is not already protected, adds the specified IP endpoint authentication settings in the specified protected object policy.

When specifying the *network* and *netmask* parameters, the C client code is dependent on the operating system-provided functions. Be aware of the following restrictions:

- IPv4 clients must provide addresses in IPv4 format even with IPv6 servers.
- IPv6 clients can provide addresses in IPv4 or IPv6 format to IPv6 servers.

For an IPv6 address to be accepted, the server must be IPv6. You cannot provide an IPv6 address to an IPv4 server.

Command-line equivalent:

```
pdadmin pop modify pop_name set ipauth add network netmask level
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_setipauth_forbidden2()

This API adds the specified network with an authentication level of `forbidden` to the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_setipauth_forbidden2(
ivadmin_context ctx,
const char *popid,
const unsigned char *network,
const unsigned char *netmask,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

network
The network address.

netmask
The netmask address.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

If the network is not already protected in the specified protected object policy, adds the network with an authentication level of `forbidden`.

When specifying the *network* and *netmask* parameters, the C client code is dependent on the operating system-provided functions. Be aware of the following restrictions:

- IPv4 clients must provide addresses in IPv4 format even with IPv6 servers.
- IPv6 clients can provide addresses in IPv4 or IPv6 format to IPv6 servers.

For an IPv6 address to be accepted, the server must be IPv6. You cannot provide an IPv6 address to an IPv4 server.

Command-line equivalent:

```
pdadmin pop modify pop_name set ipauth add network netmask forbidden
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_setqop()

This API creates or changes the quality of protection level for the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_setqop(
ivadmin_context ctx,
char *popid,
char *qop_level,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid
The name of the protected object policy.

qop_level
The new quality of protection level to set. The following string values are supported:
- `none`
- `integrity`
- `privacy`

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the quality of protection level for the specified protected object policy. The following string values are supported:
- `none`
- `integrity`
- `privacy`

Command-line equivalent:

```
pdadmin pop modify pop_name set qop [none|integrity|privacy]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_settod()

This API creates or changes the time-of-day policy for the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_settod(
ivadmin_context ctx,
char *popid,
unsigned long days,
unsigned long start,
unsigned long end,
unsigned long reference,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

days A bitmap of the days.

start The minutes for the start of the range.

end The minutes for the end of the range.

reference
The time zone: Universal Time Coordinated (UTC) or local.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the time-of-day policy for the specified protected object policy.

Command-line equivalent:

```
pdadmin pop modify pop_name set tod-access time_of_day_string
```

The following values are defined for the time-of-day settings:

```
#define IVADMIN_TIME_LOCAL (0)
#define IVADMIN_TIME_UTC (1)
#define IVADMIN_TOD_ANY (0)
#define IVADMIN_TOD_SUN (1)
#define IVADMIN_TOD_MON (2)
#define IVADMIN_TOD_TUE (4)
#define IVADMIN_TOD_WED (8)
#define IVADMIN_TOD_THU (16)
#define IVADMIN_TOD_FRI (32)
#define IVADMIN_TOD_SAT (64)
#define IVADMIN_TOD_ALL (127)
#define IVADMIN_TOD_WEEKDAY (62)
#define IVADMIN_TOD_WEEKEND (65)
#define IVADMIN_TOD_MINUTES (60)
#define IVADMIN_TOD_OCLOCK (3600)
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_pop_setwarnmode()

This API creates or changes the warning mode for the specified protected object policy.

Syntax

```
unsigned long ivadmin_pop_setwarnmode(
ivadmin_context ctx,
char *popid,
unsigned long warn_mode,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object policy.

warn_mode
The new value of the warning mode. Supported values are `IVADMIN_TRUE (1)` or `IVADMIN_FALSE (0)`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the warning mode for the specified protected object policy.

Command-line equivalent:

```
pdadmin pop modify pop_name set warning {on | off}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_access()

This API returns whether the specified access of the specified protected object is permitted by the user authenticated by the specified security context.

Syntax

```
unsigned long ivadmin_protobj_access(
ivadmin_context ctx,
const char *objid,
const char *permission_str,
azn_attrlist_h_t *app_context,
ivadmin_accessOutdata *outdata,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object.

permission_str
The permission string that describes the access requested. See the information about using access control policies in the *IBM Security Access Manager for Web: Administration Guide*.

app_context
The application context. See the description of the `azn_decision_access_allowed_ext()` function in the *IBM Security Access Manager for Web: Authorization C API Developer Reference* for information about application contexts.

Output

outdata
A pointer to an `ivadmin_accessOutdata` object. This object is populated with output data for the specified object access request. See "Enabling the return of permission information" in the *IBM Security Access Manager for Web: Authorization C API Developer Reference* for information about the data contained within this object. Free this structure when it is no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns whether the user identified by the specified security context is permitted the specified access on the specified object. To check multiple accesses or multiple objects with a single function, use the `ivadmin_protobj_multiaccess()` function.

Information is returned in an `ivadmin_accessOutdata` structure. The following information is returned:

access result
Either AZN_C_PERMITTED or AZN_C_NOT_PERMITTED. Use the `ivadmin_accessOutdata_getAccessResult()` function to extract this data.

response information
An `ivadmin_response` object that indicates the success or failure of the operation. Use the `ivadmin_accessOutdata_getResponseInfo()` function to extract the response object.

permission information
An `azn_attrlist_h_t` structure that contains supplemental permission information. Use the `ivadmin_accessOutdata_getPermInfo()` function to extract the permission information.

See the *IBM Security Access Manager for Web: Authorization C API Developer Reference* for more information about permissions and the `azn_attrlist_h_t` structure.

Command-line equivalent:

```
pdadmin object access object_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_attachacl()

This API attaches the specified access control list (ACL) to the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_attachacl(
ivadmin_context ctx,
const char *objid,
const char *aclid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object.

aclid The name of the access control list.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Attaches the specified ACL to the specified protected object. If the specified protected object already has an ACL attached, this function replaces that ACL with the new one. Understand Security Access Manager ACLs before using this function. See the information about using access control policies in the *IBM Security Access Manager for Web: Administration Guide*.

Command-line equivalent:

```
pdadmin acl attach object_name ACL_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_attachauthzrule()

This API attaches the specified authorization rule to the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_attachauthzrule(
ivadmin_context ctx,
const char *objid,
const char *authzruleid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object.

authzruleid
The name of the authorization rule.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Attaches the specified authorization rule to the specified protected object. If the specified protected object already has an authorization rule attached, this function replaces that authorization rule with the new one.

Command-line equivalent:

```
pdadmin authzrule attach object_name rule_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_attrdelkey()

This API deletes the specified extended attribute (name and value) from the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_attrdelkey(
ivadmin_context ctx,
const char *objid,
const char *attr_name,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object.

attr_name
The name of the extended attribute to delete.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified extended attribute (name and value) from the specified protected object.

Command-line equivalent:

```
pdadmin object modify object_name delete attribute_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_attrdelval()

This API deletes the specified value from the specified extended attribute key in the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_attrdelval(
ivadmin_context ctx,
char *popid,
char *attr_key,
char *attr_value,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

popid The name of the protected object.

attr_key
The name of the extended attribute.

attr_value
The name of the value to delete from the specified extended attribute.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified value from the specified extended attribute key in the specified protected object.

When you delete the last value for an attribute, it also deletes the attribute from the protected object. The *attr_value* deletes the specified value from the specified extended attribute key in the specified the protected object.

Command-line equivalent:

```
pdadmin object modify object_name delete attribute_name attribute_value
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_attrget()

This API returns the values that are associated with the specified extended attribute for the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_attrget(
ivadmin_protobj protobj,
const char *attr_key,
unsigned long *count,
char ***attr_value
);
```

Parameters

Input

protobj
The Security Access Manager protected object structure.

attr_key
The extended attribute to access.

count The number of values returned.

attr_value
An array of pointers to the extended attribute values returned. You must free the character data referenced by each pointer as well as the array of pointers when they are no longer needed.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the values for the extended attribute that are defined at the specified protected object.

Command-line equivalent:

```
pdadmin object show object_name attribute attribute_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_attrlist()

This API returns all the extended attributes that are associated with the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_attrlist(
ivadmin_protobj protobj,
unsigned long *count,
char ***attrs_list
);
```

Parameters

Input

protobj
The Security Access Manager protected object structure.

Output

count The number of extended attributes returned.

attrs_list
An array of pointers to the extended attributes returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists any objects grouped under the specified protected object. Alternatively, lists all the extended attributes that are defined at the specified protected object when the parameter `attribute` is specified.

Command-line equivalent:

```
pdadmin object list object_name attribute
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_attrput()

This API creates or changes an extended attribute with the specified name and sets a value, and adds the extended attribute to the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_attrput(
ivadmin_context ctx,
const char *objid,
const char *attr_name,
const char *attr_value,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object.

attr_name
The name of the extended attribute.

attr_value
The value for the extended attribute.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Creates or changes an extended attribute with the specified name and sets a value, and adds the extended attribute to the specified protected object.

If the extended attribute exists, this value is added to the existing values.

Command-line equivalent:

```
pdadmin object modify object_name set attribute attribute_name attribute_value
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_create()

This API creates a Security Access Manager protected object.

Syntax

```
unsigned long ivadmin_protobj_create(
ivadmin_context ctx,
const char *objid,
unsigned long type,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object that you want to create. The name can be any length and contain any character. Forward slash (/) characters are interpreted as part of the object hierarchy. You can attach access control lists at the various points indicated by the forward slash character.

type The type designation of protected object.

description
The description of the protected object.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

You must specify the *type* input parameter to `ivadmin_protobj_create()` . This object type is used by Web Portal Manager to display an appropriate icon with the object.

Table 33 on page 168 lists the supported object types.

Command-line equivalent:

```
pdadmin object create object_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_delete()

This API deletes the specified Security Access Manager protected object.

Syntax

```
unsigned long ivadmin_protobj_delete(
ivadmin_context ctx,
const char *objid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object to delete.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified Security Access Manager protected object.

Command-line equivalent:

```
pdadmin object delete object_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_detachacl()

This API detaches the access control list (ACL) from the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_detachacl(
ivadmin_context ctx,
const char *objid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Detaches the ACL from the specified protected object. Because only one ACL at a time can be attached to an object, the currently attached ACL is detached. Understand Security Access Manager ACLs before using this function. See the information about using access control policies in the *IBM Security Access Manager for Web: Administration Guide*.

Command-line equivalent:

```
pdadmin acl detach object_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_detachauthzrule()

This API detaches the authorization rule from the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_detachauthzrule(
ivadmin_context ctx,
const char *objid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object.

Output

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Detaches the authorization rule from the specified protected object. Only one authorization rule can be attached to an object simultaneously. So, the current authorized rule that is attached to the object space is detached. If the object space does not exist, then the error message **HPDBA0310E: The authorization rule specified was not found** is generated.

Command-line equivalent:

```
pdadmin authzrule detach object_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

Returns IVADMIN_FALSE if no authorization was attached to the protected object.

ivadmin_protobj_effattrget()

This API returns the values that are associated with the specified effective extended attribute for the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_effattrget(
ivadmin_protobj protobj,
const char *attr_key,
unsigned long *count,
char ***attr_value
);
```

Parameters

Input

protobj
The Security Access Manager protected object structure.

attr_key
The effective extended attribute to access.

Output

count The number of values returned.

attr_value
An array of pointers to the effective extended attribute values returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Protected objects without explicitly defined extended attributes inherit the first found set of extended attributes, which are defined at the parent object within the inheritance chain. The found set of extended attributes replaces the empty set of defined attributes. These attributes are referred to as *effective extended attributes.*

Returns the values that are associated with the specified effective extended attribute for the specified protected object.

Command-line equivalent:

```
pdadmin object show object_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_effattrlist()

This API lists all the effective extended attributes that are associated with the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_effattrlist(
ivadmin_protobj protobj,
unsigned long *count,
char ***attrs_list
);
```

Parameters

Input

protobj
The Security Access Manager protected object structure.

Output

count The number of extended attributes returned.

attrs_list
An array of pointers to the effective extended attributes returned. You must free the character data referenced by each pointer as well as the array of pointers when they are no longer needed.

Description

Protected objects without explicitly defined extended attributes inherit the first found set of extended attributes, which are defined at the parent object within the inheritance chain. The found set of extended attributes replaces the empty set of defined attributes. These attributes are referred to as *effective extended attributes*.

Lists all the effective extended attributes that are associated with the specified protected object.

Command-line equivalent:

```
pdadmin object show object_name
```

Return values

Returns the value associated with the specified effective extended attribute for the specified protected object.

ivadmin_protobj_exists()

This API returns an indication of whether the specified protected object exists.

Syntax

```
unsigned long ivadmin_protobj_exists(
ivadmin_context ctx,
const char *objid,
unsigned long *exists,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object.

Output

exists The indicator of whether the object exists.

IVADMIN_TRUE
The protected object exists.

IVADMIN_FALSE
The protected object does not exist

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Indicates whether the protected object exists in the policy database or as an object maintained by an administration service.

Command-line equivalent:

```
pdadmin object exists object_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_get3()

This API returns the specified protected object. To determine whether a protected object exists, see `ivadmin_protobj_exists()`.

Syntax

```
unsigned long ivadmin_protobj_get3(
ivadmin_context ctx,
const char *objid,
azn_attrlist_h_t *indata,
```

```
ivadmin_protobj *obj,
azn_attrlist_h_t *outdata,
unsigned long *resultcount,
char ***results,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object.

indata The pass through data that allows additional information to be communicated to the server. If a NULL is specified, it is ignored. For non-null inputs, a valid address for an `azn_attrlist_h_t` structure is expected. It is also assumed that the caller created this `azn_attrlist_h_t` structure with the `azn_attrlist_create ()` function. When this data is no longer required, free the associated memory with the `azn_attrlist_delete()` function.

Output

obj The returned object.

outdata
The pass through data that allows the server to communicate additional information to the caller. When the data is no longer required, free the associated memory with `azn_attrlist_delete()`.

resultcount
The number of result strings returned.

results
An array of pointers to the result strings returned. The result strings are the message strings returned by the task. These strings are typically output to a command-line interface (CLI) or log output and contain information about the success or failure of the task. You must free the character data referenced by each pointer and the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the specified protected object. To determine whether a protected object exists, see `ivadmin_protobj_exists()`.

Command-line equivalent:

```
pdadmin object show object_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_getaclid()

This API returns the identifier of the access control list (ACL), if any, that is attached to the specified protected object.

Syntax

```
const char* ivadmin_protobj_getaclid(
ivadmin_protobj protobj
);
```

Parameters

Input

protobj
A pointer to the protected object.

Description

Returns the identifier of the ACL that is attached to the specified protected object.

Do not free this ACL identifier. This data is maintained in the `ivadmin_protobj` object.

Command-line equivalent:

```
pdadmin object show object_name
```

The identifier of the attached ACL is part of the information returned by this `pdadmin object show` command.

Return values

Returns the identifier of the ACL that is attached to the specified protected object. If no ACL is attached to the protected object, the empty string is returned.

ivadmin_protobj_getauthzruleid()

This API returns the identifier of the authorization rule, if any, that is attached to the specified protected object.

Syntax

```
const char* ivadmin_protobj_getauthzruleid(
ivadmin_protobj protobj
);
```

Parameters

Input

protobj
A pointer to the protected object.

Description

Returns the identifier of the authorization rule that is attached to the specified protected object.

Do not free this authorization rule identifier. This data is maintained in the `ivadmin_protobj` object.

Command-line equivalent:

```
pdadmin object show object_name
```

The identifier of the attached authorization rule is part of the information returned by this `pdadmin object show` command.

Return values

Returns the identifier of the authorization rule that is attached to the specified protected object. An empty string is returned if no authorization rule is attached to the object.

ivadmin_protobj_getdesc()

This API returns the description of the specified protected object.

Syntax

```
const char* ivadmin_protobj_getdesc(
ivadmin_protobj protobj
);
```

Parameters

Input

protobj
A pointer to the protected object.

Description

Returns the description of the specified protected object. You must call `ivadmin_protobj_get3()` before calling this function.

Do not free this description. This data is maintained in the `ivadmin_protobj` object.

Command-line equivalent:

```
pdadmin object show object_name
```

The description is part of the information returned by this **pdadmin** command.

Return values

Returns the description of the specified protected object. There is no limit to the length of the description.

ivadmin_protobj_geteffaclid()

This API returns the identifier of the access control list (ACL), if any, that is in effect for the specified protected object. The effective ACL might be attached to the protected object, or it might be inherited from an object higher in the protected object space.

Syntax

```
const char* ivadmin_protobj_geteffaclid(
ivadmin_protobj protobj
);
```

Parameters

Input

protobj
A pointer to the protected object structure.

Description

Returns the identifier of the ACL that is in effect for the specified protected object.

Do not free this effective ACL identifier. This data is maintained in the `ivadmin_protobj` object.

Command-line equivalent:

```
pdadmin object show object_name
```

The identifier of the effective ACL is part of the information returned by this `pdadmin object show` command. If no access control list is attached to the protected object, the empty string is returned.

Return values

Returns the identifier of the ACL that is in effect for the specified protected object. If not effective ACL is found, returns an empty string.

ivadmin_protobj_geteffauthzruleid()

This API returns the identifier of the authorization rule, if any, that is in effect for the specified protected object. The effective authorization rule might be attached to the protected object or inherited from an object higher in the protected object space.

Syntax

```
const char* ivadmin_protobj_geteffauthzruleid(
ivadmin_protobj protobj
);
```

Parameters

Input

protobj
A pointer to the protected object.

Description

Returns the identifier of the authorization rule that is in effect for the specified protected object.

Do not free this effective authorization rule identifier. This data is maintained in the `ivadmin_protobj` object.

Command-line equivalent:

```
pdadmin object show object_name
```

The identifier of the authorization rule in effect is part of the information returned by this `pdadmin object show` command.

Return values

Returns the identifier of the authorization rule that is in effect for the specified protected object, or the empty string if no authorization rule is in effect for the object.

ivadmin_protobj_geteffpopid()

This API returns the identifier of the protected object policy, if any, that is in effect for the specified protected object. The effective POP might be attached to the protected object or inherited from an object higher in the protected object space.

Syntax

```
const char* ivadmin_protobj_geteffpopid(
ivadmin_protobj protobj
);
```

Parameters

Input

protobj
A pointer to the protected object.

Description

Returns the identifier of the protected object policy in effect for the specified protected object.

Do not free this effective protected object policy identifier. This data is maintained in the `ivadmin_protobj` object.

Command-line equivalent:

```
pdadmin object show object_name
```

The identifier of the protected object policy in effect is part of the information returned by this `pdadmin object show` command.

Return values

Returns the identifier of the protected object policy in effect for the specified protected object. If no effective protected object policy is found for the protected object, the empty string is returned.

ivadmin_protobj_getid()

This API returns the name of the specified protected object.

Syntax

```
const char* ivadmin_protobj_getid(
ivadmin_protobj protobj
);
```

Parameters

Input

protobj
A pointer to the protected object structure.

Description

Returns the name of the specified protected object. You must call `ivadmin_protobj_get2()` before calling this function.

Do not free this name. This data is maintained in the protected object structure `ivadmin_protobj`.

Command-line equivalent:

```
pdadmin object show object_name
```

The protected object name is part of the information returned by this **pdadmin** command.

Return values

Returns the name of the specified protected object. There is no limit to the length of the name.

ivadmin_protobj_getpolicyattachable()

This API returns the isPolicyAttachable attribute of the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_getpolicyattachable(
ivadmin_protobj protobj
);
```

Parameters

Input

protobj
The protected object structure.

Description

Returns the `isPolicyAttachable` attribute of the specified protected object. The `isPolicyAttachable` attribute of a protected object indicates whether a protected object policy can be attached to that protected object. The default value of this attribute is `yes`.

Command-line equivalent:

```
pdadmin object show object_name
```

The protected object `isPolicyAttachable` attribute is part of the information returned by this **pdadmin** command.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. Indicates that `isPolicyAttachable` is true.

IVADMIN_FALSE
Defined as 0. Indicates that `isPolicyAttachable` is false.

ivadmin_protobj_getpopid()

This API returns the identifier of the protected object policy, if any, for the specified protected object.

Syntax

```
const char* ivadmin_protobj_getpopid(
ivadmin_protobj protobj
);
```

Parameters

Input

protobj
A pointer to the protected object.

Description

Returns the identifier of the protected object policy that is attached to the specified protected object.

Do not free this protected object policy identifier. This data is maintained in the `ivadmin_protobj` object.

Command-line equivalent:

```
pdadmin object show object_name
```

The identifier of the attached protected object policy is part of the information returned by this **pdadmin** object show command.

Return values

Returns the identifier of the protected object policy that is attached to the specified protected object. If no protected object policy is found for the protected object, an empty string is returned.

ivadmin_protobj_gettype()

This API returns the type of the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_gettype(
ivadmin_protobj protobj
);
```

Parameters

Input

protobj
A pointer to the protected object structure.

Description

Returns the type of the specified protected object.

Command-line equivalent:

```
pdadmin object show object_name
```

The protected object type is part of the information returned by this **pdadmin** command.

Return values

Returns the type of the specified protected object.

Table 33 on page 168 in the description of the `ivadmin_objectspace_create()` function enumerates the types, values, and their descriptions.

ivadmin_protobj_list3()

This API lists the protected objects in the specified directory, not including subdirectories.

Syntax

```
unsigned long ivadmin_protobj_list3(
ivadmin_context ctx,
const char *objid,
azn_attrlist_h_t *indata,
unsigned long *objcount,
char ***objs,
azn_attrlist_h_t *outdata,
unsigned long *resultcount,
char ***results,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the parent protected object.

indata The pass through data that allows additional information to be communicated to the server. If a NULL is specified, it is ignored. For non-null inputs, a valid address for an `azn_attrlist_h_t` structure is expected. It is also assumed that the caller created this `azn_attrlist_h_t` structure with the `azn_attrlist_create()` function. When this data is no longer required, free the associated memory with the `azn_attrlist_delete()` function.

Output

objcount
The number of object names returned.

objs An array of pointers to the list of object names that exist directly below the specified parent object. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

outdata
The pass through data that allows the server to communicate additional information to the caller. When the data is no longer required, free the associated memory with the `azn_attrlist_delete()` function.

resultcount
The number of result strings returned.

results
An array of pointers to the result strings returned. The result strings are the message strings returned by the task. These strings are typically output on a command-line interface (CLI) or log output and contain information about the success or failure of the task. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists the protected objects in the specified directory, not including subdirectories. If an error occurs, NULL is returned.

Command-line equivalent:

```
pdadmin object list object_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_listbyacl()

This API finds and lists all protected objects that have the specified access control list attached.

Syntax

```
unsigned long ivadmin_protobj_listbyacl(
ivadmin_context ctx,
const char *aclid,
unsigned long *count,
char ***objids,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

aclid The name of the access control list.

Output

count The number of protected objects returned.

objids An array of pointers to the protected objects returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Finds and lists all protected objects that have the specified access control list attached.

Command-line equivalent:

```
pdadmin acl find ACL_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_listbyauthzrule()

This API finds and lists all protected objects that have the specified authorization rule attached.

Syntax

```
unsigned long ivadmin_protobj_listbyauthzrule(
ivadmin_context ctx,
const char *authzruleid,
```

```
unsigned long *count,
char ***objids,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

authzruleid
The name of the authorization rule.

Output

count The number of protected objects returned.

objids An array of pointers to protected objects returned. You must free the protected objects referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function and might contain informational, warning, or error information. Free this object when it is no longer needed.

Description

Finds and lists all protected objects that have the specified authorization rule attached. Free each protected object name pointer and the array of pointers when no longer needed.

Command-line equivalent:

```
pdadmin authzrule find rule_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_multiaccess()

This API determines whether the user authenticated in the specified security context has the specified accesses to the specified objects.

Syntax

```
unsigned long ivadmin_protobj_multiaccess(
ivadmin_context ctx,
char *objids[],
char *permission_strs[],
azn_attrlist_h_t*app_contexts[],
ivadmin_accessOutdata *outdata[],
int count
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objids The names of the protected objects.

permission_strs
The permission strings that describe the accesses requested. See the information about using access control policies in the *IBM Security Access Manager for Web: Administration Guide*.

app_contexts
The application contexts. See the description of the `azn_decision_access_allowed_ext()` function in the *IBM Security Access Manager for Web: Authorization C API Developer Reference* for information about application contexts.

count The number of objects to which access is being requested.

Output

outdata
An array of pointers to `ivadmin_accessOutdata` objects. These objects are populated with output data for each object access request. See "Enabling the return of permission information" in the *IBM Security Access Manager for Web: Authorization C API Developer Reference* for information about the data contained within these objects. Free each element and the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. The response object for the entire operation might indicate success even though one or more individual requests might fail. The output might be empty or contains more than one error, informational, and warning messages. Free this object when it is no longer needed.

Description

Provided a group of objects and a group of requested accesses, returns whether the user specified in the input context is permitted the specified accesses on each object. Use the `ivadmin_protobj_access()` function to manipulate a single object.

An array of `ivadmin_accessOutdata` objects is returned, each of which contains information for one access request. The following information is returned in each object:

access result
Either AZN_C_PERMITTED or AZN_C_NOT_PERMITTED. Use the `ivadmin_accessOutdata_getAccessResult()` function to extract this data.

response information
An `ivadmin_response` object that indicates the success or failure of the operation. Use the `ivadmin_accessOutdata_getResponseInfo()` function to extract the response object.

permission information
An `azn_attrlist_h_t` structure that contains supplemental permission information. Use the `ivadmin_accessOutdata_getPermInfo()` function to extract the permission information.

See the *IBM Security Access Manager for Web: Authorization C API Developer Reference* for more information about permissions and the `azn_attrlist_h_t` structure.

Command-line equivalent:

```
pdadmin object access object_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_setdesc()

This API creates or changes the description of the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_setdesc(
ivadmin_context ctx,
const char *objid,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object for which a new description is set.

description
The new description for the protected object.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the description of the specified protected object.

Command-line equivalent:

```
pdadmin object modify object_name description new_description
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_setpolicyattachable()

This API enables or disables the `isPolicyAttachable` attribute of the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_setpolicyattachable(
ivadmin_context ctx,
const char *objid,
unsigned long flag,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object.

flag The flag that contains the value of the `isPolicyAttachable` attribute. Supported values are `IVADMIN_TRUE` or 1 and `IVADMIN_FALSE` or 0.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Enables or disables the `isPolicyAttachable` attribute of the specified protected object. The `isPolicyAttachable` attribute of a protected object indicates whether a protected object policy can be attached to that protected object.

The initial value of this attribute is `yes`.

Command-line equivalent:

```
pdadmin object modify object_name isPolicyAttachable {yes | no}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_protobj_settype()

This API sets the type field of the specified protected object.

Syntax

```
unsigned long ivadmin_protobj_settype(
ivadmin_context ctx,
const char *objid,
unsigned long type,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

objid The name of the protected object.

type The new type for the object.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the type field of the specified protected object.

Command-line equivalent:

```
pdadmin object modify object_name type new_type
```

Table 33 on page 168 lists the supported object types.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_response_getcode()

This API returns the message code.

Syntax

```
unsigned long ivadmin_response_getcode(
ivadmin_response rsp,
unsigned long index
);
```

Parameters

Input

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

index The zero-based index of the message code requested.

Description

Returns the error or warning code that is associated with the message.

Return values

Returns the error or warning code that is associated with the message.

ivadmin_response_getcount()

This API returns the number of messages in the response object.

Syntax

```
unsigned long ivadmin_response_getcount(
ivadmin_response rsp
);
```

Parameters

Input

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the number of messages in the response object.

Return values

Returns the number of messages in the response object.

ivadmin_response_getmessage()

This API returns the message text from the specified index location in the response object.

Syntax

```
const char* ivadmin_response_getmessage(
ivadmin_response rsp,
unsigned long index
);
```

Parameters

Input

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

index The zero-based index of message text requested.

Description

Returns the message text from the specified index location in the response object.

Do *not free this object. This data is maintained in the response structure.*

Return values

Returns the message text from the specified index location in the response object.

ivadmin_response_getmodifier()

This API returns the message modifier from the specified index location in the response object.

Syntax

```
unsigned long ivadmin_response_getmodifier(
ivadmin_response rsp,
unsigned long index
);
```

Parameters

Input

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

index The zero-based index of the message modifier requested.

Description

Returns the message modifier from the specified index location in the response object. The modifier can be an error, a warning, or information. The following values are defined:

```
#define IVADMIN_RESPONSE_INFO 0
#define IVADMIN_RESPONSE_WARNING 1
#define IVADMIN_RESPONSE_ERROR 2
```

Return values

Returns the message modifier from the specified index location in the response object.

ivadmin_response_getok()

This API returns an indicator of the success of the operation.

Syntax

```
unsigned long ivadmin_response_getok(
ivadmin_response rsp
);
```

Parameters

Input

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns an indicator of the success of the operation.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_server_gettasklist()

This API Returns the list of tasks from the server.

Syntax

```
unsigned long ivadmin_server_gettasklist(
ivadmin_context ctx,
const char *server,
azn_attrlist_h_t *indata,
unsigned long *taskcount,
char ***tasks,
azn_attrlist_h_t *outdata,
unsigned long *resultcount,
char ***results,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

server The name of the server to notify of a database update. This parameter is optional. If NULL is specified, all servers configured to receive database update notifications are notified.

indata The pass through data that allows additional information to be communicated to the server. If NULL is specified, it is ignored. For non-null inputs, a valid address for an `azn_attrlist_h_t` structure is expected. It is also assumed that the caller created this `azn_attrlist_h_t` structure with the `azn_attrlist_create()` function. When this data is no longer required, free the associated memory with the `azn_attrlist_delete()` function.

Output

taskcount
The number of task strings returned.

tasks An array of pointers to the list of tasks currently supported by this server. The task strings are typically in the supported command-line interface (CLI) syntax. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

outdata
The pass through data that allows the server to communicate additional information to the caller. When the data is no longer required, free the associated memory by with the `azn_attrlist_delete()` function.

resultcount
The number of result strings returned.

results
An array of pointers to the result strings returned. The result strings are the message strings returned by the task. These strings are typically output on a command-line interface (CLI) or log output and contain information about the success or failure of the task. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the list of tasks from the server. If no tasks are supported, or an error occurs, NULL is returned.

Command-line equivalent:

```
pdadmin server listtasks server_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_server_performtask()

This API sends a command to an authorization server.

Syntax

```
unsigned long ivadmin_server_performtask(
ivadmin_context ctx,
const char *server,
const char *task,
azn_attrlist_h_t *indata,
azn_attrlist_h_t *outdata,
unsigned long *resultcount,
char ***results,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

server The name of server to notify of database update. This parameter is optional. If `NULL` is specified, all servers configured to receive database update notifications is notified.

task The task to do.

indata The pass through data that allows additional information to be communicated to the server. If `NULL` is specified, it is ignored. For non-null inputs, a valid address for an `azn_attrlist_h_t` structure is expected. It is also assumed that the caller created the `azn_attrlist_h_t` structure with

the azn_attrlist_create() function. When this data is no longer required, free the associated memory by with the azn_attrlist_delete() function.

Output

outdata
The pass through data that allows the server to communicate additional information to the caller. When the data is no longer required, free the associated memory by with the azn_attrlist_delete() function.

resultcount
The number of result strings returned.

results
An array of pointers to the result strings returned. The result strings are the message strings returned by the task. These strings are typically output on a command-line interface (CLI) or log output and contain information about the success or failure of the task. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sends a command to the authorization server.

Command-line equivalent:

```
pdadmin server task server_name task_to_perform
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_server_replicate()

This API notifies the authorization servers to receive database updates.

Syntax

```
unsigned long ivadmin_server_replicate(
ivadmin_context ctx,
const char *server,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

server The name of the server to notify of a database update. This parameter is optional. If `NULL` is specified, all servers configured to receive database update notifications are notified.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Notifies the authorization servers to receive database updates. If a server name is specified, but is not configured to receive database updates, an error message is displayed. If no server name is specified, the process of notifying all configured servers is initiated, but error messages are not displayed for individual servers. The caller must have the authority to do server administration tasks on the policy server. (The `azn_operation_server_admin` permission is required on the policy server object.)

Command-line equivalent:

```
pdadmin server replicate [server-name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. If a server is specified, this value indicates the successful notification and database replication by that server. If no server is specified, this value indicates that the policy server began to notify each authorization server. In this case, a return code of `IVADMIN_TRUE` is not an indication of successful notification or replication for any one of the servers.

IVADMIN_FALSE
Defined as 0. If a server is specified, this value indicates the failure of the notification and database replication by that server. If no server is specified, this value indicates that a failure occurred in requesting that the policy server notifies each authorization server.

ivadmin_ssocred_create()

This API creates a single sign-on credential.

Syntax

```
unsigned long ivadmin_ssocred_create(
ivadmin_context ctx,
const char *ssoid,
unsigned long ssotype,
const char *userid,
const char *ssouserid,
const char *ssopassword,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssoid Single sign-on resource name with which the single sign-on credential is associated. This resource must exist.

ssotype
Single sign-on resource type. The following types are defined:
- IVADMIN_SSOCRED_SSOWEB
- IVADMIN_SSOCRED_SSOGROUP

userid The user ID that is associated with the single sign-on credential.

ssouserid
The user name that this user uses to access the specified resource.

ssopassword
The password that this user uses to access the specified resource.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates a single sign-on credential.

Command-line equivalent:

```
pdadmin rsrccred create resource_name rsrcuser resource_userid rsrcpwd \
resource_password rsrctype {web | group} user user_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssocred_delete()

This API deletes a single sign-on credential.

Syntax

```
unsigned long ivadmin_ssocred_delete(
ivadmin_context ctx,
const char *ssoid,
unsigned long ssotype,
const char *userid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssoid Single sign-on resource name with which the single sign-on credential is associated.

ssotype
Single sign-on resource type. The following types are defined:

- IVADMIN_SSOCRED_SSOWEB
- IVADMIN_SSOCRED_SSOGROUP

userid
The user ID that is associated with the single sign-on credential.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes a single sign-on credential.

Command-line equivalent:

```
pdadmin rsrccred delete resource_name rsrctype {web | group} user user_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssocred_get()

This API returns the specified single sign-on credential.

Syntax

```
unsigned long ivadmin_ssocred_get(
ivadmin_context ctx,
const char *ssoid,
unsigned long ssotype,
const char *userid,
ivadmin_ssocred *ssocred,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssoid Single sign-on resource name with which the single sign-on credential is associated.

ssotype
Single sign-on resource type. The following types are defined:

- IVADMIN_SSOCRED_SSOWEB
- IVADMIN_SSOCRED_SSOGROUP

userid The user name that is associated with the single sign-on credential.

Output

ssocred
The returned single sign-on credential. Free this credential when it is no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the specified single sign-on credential.

Specify the single sign-on credential type when using this function. The following single sign-on credential types are defined:

```
#define IVADMIN_SSOCRED_SSOWEB0
#define IVADMIN_SSOCRED_SSOGROUP1
```

Command-line equivalent:

```
pdadmin rsrccred show resource_name rsrctype {web | group} user user_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssocred_getid()

This API returns the name of the single sign-on resource that is associated with this credential.

Syntax

```
const char* ivadmin_ssocred_getid(
ivadmin_ssocred ssocred
);
```

Parameters

Input

ssocred
A pointer to the single sign-on credential.

Description

Returns the name of the single sign-on resource that is associated with this credential. You must call `ivadmin_ssocred_get()` to obtain an `ivadmin_ssocred`object before calling this function.

Do not free this name. This data is maintained in the single sign-on credential structure (`ivadmin_ssocred`).

Command-line equivalent:

```
pdadmin rsrccred show resource_name rsrctype {web | group} user user_name
```

The credential identifier is part of the information returned by the **pdadmin** command.

Return values

Returns the name of the single sign-on resource that is associated with this credential.

User registry difference: The maximum length of the name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

ivadmin_ssocred_getssopassword()

This API returns the password that is associated with the single sign-on credential.

Syntax

```
const char* ivadmin_ssocred_getssopassword(
ivadmin_ssocred ssocred
);
```

Parameters

Input

ssocred
A pointer to the single sign-on credential.

Description

Returns the password that is associated with the single sign-on credential. You must call `ivadmin_ssocred_get()` to obtain an `ivadmin_ssocred` object before calling this function.

Do not free this password. This data is maintained in the single sign-on credential structure (`ivadmin_ssocred`).

Return values

Returns the password that is associated with the single sign-on credential. There is no limit to the length of the password.

ivadmin_ssocred_getssouser()

This API returns the name of the user that is associated with the specified single sign-on credential.

Syntax

```
const char* ivadmin_ssocred_getssouser(
ivadmin_ssocred ssocred
);
```

Parameters

Input

ssocred
A pointer to the single sign-on credential.

Description

Returns the name of the user that is associated with the specified single sign-on credential. You must call `ivadmin_ssocred_get()` to obtain an `ivadmin_ssocred` object before calling this function.

Do not free this name. This data is maintained in the single sign-on credential structure (`ivadmin_ssocred`).

Return values

Returns the name of the user that is associated with the specified single sign-on credential.

User registry difference: The maximum length of the name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

ivadmin_ssocred_gettype()

This API returns the type of the single sign-on resource that is associated with the specified single sign-on credential.

Syntax

```
unsigned long ivadmin_ssocred_gettype(
ivadmin_ssocred ssocred
);
```

Parameters

Input

ssocred
A pointer to the single sign-on credential.

Description

Returns the type of the single sign-on resource that is associated with the specified single sign-on credential. You must call `ivadmin_ssocred_get()` to obtain an `ivadmin_ssocred` object before calling this function.

The defined types are:

```
#define IVADMIN_SSOCRED_SSOWEB     0
#define IVADMIN_SSOCRED_SSOGROUP   1
```

Do not free the resource credential type (integer) when it is no longer needed. This data is maintained in the `ivadmin_ssocred` object.

Command-line equivalent:

```
pdadmin rsrccred show resource_name rsrctype {web | group} user user_name
```

The credential type is part of the information returned by the **pdadmin** command.

Return values

Returns the type of the single sign-on resource that is associated with the specified single sign-on credential.

ivadmin_ssocred_getuser()

This API returns the name of the user that is associated with the single sign-on credential.

Syntax

```
const char* ivadmin_ssocred_getuser(
ivadmin_ssocred ssocred
);
```

Parameters

Input

ssocred
A pointer to the single sign-on credential.

Description

Returns the name of the user that is associated with the single sign-on credential. You must call `ivadmin_ssocred_get()` to obtain an **ivadmin_ssocred** object before calling this function.

Do not free this name. This data is maintained in the single sign-on credential structure (`ivadmin_ssocred`).

Command-line equivalent:

```
pdadmin rsrccred show resource_name rsrctype {web | group} user user_name
```

The user name is part of the information returned by the **pdadmin** command.

Return values

Returns the name of the user that is associated with the single sign-on credential.

User registry difference: The maximum length of the name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

ivadmin_ssocred_list()

This API lists the single sign-on credentials for the specified user.

Syntax

```
unsigned long ivadmin_ssocred_list(
ivadmin_context ctx,
const char *userid,
```

```
unsigned long *count,
ivadmin_ssocred **ssocreds,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid
The user ID of the user for whom the single sign-on credentials are retrieved.

Output

count The number of single sign-on credentials returned.

ssocreds
An array of pointers to single sign-on credentials. You must free the data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists the single sign-on credentials for the specified user.

Command-line equivalent:

```
pdadmin rsrccred list user user_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssocred_set()

This API creates or changes a single sign-on credential.

Syntax

```
unsigned long ivadmin_ssocred_set(
ivadmin_context ctx,
const char *ssoid,
unsigned long ssotype,
const char *userid,
const char *ssouserid,
const char *ssopassword,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssoid The single sign-on resource name with which the single sign-on credential is associated.

ssotype
The single sign-on resource type. The following types are defined:
- IVADMIN_SSOCRED_SSOWEB
- IVADMIN_SSOCRED_SSOGROUP

userid The user name associated with the single sign-on credential.

ssouserid
The user name that the user (as specified by the input parameter `userid`) uses to access the specified resource.

ssopassword
The password that this user uses to access the specified resource.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes a single sign-on credential.

Command-line equivalent:

```
pdadmin rsrccred modify resource_name rsrctype {web | group} set \
[-rsrcuser resource_userid] [-rsrcpwd resource_password] user user_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssogroup_addres()

This API adds a single sign-on resource to a single sign-on resource group.

Syntax

```
unsigned long ivadmin_ssogroup_addres(
ivadmin_context ctx,
const char *ssogroupid,
const char *ssoid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssogroupid
Single sign-on resource group name.

ssoid The new member single sign-on resource name.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Adds a single sign-on resource to a single sign-on resource group. Security Access Manager does not support a resource group as a resource group member.

Command-line equivalent:

```
pdadmin rsrcgroup modify resource_group_name add rsrcname resource_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssogroup_create()

This API creates a single sign-on group resource.

Syntax

```
unsigned long ivadmin_ssogroup_create(
ivadmin_context ctx,
const char *ssogroupid,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssogroupid
The single sign-on group resource name.

description
The description of the single sign-on group resource.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates a single sign-on group resource.

Command-line equivalent:

```
pdadmin rsrcgroup create resource_group_name [-desc description]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssogroup_delete()

This API deletes a single sign-on group resource.

Syntax

```
unsigned long ivadmin_ssogroup_delete(
ivadmin_context ctx,
const char *ssogroupid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssogroupid
The single sign-on group resource name.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes a single sign-on group resource.

Command-line equivalent:

```
pdadmin rsrcgroup delete resource_group_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssogroup_get()

This API returns the specified single sign-on group resource.

Syntax

```
unsigned long ivadmin_ssogroup_get(
ivadmin_context ctx,
const char *ssogroupid,
ivadmin_ssogroup *ssogroup,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssogroupid
Single sign-on group resource name.

Output

ssogroup
The returned single sign-on group resource. Free the memory that contains the returned single sign-on group resource when it is no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the specified single sign-on group resource. The `ivadmin_ssogroup` object contains the resource group name, the resource group description, and a list of the names of the resource group members. The resource group members are the individual web resources (servers).

Command-line equivalent:

```
pdadmin rsrcgroup show resource_group_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssogroup_getdescription()

This API returns the description of the single sign-on group resource.

Syntax

```
const char* ivadmin_ssogroup_getdescription(
ivadmin_ssogroup ssogroup
);
```

Parameters

Input

ssogroup
A pointer to the single sign-on group resource.

Description

Returns the description of the single sign-on group resource. You must call `ivadmin_ssogroup_get()` to obtain an `ivadmin_ssogroup` object before calling this function.

Do not free this description. This data is maintained in the single sign-on group resource structure.

Command-line equivalent:

```
pdadmin rsrcgroup show resource_group_name
```

The description is part of the information returned by the **pdadmin** command.

Return values

Returns the description of the single sign-on group resource. The maximum length of the description is 1024 characters.

ivadmin_ssogroup_getid()

This API returns the name of the single sign-on group resource.

Syntax

```
const char* ivadmin_ssogroup_getid(
ivadmin_ssogroup ssogroup
);
```

Parameters

Input

ssogroup
A pointer to the single sign-on group resource.

Description

Returns the name of the single sign-on group resource. You must call `ivadmin_ssogroup_get()` to obtain an `ivadmin_ssogroup` object before calling this function.

Do not free this name. This data is maintained in the single sign-on group resource structure.

Command-line equivalent:

```
pdadmin rsrcgroup show resource_group_name
```

The name is part of the information returned by the **pdadmin** command.

Return values

Returns the name of the single sign-on group resource.

User registry difference: The maximum length of the name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

ivadmin_ssogroup_getresources()

This API returns the member single sign-on resource names for the specified single sign-on group.

Syntax

```
unsigned long ivadmin_ssogroup_getresources(
ivadmin_ssogroup ssogroup,
unsigned long *count,
char ***ssoids
);
```

Parameters

Input

ssogroup
A pointer to the single sign-on group resource.

Output

count The number of single sign-on resource names returned.

ssoids An array of pointers to the single sign-on resource names returned. You must free the character data referenced by each pointer as well as the array of pointers when they are no longer needed.

Description

Returns the member single sign-on resource names.

Command-line equivalent:

```
pdadmin rsrcgroup show resource_group_name
```

The resource name is part of the information returned by the **pdadmin** command.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssogroup_list()

This API lists all the single sign-on group resource names.

Syntax

```
unsigned long ivadmin_ssogroup_list(
ivadmin_context ctx,
unsigned long *count,
char ***ssogroupids,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

count The number of single sign-on group resource names returned.

ssogroupids
An array of pointers to the single sign-on group resource names returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists all the single sign-on group resource names.

Command-line equivalent:

```
pdadmin rsrcgroup list
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssogroup_removeres()

This API removes a single sign-on resource from the specified single sign-on resource group.

Syntax

```
unsigned long ivadmin_ssogroup_removeres(
ivadmin_context ctx,
const char *ssogroupid,
const char *ssoid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssogroupid
The single sign-on resource group name.

ssoid The member single sign-on resource name to remove.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Removes a single sign-on resource from the specified single sign-on resource group.

Command-line equivalent:

```
pdadmin rsrcgroup modify resource_group_name remove rsrcname resource_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssoweb_create()

This API creates a single sign-on web resource.

Syntax

```
unsigned long ivadmin_ssoweb_create(
ivadmin_context ctx,
const char *ssowebid,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssowebid
The single sign-on web resource name.

description
The description of the single sign-on web resource.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates a single sign-on web resource. The name of the web server does not need to match the junction. You can use this function call before joining the web server to the Security Access Manager WebSEAL server.

Command-line equivalent:

```
pdadmin rsrc create resource_name [-desc description]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssoweb_delete()

This API deletes the specified single sign-on web resource.

Syntax

```
unsigned long ivadmin_ssoweb_delete(
ivadmin_context ctx,
const char *ssowebid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssowebid
The name of the single sign-on web resource to delete.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the specified single sign-on web resource.

Command-line equivalent:

```
pdadmin rsrc delete resource_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssoweb_get()

This API returns the specified single sign-on web resource.

Syntax

```
unsigned long ivadmin_ssoweb_get(
ivadmin_context ctx,
const char *ssowebid,
ivadmin_ssoweb *ssoweb,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

ssowebid
The name of the single sign-on web resource.

Output

ssoweb
The returned single sign-on web resource. Free the memory for the single sign-on web resource (`ivadmin_ssoweb`) when it is no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the specified single sign-on web resource.

Command-line equivalent:

```
pdadmin rsrc show resource_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_ssoweb_getdescription()

This API returns the description of the specified single sign-on web resource.

Syntax

```
const char* ivadmin_ssoweb_getdescription(
ivadmin_ssoweb ssoweb
);
```

Parameters

Input

ssoweb
A pointer to the single sign-on web resource.

Description

Returns the description of the specified single sign-on web resource. You must call `ivadmin_ssoweb_get()` to obtain an `ivadmin_ssoweb` object before calling this function.

Do not free this description. This data is maintained in the single sign-on web resource structure (`ivadmin_ssoweb`).

Command-line equivalent:

```
pdadmin rsrc show resource_name
```

The description is part of the information returned by the **pdadmin** command.

Return values

Returns the description of the specified single sign-on web resource. The maximum length of the description is 1024 characters.

ivadmin_ssoweb_getid()

This API returns the name (identifier) of the specified single sign-on web resource.

Syntax

```
const char* ivadmin_ssoweb_getid(
ivadmin_ssoweb ssoweb
);
```

Parameters

Input

ssoweb
A pointer to the single sign-on web resource.

Description

Returns the name (identifier) of the specified single sign-on web resource. You must call `ivadmin_ssoweb_get()` to obtain an `ivadmin_ssoweb` object before calling this function.

Do not free this name. This data is maintained in the single sign-on web resource structure (`ivadmin_ssoweb`).

User registry difference: The maximum length of the name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

Command-line equivalent:

```
pdadmin rsrc show resource_name
```

The name is part of the information returned by the **pdadmin** command.

Return values

Returns the name (identifier) of the specified single sign-on web resource.

ivadmin_ssoweb_list()

This API lists all the single sign-on web resource names.

Syntax

```
unsigned long ivadmin_ssoweb_list(
ivadmin_context ctx,
unsigned long *count,
char ***ssowebids,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

Output

count The number of single sign-on web resource names returned.

ssowebids
An array of pointers to the single sign-on web resource names returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists all the single sign-on web resource names.

Command-line equivalent:

```
pdadmin rsrc list
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_create3()

This API creates a user in the user registry that is used by the Security Access Manager policy server and initially associates that user with one or more groups.

Syntax

```
unsigned long ivadmin_user_create3(
ivadmin_context ctx,
const char *userid,
const char *dn,
const char *cn,
const char *sn,
const char *pwd,
unsigned long group_count,
const char **groups,
unsigned long ssouser,
unsigned long nopwdpolicy,
ivadmin_response rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid Security Access Manager user name.

dn The user registry distinguished name.

cn The user registry attribute common name.

sn The user registry attribute surname.

pwd The user registry attribute password.

group_count
The number of groups to which the user initially belongs.

groups
The initial user registry groups to which the user belongs. Specify NULL to indicate no initial group membership.

ssouser
The user can have single sign-on credentials.

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

nopwdpolicy
Password policy is not enforced during creation. This setting has no effect on password policy enforcement after user creation.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates a user in the user registry. This user is used by the Security Access Manager policy server and initially associates that user with one or more groups. Accounts are created as not valid by default. Use `ivadmin_user_setaccountvalid()` to enable the account.

Groups created in the Active Directory Lightweight Directory Service (AD LDS) user registry must be created in the same AD LDS partition where the Security Access Manager Management Domain information is stored.

User registry difference: Leading and trailing blanks in a user name do not make the name unique when using an LDAP or Active Directory user registry. To keep name processing consistent regardless of what user registry is being used, do not define user names with leading or trailing blanks.

Command-line equivalent:

```
pdadmin user create [-gsouser] [-no-password-policy] user_name dn cn sn \
pwd group

pdadmin user create [-gsouser] [-no-password-policy] user_name dn cn sn \
pwd (group_1 group_2 ... group_n)
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_delete2()

This API deletes the Security Access Manager user and optionally deletes the user from the user registry.

Syntax

```
unsigned long ivadmin_user_delete2(
ivadmin_context ctx
const char *userid,
unsigned long registry
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The Security Access Manager user name.

registry
Specifies whether to delete just the Security Access Manager user (`IVADMIN_FALSE`) or to delete the user name and information deleted from Security Access Manager and the user registry (`IVADMIN_TRUE`).

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Deletes the Security Access Manager user and optionally deletes the user from the user registry. If *registry* is `IVADMIN_FALSE`, the user is deleted only from Security

Access Manager, but the information about this user remains in the user registry. If *registry* is IVADMIN_TRUE, the user is deleted from Security Access Manager and the user registry.

Command-line equivalent:

```
pdadmin user delete [-registry] user_name
```

The –registry option of the user delete command causes the entire user object to be deleted from the user registry.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_get()

This API returns the user object for the specified user.

Syntax

```
unsigned long ivadmin_user_get(
ivadmin_context ctx,
const char *userid,
ivadmin_ldapuser *user,
ivadmin_response rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The Security Access Manager user name.

Output

user The Security Access Manager user object returned. Free this memory when it is no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the user object for the specified user.

Free the memory used by the ivadmin_ldapuser object when it is no longer needed.

Command-line equivalent:

```
pdadmin user show user_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getaccexpdate()

This API returns the account expiration date for the specified user.

Syntax

```
unsigned long ivadmin_user_getaccexpdate(
ivadmin_context ctx,
const char *userid,
unsigned long *seconds,
unsigned long *unlimited,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

seconds
The returned date and time of the expiration of the specified user account. This value is the number of seconds since 00:00:00 Universal time, 1 January 1970 (same as `time_t`).

unlimited
The `account-expiration-not-restricted` indicator to return

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the account expiration date for the specified user.

Command-line equivalent:

```
pdadmin policy get account-expiry-date [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getaccountvalid()

This API returns the account-valid indicator from the specified user object.

Syntax

```
unsigned long ivadmin_user_getaccountvalid(
ivadmin_ldapuser user
);
```

Parameters

Input

user A pointer to the user structure.

Description

Returns the account-valid indicator from the specified user object.

Command-line equivalent:

```
pdadmin user show user_name
```

The account-valid status is part of the information returned by the **pdadmin** command.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getbydn()

This API returns the Security Access Manager user object with the user registry distinguished name.

Syntax

```
unsigned long ivadmin_user_getbydn(
ivadmin_context ctx,
const char *dn,
ivadmin_ldapuser *user,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

dn The user registry distinguished name of the user.

Output

user The Security Access Manager user object returned. Free the memory for this object when it is no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the Security Access Manager user object with the user registry distinguished name.

User registry difference: The maximum length of the distinguished name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

Command-line equivalent:

```
pdadmin user show-dn dn
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getcn()

This API returns the user registry common name attribute from the specified user object.

Syntax

```
const char* ivadmin_user_getcn(
ivadmin_ldapuser user
);
```

Parameters

Input

user A pointer to the user structure.

Description

Returns the user registry common name attribute from the specified user object.

Do not free the character string that is returned. This data is maintained in the `ivadmin_ldapuser` object.

Command-line equivalent:

```
pdadmin user show user_name
```

The user registry common name for the user is part of the information returned by the **pdadmin** command.

Return values

Returns the user registry common name attribute from the specified user object.

User registry difference: The maximum length of the common name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

ivadmin_user_getdescription()

This API returns the user description from the specified user object.

Syntax

```
const char* ivadmin_user_getdescription(
ivadmin_ldapuser user
);
```

Parameters

Input

user A pointer to the user structure.

Description

Returns the user description from the specified user object.

Do not free the character string that is returned. This data is maintained in the `ivadmin_ldapuser` object.

Command-line equivalent:

```
pdadmin user show user_name
```

The user description is part of the information returned by the **pdadmin** command.

Return values

Returns the user description from the specified user object. The maximum length of the description is 1024 characters.

ivadmin_user_getdisabletimeint()

This API returns the amount of time to disable the specified user account if the maximum number of login failures is exceeded.

Syntax

```
unsigned long ivadmin_user_getdisabletimeint(
ivadmin_context ctx,
const char *userid,
unsigned long *seconds,
unsigned long *disable,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

Output

seconds
The specified number of seconds that the user account is disabled if the maximum number of login failures is exceeded.

disable
The user account to disable when the maximum number of login failures is exceeded. An administrator action is required to enable the account.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the amount of time to disable each user account if the maximum number of login failures is exceeded.

Command-line equivalent:

```
pdadmin policy get disable-time-interval [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getdn()

This API returns the user registry distinguished name from the specified user object.

Syntax

```
const char* ivadmin_user_getdn(
ivadmin_ldapuser user
);
```

Parameters

Input

user A pointer to the user structure.

Description

Returns the user registry distinguished name from the specified user object.

Do not free the character string that is returned. This data is maintained in the `ivadmin_ldapuser` object.

Command-line equivalent:

```
pdadmin user show user_name
```

The user registry distinguished name for the user is part of the information returned by the **pdadmin** command.

Return values

Returns the user registry distinguished name from the specified user object.

User registry difference: The maximum length of the distinguished name varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

ivadmin_user_getid()

This API returns the user name from the specified user object.

Syntax

```
const char* ivadmin_user_getid(
ivadmin_ldapuser user
);
```

Parameters

Input

user A pointer to the user structure.

Description

Returns the user name from the specified user object.

Do not free the character string that is returned. This data is maintained in the `ivadmin_ldapuser` object.

Command-line equivalent:

```
pdadmin user show user_name
```

The user name (login identifier) is part of the information returned by the **pdadmin** command.

Return values

Returns the user name from the specified user object. The maximum length of the name is 256 characters.

ivadmin_user_getlastlogin()

This API returns the last login time of a specific user account.

Syntax

```
unsigned long ivadmin_user_getlastlogin(ivadmin_ldapuser user);
```

Parameters

Input

ivadmin_ldapuser user
The user account name

Description

When the **pdadmin** command *user show* is run, it displays an additional line of information that contains the Last login and Last password change time. The local time of the computer on which **pdadmin** was run is displayed.

Return values

The following values are returned:

- 0 is returned if user is NULL
- A long integer value that indicates the date and time that the user last logged in

ivadmin_user_getmaxconcurwebsess()

This API returns the maximum number of concurrent web sessions that are allowed for the specified user account.

Syntax

```
unsigned long ivadmin_user_getmaxconcurwebsess(
ivadmin_context ctx,
const char *userid,
unsigned long *sessions,
unsigned long *displace,
unsigned long *unlimited,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

sessions
The maximum number of concurrent web sessions that are allowed. If nonzero, the parameters `unset`, `displace`, and `unlimited` are set to false. A value of zero is not permitted.

displace
The policy is set to displace if set to true. If set to false, the policy is set as specified. Only one of the following parameters can be set to true: `displace`, `unlimited`, or `unset`

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

unlimited
The policy is set to unlimited if set to true. If set to false, the policy is set as specified. Only one of the following parameters can be set to true: displace, unlimited, or unset

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified. Only one of the following parameters can be set to true: displace, unlimited, or unset

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the value of the maximum number of concurrent web sessions for each user account.

This policy applies only to certain components of Security Access Manager. A *web session is a user session that is maintained by these Security Access Manager systems: Security Access Manager WebSEAL and Security Access Manager Plug-ins for Web Servers.*

Command-line equivalent:

```
pdadmin policy get max-concurrent-web-sessions [user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The context has a delegated credential.

IVADMIN_FALSE
Defined as 0. The context does not have a delegated credential.

ivadmin_user_getmaxlgnfails()

This API returns the maximum number of login failures that are allowed for the specified user account.

Syntax

```
unsigned long ivadmin_user_getmaxlgnfails(
ivadmin_context ctx,
const char *userid,
unsigned long *failures,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

Output

failures
The maximum number of login failures allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the maximum number of login failures that are allowed for the specified user account.

Command-line equivalent:

```
pdadmin policy get max-login-failures [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getmaxpwdage()

This API returns the maximum password age for the specified user account.

Syntax

```
unsigned long ivadmin_user_getmaxpwdage(
ivadmin_context ctx,
const char *userid,
unsigned long *seconds,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

Output

seconds
The maximum lifetime, in seconds, before expiration of a password. A value of zero means the password never expires.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the maximum password age for the specified user account.

Command-line equivalent:

```
pdadmin policy get max-password-age [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getmaxpwdrepchars()

This API returns the maximum number of repeated characters that are allowed in a password for the specified user account.

Syntax

```
unsigned long ivadmin_user_getmaxpwdrepchars(
ivadmin_context ctx,
const char *userid,
unsigned long *chars,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

Output

chars The maximum number of repeated characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the maximum number of repeated characters that are allowed in a password for the specified user account.

Command-line equivalent:

```
pdadmin policy get max-password-repeated-chars [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getmemberships()

This API returns a list of groups in which the specified user is a member.

Syntax

```
unsigned long ivadmin_user_getmemberships(
ivadmin_context ctx,
const char *userid,
unsigned long *count,
char ***groupids,
ivadmin_response rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The Security Access Manager user name.

Output

count The number of group names returned.

groupids
An array of pointers to the group names returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns a list of groups in which the specified user is a member.

Command-line equivalent:

```
pdadmin user show-groups user_name
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getminpwdalphas()

This API returns the minimum number of alphabetic characters that are allowed in a password for the specified user account.

Syntax

```
unsigned long ivadmin_user_getminpwdalphas(
ivadmin_context ctx,
const char *userid,
unsigned long *chars,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

Output

chars The minimum number of alphabetic characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the minimum number of alphabetic characters that are allowed in a password for the specified user account.

Command-line equivalent:

```
pdadmin policy get min-password-alphas [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getminpwdlen()

This API returns the minimum password length that is allowed for the specified user account.

Syntax

```
unsigned long ivadmin_user_getminpwdlen(
ivadmin_context ctx,
const char *userid,
unsigned long *length,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

Output

length The minimum password length allowed to be set.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the minimum password length for the specified user account.

Command-line equivalent:

```
pdadmin policy get min-password-length [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getminpwdnonalphas()

This API returns the minimum number of non-alphabetic characters that are allowed in a password for the specified user account.

Syntax

```
unsigned long ivadmin_user_getminpwdnonalphas(
ivadmin_context ctx,
const char *userid,
```

```
unsigned long *chars,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

Output

chars The minimum number of non-alphabetic characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the minimum number of non-alphabetic characters that are allowed in a password for the specified user account.

Command-line equivalent:

```
pdadmin policy get min-password-non-alphas [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getpasswordvalid()

This API returns the password-valid indicator.

Syntax

```
unsigned long ivadmin_user_getpasswordvalid(
ivadmin_ldapuser user
);
```

Parameters

Input

user A pointer to the user structure.

Description

Returns the password valid indicator. Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

Command-line equivalent:

pdadmin user show *user_name*

The password-valid status is part of the information returned by the **pdadmin** command.

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. Indicates that the password is valid.

IVADMIN_FALSE
Defined as 0. Indicates that the password expired.

ivadmin_user_getpwdspaces()

This API returns whether spaces are allowed in passwords for the specified user account.

Syntax

```
unsigned long ivadmin_user_getpwdspaces(
ivadmin_context ctx,
const char *userid,
unsigned long *allowed,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

Output

allowed
The indicator of whether spaces are allowed in passwords.

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns whether spaces are allowed in passwords for the specified user account.

Command-line equivalent:

```
pdadmin policy get password-spaces [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_getsn()

This API returns the user registry surname attribute for the specified user.

Syntax

```
const char* ivadmin_user_getsn(
ivadmin_ldapuser user
);
```

Parameters

Input

user A pointer to the user structure.

Description

Returns the user registry surname attribute for the specified user.

Do not free the character string that is returned. This data is maintained in the `ivadmin_ldapuser` structure.

Command-line equivalent:

```
pdadmin user show user_name
```

The user registry surname for the user is part of the information returned by the **pdadmin** command.

Return values

Returns the user registry surname attribute for the specified user.

User registry difference: The maximum length of the surname attribute varies depending on the user registry that is used. See Appendix B, "User registry differences," on page 293 to determine the maximum length in your environment.

ivadmin_user_getssouser()

This API returns a setting that indicates whether the user account has single sign-on capabilities.

Syntax

```
unsigned long ivadmin_user_getssouser(
ivadmin_ldapuser user
);
```

Parameters

Input

user A pointer to the user structure.

Description

Returns a setting that indicates whether the user account has single sign-on capabilities.

Command-line equivalent:

```
pdadmin user show user_name
```

The single sign-on status for the user is part of the information returned by the **pdadmin** command.

Return values

Returns one of the following values:

IVADMIN_TRUE
Defined as 1. Indicates that the user account is single-sign-on capable.

IVADMIN_FALSE
Defined as 0. Indicates that the user account is not single-sign-on capable.

ivadmin_user_gettodaccess()

This API returns the time-of-day access policy for the specified user.

Syntax

```
unsigned long ivadmin_user_gettodaccess(
ivadmin_context ctx,
const char *userid,
unsigned long *days,
unsigned long *start,
unsigned long *end,
unsigned long *reference,
unsigned long *unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server

userid The user registry user name.

Output

days A bitmap of the days for the time-of-day access policy.

start The minutes after midnight for the start of the time range.

end The minutes after midnight for the end of the time range.

reference
The time zone: Universal Time Coordinated (UTC) or local.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns the time-of-day access policy for the specified user.

Command-line equivalent:

```
pdadmin policy get todaccess -user userID
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_import2()

This API creates a Security Access Manager user by importing an existing user in the user registry.

Syntax

```
unsigned long ivadmin_user_import2(
ivadmin_context ctx,
const char *userid,
const char *dn,
const char *groupid,
unsigned long ssouser,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

dn The user registry distinguished name.

groupid
The initial user registry group to which the user belongs. This value can be `NULL` to indicate no initial group membership.

ssouser
The user uses single sign-on credentials.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates a Security Access Manager user by importing an existing user in the user registry.

Accounts are created as not valid by default. You must use `ivadmin_user_setaccountvalid()` to enable the account.

Groups created in the Active Directory Lightweight Directory Service (AD LDS) user registry must be created in the same AD LDS partition where the Security Access Manager Management Domain information is stored.

Command-line equivalent:

`pdadmin user import [-gsouser] user_name dn`

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_list()

This API lists the names of the Security Access Manager users that match the specified pattern.

Syntax

```
unsigned long ivadmin_user_list(
ivadmin_context ctx,
const char *pattern,
unsigned long maxreturn,
unsigned long *count,
char ***userids,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

pattern
The pattern match for user names. `IVADMIN_ALLPATTERN` indicates all users.

maxreturn
The maximum number to return. `IVADMIN_MAXRETURN` indicates unlimited. This number can be limited by the user registry server so that the maximum returned is really the minimum of the server configuration and this value.

Output

count The number of user names returned.

userids
An array of pointers to the user names returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Lists the names of the Security Access Manager users in the user registry that match the specified pattern. Returns an array of pointers to character strings that contain the user IDs.

The following constants are defined:

```
#define IVADMIN_MAXRETURN 0
#define IVADMIN_ALLPATTERN "*"
```

Command-line equivalent:

```
pdadmin user list pattern max_return
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_listbydn()

This API returns a list of user registry distinguished names whose user registry common name attribute matches the specified pattern.

Syntax

```
unsigned long ivadmin_user_listbydn(
ivadmin_context ctx,
const char *pattern,
unsigned long maxreturn,
unsigned long *count,
char ***dns,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

pattern
The pattern match for user registry common name attribute. IVADMIN_ALLPATTERN indicates all users.

maxreturn
The maximum number to return. IVADMIN_MAXRETURN indicates unlimited. This number can be limited by the user registry server so that the maximum returned is really the minimum of the server configuration and this value.

Output

count The number of user registry distinguished names returned.

dns An array of pointers to the user registry distinguished names returned. You must free the character data referenced by each pointer and also the array of pointers when they are no longer needed.

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Returns a list of user registry distinguished names whose user registry common name attribute matches the pattern specified. Returns an array of pointers to character strings that contain each user distinguished name.

Command-line equivalent:

```
pdadmin user list-dn pattern max_return
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setaccexpdate()

This API sets the account expiration date for the specified user.

Syntax

```
unsigned long ivadmin_user_setaccexpdate(
ivadmin_context ctx,
const char *userid,
unsigned long seconds,
unsigned long unlimited,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

seconds
The date and time of the expiration of specified user account. This value is the number of seconds since 00:00:00 Universal time, 1 January 1970 (same as `time_t`).

unlimited
Prevents a specified user account from expiring and ignores the *seconds* parameter if set to true.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the account expiration date for the specified user.

Command-line equivalent:

```
pdadmin policy set account-expiry-date {unlimited|absolute_time|unset} \
[-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setaccountvalid()

This API enables or disables the specified Security Access Manager user account.

Syntax

```
unsigned long ivadmin_user_setaccountvalid(
ivadmin_context ctx,
const char *userid,
unsigned long valid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

valid A Boolean indicator of account validity.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Enables or disables the specified Security Access Manager user account. Use this function to enable an account after it is created with `ivadmin_user_create3()` or `ivadmin_user_import()`.

The initial value of this attribute is `yes`.

Command-line equivalent:

```
pdadmin user modify user_name account-valid {yes | no}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setdescription()

This API creates or changes the user description.

Syntax

```
unsigned long ivadmin_user_setdescription(
ivadmin_context ctx,
const char *userid,
const char *description,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

description
The new description.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the user description. The description is an arbitrary text string. For example:

```
Xian Arroyo, Credit Dept HCUS
```

Command-line equivalent:

```
pdadmin user modify user_name description description
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setdisabletimeint()

This API sets the time, in seconds, to disable the specified user account if the maximum number of login failures is exceeded.

Syntax

```
unsigned long ivadmin_user_setdisabletimeint(
ivadmin_context ctx,
const char *userid,
unsigned long seconds,
unsigned long disable,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

seconds
The specified number of seconds that the user account is disabled if the maximum number of login failures is exceeded.

disable
The user account that is disabled when the maximum number of login failures is exceeded. Administrator action is required to enable the account.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the time, in seconds, to disable the specified user account if the maximum number of login failures is exceeded.

Command-line equivalent:

```
pdadmin policy set disable-time-interval {number | unset | disable} \
[-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setmaxconcurwebsess()

This API sets the maximum number of concurrent web sessions that are allowed for the specified user account.

Syntax

```
unsigned long ivadmin_user_setmaxconcurwebsess(
ivadmin_context ctx,
const char *userid,
unsigned long sessions,
unsigned long displace,
unsigned long unlimited,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

sessions
The maximum number of concurrent web sessions that are allowed. This parameter is overridden when one of the following parameters is set to true: `unset`, `displace`, or `unlimited`. If nonzero, the parameters `unset`, `displace`, and `unlimited` are set to false. When specified, the value cannot be `0`.

displace
The policy is set to displace if set to true. If set to false, the policy is set as specified. When set to true, this parameter overrides the `sessions` and `unlimited` parameters. This parameter is overridden when parameter `unset` is set to true.

Required value is `IVADMIN_TRUE` or `IVADMIN_FALSE`.

unlimited
The policy is set to unlimited if set to true. If set to false, the policy is set as specified. When set to true, this parameter overrides the `sessions` parameter. This parameter is overridden when parameter `unset` or `displace` is set to true.

Required value is `IVADMIN_TRUE` or `IVADMIN_FALSE`.

unset The policy is ignored and not enforced if set to true. If set to false, the

policy is set as specified. When set to true this parameter overrides the `sessions`, `displace`, and `unlimited` parameters.

Required value is `IVADMIN_TRUE` or `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the maximum number of concurrent web sessions that are allowed for each user account.

Sets concurrent web session policy to be limited to a maximum number of concurrent sessions. The setting can specify unlimited concurrent sessions or session displacement. When set to `displace`, a new web session for a user terminates any existing web session for that user. When set to *number*, the number represents the maximum number of web sessions that can be established. The number cannot be zero.

This policy applies only to certain components of Security Access Manager. A *web session is a user session that is maintained by these Security Access Manager systems: Security Access Manager WebSEAL and Security Access Manager Plug-ins for Web Servers.*

Command-line equivalent:

```
pdadmin policy set max-concurrent-web-sessions {number | displace | unlimited
| unset} [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setmaxlgnfails()

This API sets the maximum number of login failures that are allowed for the specified user account.

Syntax

```
unsigned long ivadmin_user_setmaxlgnfails(
ivadmin_context ctx,
const char *userid,
unsigned long failures,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

failures
The maximum number of login failures allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the maximum number of login failures that are allowed for the specified user account.

Command-line equivalent:

```
pdadmin policy set max-login-failures number | unset [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setmaxpwdage()

This API sets the maximum password age that is allowed for the specified user account.

Syntax

```
unsigned long ivadmin_user_setmaxpwdage(
ivadmin_context ctx,
const char *userid,
unsigned long seconds,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

seconds
The maximum lifetime in seconds before the expiration of the password. A value of zero means the password never expires.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the maximum password age that is allowed for the specified user account.

Command-line equivalent:

```
pdadmin policy set max-password-age {unset | relative_time} [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setmaxpwdrepchars()

This API sets the maximum number of repeated characters that are allowed in a password for the specified user account.

Syntax

```
unsigned long ivadmin_user_setmaxpwdrepchars(
ivadmin_context ctx,
const char *userid,
unsigned long chars,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

chars The maximum number of repeated characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the maximum number of repeated characters that are allowed in a password for the specified user account.

Command-line equivalent:

```
pdadmin policy set max-password-repeated-chars number | unset [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setminpwdalphas()

This API sets the minimum number of alphabetic characters that are allowed in a password for the specified user account.

Syntax

```
unsigned long ivadmin_user_setminpwdalphas(
ivadmin_context ctx,
const char *userid,
unsigned long chars,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

chars The minimum number of alphabetic characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the minimum number of alphabetic characters that are allowed in a password for the specified user account.

Command-line equivalent:

```
pdadmin policy set min-password-alphas {unset | number}[-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setminpwdlen()

This API sets the minimum password length for the specified user account.

Syntax

```
unsigned long ivadmin_user_setminpwdlen(
ivadmin_context ctx,
const char *userid,
unsigned long length,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

length The minimum password length allowed to be set.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the minimum password length for the specified user account.

Command-line equivalent:

```
pdadmin policy set min-password-length {unset | number} [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setminpwdnonalphas()

This API sets the minimum number of non-alphabetic characters that are allowed in a password for the specified user account.

Syntax

```
unsigned long ivadmin_user_setminpwdnonalphas(
ivadmin_context ctx,
const char *userid,
unsigned long chars,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

chars The minimum number of non-alphabetic characters allowed.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the minimum number of non-alphabetic characters that are allowed in a password for the specified user account.

Command-line equivalent:

```
pdadmin policy set min-password-non-alphas {unset | number} [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setpassword()

This API creates or changes the password for the specified user.

Syntax

```
unsigned long ivadmin_user_setpassword(
ivadmin_context ctx,
const char *userid,
const char *pwd,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

pwd The new password.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Creates or changes the password for the specified user.

If the user that is having its password set is the same user that created the security context, *ctx*, no further authorization checks are performed.

Command-line equivalent:

```
pdadmin user modify user_name password password
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setpasswordvalid()

This API enables or disables the expiration of the Security Access Manager account password.

Syntax

```
unsigned long ivadmin_user_setpasswordvalid(
ivadmin_context ctx,
const char *userid,
unsigned long valid,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

valid A setting that indicates whether the password is valid or expired.

Supported values are `IVADMIN_FALSE (expired)` or `IVADMIN_TRUE (valid)`.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Enables or disables the expiration of the Security Access Manager account password. This expiration of the account password forces the user to change the password at the next login attempt.

The initial value of this attribute is **yes**.

Command-line equivalent:

```
pdadmin user modify user_name password-valid {yes | no}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setpwdspaces()

This API specifies whether spaces are allowed in passwords for the specified user account.

Syntax

```
unsigned long ivadmin_user_setpwdspaces(
ivadmin_context ctx,
const char *userid,
unsigned long allowed,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

allowed
A setting that indicates whether spaces are allowed in passwords.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Indicates whether spaces are allowed in passwords for the specified user account.

The initial value of this attribute is unset.

Command-line equivalent:

```
pdadmin policy set password-spaces {yes | no | unset} [-user user_name]
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_setssouser()

This API enables or disables the single sign-on capabilities of a Security Access Manager user.

Syntax

```
unsigned long ivadmin_user_setssouser(
ivadmin_context ctx,
const char *userid,
unsigned long ssouser,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user name.

ssouser
The user can have single sign-on credentials.

Supported values are IVADMIN_TRUE and IVADMIN_FALSE.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Enables or disables the single sign-on capabilities of a Security Access Manager user.

The initial value of this attribute is `yes`.

Command-line equivalent:

```
pdadmin user modify user_name gsouser {yes | no}
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

ivadmin_user_settodaccess()

This API sets the time-of-day access policy for the specified user.

Syntax

```
unsigned long ivadmin_user_settodaccess(
ivadmin_context ctx,
const char *userid,
unsigned long days,
unsigned long start,
unsigned long end,
unsigned long reference,
unsigned long unset,
ivadmin_response *rsp
);
```

Parameters

Input

ctx The context to use when communicating with the policy server.

userid The user registry user name.

days A bitmap of the days for the time-of-day access policy.

start The minutes after midnight for the start of the time range.

end The minutes after midnight for the end of the time range.

reference
The time zone: Universal Coordinated Time (UTC) or local.

unset The policy is ignored and not enforced if set to true. If set to false, the policy is set as specified.

Supported values are `IVADMIN_TRUE` and `IVADMIN_FALSE`.

Note: When setting a password policy, you provide a list of days, start time, and end time. The start time and end time apply to each day on the list. If the specified start time is later than the specified end time, then the access is allowed until the specified end time is reached on the next day.

Output

rsp The response object. Indicates the success or failure of the function. Contains error information. Free this object when it is no longer needed.

Description

Sets the time-of-day access policy for the specified user.

Command-line equivalent:

```
pdadmin policy set todaccess todaccess_string -user userID
```

Return values

Returns the following values:

IVADMIN_TRUE
Defined as 1. The function was successful.

IVADMIN_FALSE
Defined as 0. The function encountered an error.

Appendix A. Deprecated APIs and constants

This section describes APIs and constants that were deprecated in Security Access Manager, version 8.0. The APIs and constants might be deprecated in previous versions of this product or in SecureWay Policy Directory.

APIs deprecated in version 8.0

This section lists APIs that are deprecated in Security Access Manager, version 8.0.

The `ivadmin_deprecated.h` header file contains the prototypes and definitions for these deprecated APIs. Avoid including this header file because the symbols it declares are not supported. Instead, change existing applications to use any replacement APIs that are listed in Table 36.

Table 36. APIs deprecated in Security Access Manager version 8.0

Deprecated API	Replacement API
`ivadmin_pop_getipauth2()`	`ivadmin_pop_getipauth3()`

APIs deprecated in previous releases

This section identifies APIs that were deprecated in previous versions of this product and Tivoli SecureWay Policy Director.

The APIs listed in Table 37 were deprecated in previous versions of this product or in previous versions of Tivoli SecureWay Policy Director.

Table 37. APIs deprecated in previous versions of Access Manager or Tivoli SecureWay Policy Director

Deprecated API	Replacement API
`ivadmin_cfg_addreplica()`	`ivadmin_cfg_addreplica2()`
`ivadmin_cfg_chgreplica()`	`ivadmin_cfg_chgreplica2()`
`ivadmin_cfg_configureserver2()`	`ivadmin_cfg_configureserver3()`
`ivadmin_cfg_rmvreplica()`	`ivadmin_cfg_rmvreplica2()`
`ivadmin_cfg_setapplicationcert()`	`ivadmin_cfg_setapplicationcert2()`
`ivadmin_cfg_setkeyringpwd()`	`ivadmin_cfg_setkeyringpwd2()`
`ivadmin_cfg_setlistening()`	`ivadmin_cfg_setlistening2()`
`ivadmin_cfg_setport()`	`ivadmin_cfg_setport2()`
`ivadmin_cfg_setssltimeout()`	`ivadmin_cfg_setssltimeout2()`
`ivadmin_context_create`	`ivadmin_context_create3`
`ivadmin_context_create2()`	`ivadmin_context_create3()`
`ivadmin_context_createdefault()`	`ivadmin_context_createdefault2()`
`ivadmin_pop_getanyothernw()`	`ivadmin_pop_getanyothernw2()`
`ivadmin_pop_getipauth()`	`ivadmin_pop_getipauth2()`
`ivadmin_pop_removeipauth()`	`ivadmin_pop_removeipauth2()`
`ivadmin_pop_setanyothernw()`	`ivadmin_pop_setanyothernw2()`

Table 37. APIs deprecated in previous versions of Access Manager or Tivoli SecureWay Policy Director (continued)

Deprecated API	Replacement API
ivadmin_pop_setanyothernw_forbidden()	ivadmin_pop_setanyothernw_forbidden2()
ivadmin_pop_setipauth()	ivadmin_pop_setipauth2()
ivadmin_pop_setipauth_forbidden()	ivadmin_pop_setipauth_forbidden2()
ivadmin_protobj_get2()	ivadmin_protobj_get3()
ivadmin_protobj_getacl()	ivadmin_protobj_getaclid ivadmin_acl_get
ivadmin_protobj_getauthzrule()	ivadmin_protobj_getauthzruleid() ivadmin_authzrule_get()
ivadmin_protobj_getpop()	ivadmin_protobj_getpopid() ivadmin_pop_get()
ivadmin_protobj_setname()	None
ivadmin_cfg_configureserver	ivadmin_cfg_configureserver3
ivadmin_group_addmember	ivadmin_group_addmembers
ivadmin_group_removemember	ivadmin_group_removemembers
ivadmin_user_create2	ivadmin_user_create3
ivadmin_user_getauthmech	None
ivadmin_user_setauthmech	None
ivadmin_group_create	ivadmin_group_create3
ivadmin_group_delete	ivadmin_group_delete2
ivadmin_group_import	ivadmin_group_import2
ivadmin_protobj_get	ivadmin_protobj_get3
ivadmin_protobj_list2	ivadmin_protobj_list3
ivadmin_user_create	ivadmin_user_create3
ivadmin_user_delete	ivadmin_user_delete2
ivadmin_user_import	ivadmin_user_import2
ivadmin_cfg_renewservercert	ivadmin_cfg_renewservercert2
ivadmin_cfg_setport	ivadmin_cfg_setport2
ivadmin_cfg_setlistening	ivadmin_cfg_setlistening2
ivadmin_cfg_setkeyringpwd	ivadmin_cfg_setkeyringpwd2
ivadmin_cfg_setssltimeout	ivadmin_cfg_setssltimeout2
ivadmin_cfg_setapplicationcert	ivadmin_cfg_setapplicationcert2
ivadmin_cfg_addreplica	ivadmin_cfg_addreplica2
ivadmin_cfg_chgreplica	ivadmin_cfg_chgreplica2
ivadmin_cfg_rmvreplica	ivadmin_cfg_rmvreplica2
ivadmin_pop_getanyothernw	ivadmin_pop_getanyothernw2
ivadmin_pop_setanyothernw	ivadmin_pop_setanyothernw2
ivadmin_pop_setanyothernw_forbidden	ivadmin_pop_setanyothernw_forbidden2
ivadmin_pop_getipauth	ivadmin_pop_getipauth3

Table 37. APIs deprecated in previous versions of Access Manager or Tivoli SecureWay Policy Director (continued)

Deprecated API	Replacement API
ivadmin_pop_getipauth2	ivadmin_pop_getipauth3
ivadmin_pop_setipauth	ivadmin_pop_setipauth2
ivadmin_pop_setipauth_forbidden	ivadmin_pop_setipauth_forbidden2
ivadmin_pop_removeipauth	ivadmin_pop_removeipauth2

Deprecated constants

This section identifies constants that were deprecated in previous versions of this product and Tivoli SecureWay Policy Director.

The following constants were deprecated in previous versions of this product and Tivoli SecureWay Policy Director.

- IVADMIN_USER_DCEAUTHMETH
- IVADMIN_USER_LDAPAUTHMETH
- IVADMIN_PROTOBJ_TYPE_UNKNOWN

The ivadmin_deprecated.h header file contains the definitions for these deprecated constants. Avoid including this header file because the symbols it declares are not supported.

Appendix B. User registry differences

Each user registry presents unique concerns when integrated with Security Access Manager.

This release of Security Access Manager supports LDAP and URAF user registries.

Security Access Manager supports the following LDAP user registries:

- Tivoli Directory Server
- IBM z/OS® Security Server LDAP Server
- Microsoft Active Directory 2008
- Microsoft Active Directory Lightweight Directory Services (AD LDS) (Windows 2008)
- Novell eDirectory, versions 8.7 and 8.8
- Oracle Directory Server 11g Enterprise Release 1 11.1.1.5.0
- Sun Java System Directory Server, version 7.0

Security Access Manager supports the following URAF user registries:

- Microsoft Active Directory Server 2008

General concerns

This section describes concerns associated with all supported user registries.

The following concerns are specific to all the supported user registries:

- Avoid with forward slash (/) character when defining the names for users and groups when that name is defined with distinguished names strings. Each user registry treats this character differently.
- Avoid the use of leading and trailing blanks in user and group names. Each user registry treats blanks differently.

LDAP concerns

This section describes concerns that are specific to all supported LDAP user registries.

The following concerns are specific to all the supported LDAP user registries:

- There are no configuration steps needed in Security Access Manager to make it support a policy of LDAP. Security Access Manager does not assume the existence or non-existence of an LDAP password policy at all. Security Access Manager enforces its own password policy first. Security Access Manager attempts to update a password in LDAP only when the provided password passes the password policy check of Security Access Manager.

  Then Security Access Manager tries to accommodate the password policy of LDAP with the return code that it receives from LDAP during a password-related update.

  If Security Access Manager can map this return code with the corresponding Security Access Manager error code, it returns an error message.

- To take advantage of the multi-domain support in Security Access Manager, you must use an LDAP user registry. When you use a URAF user registry, only a single Security Access Manager domain is supported.
- When using an LDAP user registry, the capability to own global sign-on credentials must be explicitly granted to a user. After this capability is granted, it can then be removed. Conversely, users that are created in a URAF user registry are automatically given this capability. This capability cannot be removed.
- Leading and trailing blanks in user names and group names are ignored when using an LDAP user registry in a Security Access Manager secure domain. To ensure consistent processing regardless of the user registry, define user names and group names without leading or trailing blanks.
- Attempting to add a single duplicate user to a group does not produce an error when using an LDAP user registry.
- The Security Access Manager authorization API provides a credential attribute entitlements service. This service is used to retrieve user attributes from a user registry. When this service is used with an LDAP user registry, the retrieved attributes can be string data or binary data. However, when used with a URAF user registry, the retrieved attributes can be string data, binary data, or integer data.

Sun Java System Directory Server concerns

This section explains how to modify the default look-through limit for Sun Java System Directory Server.

About this task

A concern specific to Sun Java System Directory Server is that the user registry might contain more entries than the defined look-through limit. The directory server might return the following status that Security Access Manager treats as an error:

```
LDAP_ADMINLIMIT_EXCEEDED
```

When the directory server is installed, the default value is 5000. You can modify this value.

Procedure

1. On the Sun Java System Directory Server Console, select the **Configuration** tab.
2. Expand the **Data** entry.
3. Select **Database Settings**.
4. Select the **LDBM Plug-in Settings** tab.
5. In the **Look-through Limit** field, type the maximum number of entries that you want the server to check in response to the search. Alternatively, type `-1` to define no maximum limit. If you bind the directory as the Directory Manager, the look-through limit is unlimited and overrides any settings specified in this field.

Microsoft Active Directory Lightweight Directory Server (AD LDS) concerns

This section describes concerns specific to Microsoft Active Directory Lightweight Directory Server (AD LDS).

The following concerns are specific to AD LDS:

- Use the Security Access Manager configuration to select between a standard or minimal data model for the user registry. Because AD LDS allows only a single naming attribute to be used when creating LDAP objects, AD LDS requires the minimal data model. Irrespective of which data model is chosen during Security Access Manager configuration, Security Access Manager always uses the minimal data model when AD LDS is selected as the user registry.
- The common name (cn) value in AD LDS must be single-valued. The value specified for the `cn` attribute must be the same value used for the distinguished name when a user or group is created and `cn` is used as the naming attribute in the dn. For example, the following command to create a user would *not* be allowed:

  ```
  pdadmin user create user1 cn=user1,o=ibm,c=us fred user1 password1
  ```

  In the example, the `cn` value, `fred`, is different from the `cn` naming attribute in the dn, `user1`.

URAF concerns

This section describes concerns specific to all supported URAF user registries.

The following concerns are specific to all the supported URAF user registries:

- When using a URAF user registry, only a single Security Access Manager domain is supported. To take advantage of the Security Access Manager multi-domain support, use an LDAP user registry.
- Users created in a URAF user registry are automatically given the capability to own global sign-on credentials. This capability cannot be removed. When using an LDAP user registry, this capability must be explicitly granted. After this capability is granted, it can be removed later.
- The Security Access Manager authorization API provides a credential attribute entitlements service. This service is used to retrieve user attributes from a user registry. When this service is used with a URAF user registry, the retrieved attributes can be string data, binary data, or integer data. However, when used with an LDAP user registry, the retrieved attributes can be only string data or binary data.

Microsoft Active Directory Server concerns

This section describes concerns specific to Microsoft Active Directory Server.

In addition to the general URAF-specific concerns, the following concerns are specific to Microsoft Active Directory Server:

- Users created in Active Directory might have an associated primary group. The Active Directory default primary group is Domain Users.

  But Active Directory does not add the primary group information to the user `memberOf` or the group `member` attribute. When Security Access Manager queries for a list of members of a group, the result does not include any members for whom the group is the primary group. When Security Access Manager queries for all the groups to which a user belongs, the query result does not display the primary group of the user.

  For this reason, avoid the use of a Security Access Manager group as the Active Directory primary group for Security Access Manager users.
- Security Access Manager does not support cross domain group membership or universal groups. Security Access Manager does not support importing these types of groups.

- When Security Access Manager imports a dynamic group, the `ivacld-servers` and `remote-acl-users` groups apply read permission on each authorization store to which the dynamic group belongs.

  Read permission enables Security Access Manager blade servers, such as WebSEAL, to have the read permission to the registry authorization store. The permission provides the blade server with the ability to read dynamic group data, such as group membership for building Security Access Manager credentials. Manually removing this read permission while Security Access Manager is configured to the Active Directory registry results in adverse behavior, such as inaccurate group membership.
- The option to change a user password with LDAP APIs can be enabled in an environment in this configuration:
  - Security Access Manager is configured to use the Active Directory user registry.
  - Security Access Manager blade servers use LDAP APIs to communicate with the Active Directory server.

  In this case, Security Access Manager must be configured with Secure Socket Layer (SSL) to allow connections between the LDAP client and the Active Directory server. The Active Directory environment must also be enabled to accept LDAP connections over Secure Socket Layer (SSL).
- You might use an Active Directory user registry in a Security Access Manager configuration with blade servers that use LDAP APIs to communicate with the Active Directory server. Security Access Manager supports user password change requests with either the policy server or LDAP APIs. Requests to change user passwords with LDAP APIs do not require the policy server to running.

  The use of LDAP APIs to communicate with the Active Directory Server for blade servers is a multi-platform support. Blade servers can be installed on computers that are not clients of the same domain as the policy server. In this configuration, the policy server must be installed and configured on a Windows operating system.
- When using an Active Directory user registry, each user name and each group name in a domain must be unique. User and group short name values are stored in the `sAMAccountName` attribute of Active Directory user objects and group objects. Active Directory user objects and group objects both have the `sAMAccountName` attribute as one of their attributes. Microsoft requires that the `sAMAccountName` attributes be unique within an Active Directory domain.
- When using a multi-domain Active Directory user registry, multiple users and groups can be defined with the same short name. The users and groups must be in different domains. The full name of the user or group, including the domain suffix, must always be specified to Security Access Manager.
- Leading and trailing blanks in user names and group names are ignored with Microsoft Active Directory Server as the user registry in a Security Access Manager secure domain. To ensure consistent processing, regardless of the user registry, define user names and group names without leading or trailing blanks.
- Security Access Manager supports the use of an email address or other format of the `userPrincipalName` attribute of the Active Directory registry user object as a Security Access Manager user identity. When this optional enhancement is enabled, both the default and the email address or other format of the `userPrincipalName` can co-exist in the Security Access Manager environment.

  The default format of the `userPrincipalName` registry attribute is `user_id@`*`domain_suffix`*, where *domain_suffix* is the Active Directory domain where the user identity is created.

For example, johndoe@example.com is the value of the userPrincipalName; **example.com** is the Active Directory domain where the user identity is created.

The Security Access Manager user identity corresponding to the registry user in this example is either **johndoe@example.com** or **johndoe**. The user identity depends on whether Security Access Manager is configured to use Active Directory with multiple domains or a single domain.

The other format of the userPrincipalName attribute is user_id@any_suffix. In this format, any_suffix can be any domain (Active Directory or non-Active Directory) other than the Active Directory domain in which the user identity is created.

For example, if the registry user johndoe@other_domain.com is created in Active Directory example.com, and the registry user johndoe@example.com is created in Active Directory domain child_domain.example.com. Both of these users can be Security Access Manager users, and their user identities are johndoe@other_domain.com and johndoe@example.com.

An alternative user principal name (UPN) support must be enabled in all Security Access Manager runtime environments. Enablement ensures that Security Access Manager user identities work properly with alternative UPNs.

After the use of an alternative UPN format as Access Manager user identity is enabled, it cannot be reversed without breaking Security Access Manager functionalities.

- Users and groups can be created with names that use a distinguished name string that contains a forward slash (/) character. However, subsequent operations on the object might fail. Some Active Directory functions interpret the forward slash character as a separator between the object name and the host name. To avoid the problem, do not use a forward slash character to define the user.

Length of names

This section describes the maximum lengths of the names associated with Security Access Manager.

The maximum lengths of various names that are associated with Security Access Manager vary depending on the user registry that is being used. See Table 38 for a comparison of the maximum lengths that are allowed and the maximum length to use to ensure compatibility with all the user registries that are supported by Security Access Manager.

Table 38. Maximum lengths for names by user registry and the optimal length across user registries

Name	IBM Tivoli Directory Server	IBM z/OS Security Server	Novell eDirectory Server	Sun Java System Directory Server	Microsoft Active Directory Server	Active Directory Lightweight Directory Server (AD LDS)	Optimal length
First name (LDAP CN)	256	256	64	256	64	64	64
Middle name	128	128	128	128	64	65535	64
Last name (surname)	128	128	128	128	64	960	64
Registry UID (LDAP DN)	1024	1024	1024	1024	2048	255	1024

Table 38. Maximum lengths for names by user registry and the optimal length across user registries (continued)

Name	IBM Tivoli Directory Server	IBM z/OS Security Server	Novell eDirectory Server	Sun Java System Directory Server	Microsoft Active Directory Server	Active Directory Lightweight Directory Server (AD LDS)	Optimal length
Security Access Manager user identity	256	256	256	256	64	196 - domain_ name_length	64
User password	unlimited	unlimited	unlimited	unlimited	256	unlimited	128
User description	1024	1024	1024	1024			
Group name	256	256	256	256	64	196 - domain_ name_length	64
Group description	1024	1024	1024	1024			
Single sign-on resource name	240	240	240	240	60	256	240
Single sign-on resource description	1024	1024	1024	1024			
Single sign-on user ID	240	240	240	240	60	256	240
Single sign-on password	unlimited	unlimited	unlimited	unlimited	256	unlimited	256
Single sign-on group name	240	240	240	240	60	256	240
Single sign-on group description	1024	1024	1024	1024			
Action name	1	1	1	1			
Action description, action type	unlimited	unlimited	unlimited	unlimited			
Object name, object description	unlimited	unlimited	unlimited	unlimited			
Object space name, object space description	unlimited	unlimited	unlimited	unlimited			

Table 38. Maximum lengths for names by user registry and the optimal length across user registries (continued)

Name	IBM Tivoli Directory Server	IBM z/OS Security Server	Novell eDirectory Server	Sun Java System Directory Server	Microsoft Active Directory Server	Active Directory Lightweight Directory Server (AD LDS)	Optimal length
ACL name, ACL descriptions	unlimited	unlimited	unlimited	unlimited			
POP name, POP description	unlimited	unlimited	unlimited	unlimited			

Although the maximum length of an Active Directory distinguished name (registry UID) is 2048, the maximum length of each relative distinguished name (RDN®) is 64.

You might configure Security Access Manager to use multiple Active Directory domains. In this case, the maximum length of the user identity and group name does not include the domain suffix. With multiple domains, the format of a user identity is *user_id@domain_suffix*.

The maximum length of 64 applies only to the `user_id` portion. You might use an email address or other format for the Security Access Manager user identity in the Active Directory. In this case, the maximum name length remains the same, but includes the suffix.

Although the lengths of some names can be of unlimited, excessive lengths can result in policy that is difficult to manage. The condition might result in poor system performance. Choose maximum values that are logical for your environment.

Appendix C. Administration API equivalents

This appendix shows the mapping that exists between the administration C API functions, the administration Java classes and methods, the **pdadmin** commands, and Web Portal Manager.

In some cases, a given operation can be performed in different ways. In some cases two or more method calls might be necessary to achieve the same effect as a single C API function.

See the following books for more information:

- Information about the administration C API can be found in this document.
- Information about the administration Java classes and methods can be found in the Javadoc information and the *IBM Security Access Manager for Web: Administration Java Classes Developer Reference*.
- Information about the **pdadmin** commands can be found in the *IBM Security Access Manager for Web: Command Reference*.
- Information on Web Portal Manager can be found in its online help and in the *IBM Security Access Manager for Web: Administration Guide*.

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_acl_attrdelkey()	**PDAcl.deleteAttribute** *PDAcl object*.**deleteAttribute**	`pdadmin acl modify` *`acl_name`* `delete attribute` *`attribute_name`*	**ACL** > **Search ACL** > click ACL name > **Extended Attribute** tab > select attributes > **Delete**
ivadmin_acl_attrdelval()	**PDAcl.deleteAttributeValue** *PDAcl object*.**deleteAttributeValue**	`pdadmin acl modify` *`acl_name`* `delete attribute` *`attribute_name attribute_value`*	Not supported
ivadmin_acl_attrget()	*PDAcl object*.**getAttributeValues**	`pdadmin acl show` *`acl_name`* `attribute` *`attribute_name`*	**ACL** > **Search ACL** > click ACL name > **Extended Attribute** tab
ivadmin_acl_attrlist()	*PDAcl object*.**getAttributeNames**	`pdadmin acl list` *`acl_name`* `attribute`	**ACL** > **Search ACL** > click ACL name > **Extended Attribute** tab
ivadmin_acl_attrput()	**PDAcl.setAttributeValue** *PDAcl object*.**setAttributeValue**	`pdadmin acl modify` *`acl_name`* `set attribute` *`attribute_name attribute_value`*	**ACL** > **Search ACL** > click ACL name > **Extended Attribute** tab > **Create** > fill in form > **Apply**
ivadmin_acl_create()	**PDAcl.createAcl**	`pdadmin acl create` *`acl_name`*	**ACL** > **Create ACL** > fill in form > **Create**
ivadmin_acl_delete()	**PDAcl.deleteAcl**	`pdadmin acl delete` *`acl_name`*	**ACL** > **Search ACL** > select ACL names > **Delete**
ivadmin_acl_get()	**PDAcl** constructor	`pdadmin acl show` *`acl_name`*	**ACL** > **Search ACL** > click ACL name
ivadmin_acl_getanyother()	*PDAcl object*.**getPDAclEntryAnyOther**	`pdadmin acl show any-other`	**ACL** > **Search ACL** > click **any-other**
ivadmin_acl_getdescription()	*PDAcl object*.**getDescription**	`pdadmin acl show` *`acl_name`*	**ACL** > **Search ACL** > click ACL name
ivadmin_acl_getgroup()	*PDAcl object*.**getPDAclEntriesGroup**	`pdadmin acl show` *`acl_name`*	**ACL** > **Search ACL** > click ACL name
ivadmin_acl_getid()	*PDAcl object*.**getId**	`pdadmin acl show` *`acl_name`*	**ACL** > **Search ACL** > click ACL name

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_acl_getunauth()	*PDAcl object*.**getPDAclEntryUnAuth**	`pdadmin acl show acl_name`	**ACL** > **Search ACL** > click ACL name
ivadmin_acl_getuser()	*PDAcl object*.**getPDAclEntriesUser**	`pdadmin acl show acl_name`	**ACL** > **Search ACL** > click ACL name
ivadmin_acl_list()	**PDAcl.listAcls**	`pdadmin acl list`	**ACL** > **Search ACL**
ivadmin_acl_list2()	**PDAcl.listAcls**	`pdadmin acl list [<pattern> <max-return>]`	**ACL** > **Search ACL**
ivadmin_acl_listgroups()	*PDAcl object*.**getPDAclEntriesGroup**	`pdadmin acl show acl_name`	**ACL** > **Search ACL** > click ACL name
ivadmin_acl_listusers()	*PDAcl object*.**getPDAclEntriesUser**	`pdadmin acl show acl_name`	**ACL** > **Search ACL** > click ACL name
ivadmin_acl_removeanyother()	**PDAcl.removePDAclEntryAnyOther** *PDAcl object*.**removePDAclEntry AnyOther**	`pdadmin acl modify acl_name remove any-other`	**ACL** > **Search ACL** > click ACL name > select **Any-other** > **Delete**
ivadmin_acl_removegroup()	**PDAcl.removePDAclEntryGroup** *PDAcl object*.**removePDAclEntryGroup**	`pdadmin acl modify acl_name remove group group_name`	**ACL** > **Search ACL** > click ACL name > select group name > **Delete**
ivadmin_acl_removeunauth()	**PDAcl.removePDAclEntryUnAuth** *PDAcl object*.**removePDAclEntryUnAuth**	`pdadmin acl modify acl_name remove unauthenticated`	**ACL** > **Search ACL** > click ACL name > select **Unauthenticated** > **Delete**
ivadmin_acl_removeuser()	**PDAcl.removePDAclEntryUser** *PDAcl object*.**removePDAclEntryUser**	`pdadmin acl modify acl_name remove user user_name`	**ACL** > **Search ACL** > click ACL name > select user name > **Delete**
ivadmin_acl_setanyother()	**PDAcl.setPDAclEntryAnyOther** *PDAcl object*.**setPDAclEntryAnyOther**	`pdadmin acl modify acl_name set any-other permissions`	**ACL** > **Search ACL** > click ACL name > select **Any-other** > **Create** > select permissions > **Apply**
ivadmin_acl_setdescription()	**PDAcl.setDescription** *PDAcl object*.**setDescription**	`pdadmin acl modify acl_name description description`	**ACL** > **Search ACL** > click ACL name > modify description > **Set**
ivadmin_acl_setgroup()	**PDAcl.setPDAclEntryGroup** *PDAcl object*.**setPDAclEntryGroup**	`pdadmin acl modify acl_name set group group_name permissions`	**ACL** > **Search ACL** > click ACL name > **Create** > select **Group**> specify group name > select permissions > **Apply**
ivadmin_acl_setunauth()	**PDAcl.setPDAclEntryUnAuth** *PDAcl object*.**setPDAclEntryUnAuth**	`pdadmin acl modify acl_name set unauthenticated permissions`	**ACL** > **Search ACL** > click ACL name > **Create** > select **Unauthenticated** > select permissions > **Apply**
ivadmin_acl_setuser()	**PDAcl.setPDAclEntryUser** *PDAcl object*.**setPDAclEntryUser**	`pdadmin acl modify acl_name set user user_name permissions`	**ACL** > **Search ACL** > click ACL name > **Create** > select **User** > specify user name > select permissions > **Apply**
ivadmin_action_create()	**PDAction.createAction**	`pdadmin action create name description action_type`	**ACL** > **List Action Groups** > click `primary` action group > **Create** > fill in form > **Create**
ivadmin_action_create_in_group()	**PDAction.createAction**	`pdadmin action create name description action_type action_group_name`	**ACL** > **List Action Groups** > click action group > **Create** > fill in form > **Create**
ivadmin_action_delete()	**PDAction.deleteAction**	`pdadmin action delete name`	**ACL** > **List Action Groups** > select `primary` action group > select actions > **Delete**
ivadmin_action_delete_from_group()	**PDAction.deleteAction**	`pdadmin action delete name action_group_name`	**ACL** > **List Action Groups** > select action group > select actions > **Delete**

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_action_getdescription()	*PDAction object*.**getDescription**	`pdadmin action list`	**ACL** > **List Action Groups** > click primary action group
ivadmin_action_getid()	*PDAction object*.**getId**	`pdadmin action list`	**ACL** > **List Action Groups** > click primary action group
ivadmin_action_gettype()	*PDAction object*.**getType**	`pdadmin action list`	**ACL** > **List Action Groups** > click primary action group
ivadmin_action_group_create()	**PDActionGroup.createActionGroup**	`pdadmin action group create` *action_group_name*	**ACL** > **Create Action Group** > type group name > **Create**
ivadmin_action_group_delete()	**PDActionGroup.deleteActionGroup**	`pdadmin action group delete` *action_group_name*	**ACL** > **List Action Groups** > select action groups > **Delete**
ivadmin_action_group_list()	**PDActionGroup.listActionGroups**	`pdadmin action group list`	**ACL** > **List Action Groups**
ivadmin_action_list()	**PDAction.listActions**	`pdadmin action list`	**ACL** > **List Action Groups** > click `primary` action group
ivadmin_action_list_in_group()	**PDAction.listActions**	`pdadmin action list` *action_group_name*	**ACL** > **List Action Groups** > click action group
ivadmin_authzrule_create()	**PDAuthzRule.createAuthzRule**	`pdadmin authzrule create` *ruleid* `–rulefile {`*filename* | *ruletext*`} [–desc` *description*`] [–failreason` *failreason*`]`	**AuthzRule** > **Create AuthzRule** > fill in form > **Create**
ivadmin_authzrule_delete()	**PDAuthzRule.deleteAuthzRule**	`pdadmin authzrule delete` *ruleid*	**AuthzRule** > **List AuthzRule** > select authorization rule name > **Delete**
ivadmin_authzrule_get()	**PDAuthzRule** constructor	`pdadmin authzrule show` *ruleid*	**AuthzRule** > **List AuthzRule** > click authorization rule name
ivadmin_authzrule_getdescription()	*PDAuthzRule object*.**getDescription**	`pdadmin authzrule show` *ruleid*	**AuthzRule** > **List AuthzRule** > click authorization rule name
ivadmin_authzrule_getfailreason()	*PDAuthzRule object*.**getFailReason**	`pdadmin authzrule show` *ruleid*	**AuthzRule** > **List AuthzRule** > click authorization rule name
ivadmin_authzrule_getid()	*PDAuthzRule object*.**getID**	`pdadmin authzrule show` *ruleid*	**AuthzRule** > **List AuthzRule** > click authorization rule name
ivadmin_authzrule_getruletext()	*PDAuthzRule object*.**getRuleText**	`pdadmin authzrule show` *ruleid*	**AuthzRule** > **List AuthzRule** > click authorization rule name
ivadmin_authzrule_list()	**PDAuthzRule.listAuthzRules**	`pdadmin authzrule list`	**AuthzRule** > **List AuthzRule**
ivadmin_authzrule_setdescription()	**PDAuthzRule.setDescription** *PDAuthzRule object*.**setDescription**	`pdadmin authzrule modify` *ruleid* `description` *description*	**AuthzRule** > **List AuthzRule** > click authorization rule name > **General** tab > modify fields > **Apply**
ivadmin_authzrule_setfailreason()	**PDAuthzRule.setFailReason** *PDAuthzRule object*.**setFailReason**	`pdadmin authzrule modify` *ruleid* `failreason` *failreason*	**AuthzRule** > **List AuthzRule** > click authorization rule name > **General** tab > modify fields > **Apply**
ivadmin_authzrule_setruletext()	**PDAuthzRule.setRuleText** *PDAuthzRule object*.**setRuleText**	`pdadmin authzrule modify` *ruleid* `–rulefile {`*filename* | *ruletext*`}`	**AuthzRule** > **List AuthzRule** > click authorization rule name > modify fields > **Apply**

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_cfg_addreplica2()	**PDAppSvrConfig.addPDServer**	`svrsslcfg -add_replica -f cfg_file -h host_name [-p port] [-k rank]`	Not supported.
ivadmin_cfg_chgreplica2()	**PDAppSvrConfig.changePDServer**	`svrsslcfg -chg_replica -f cfg_file -h host_name [-p port] [-k rank]`	Not supported.
ivadmin_cfg_configureserver3()	**PDAppSvrConfig.configureAppSvr**	`svrsslcfg -config -f cfg_file -d kdb_dir_name -n server_name ...`	Not supported.
ivadmin_cfg_getvalue()	Not supported.	pdadmin config show *config_file stanza key*	Not supported.
ivadmin_cfg_removevalue()	Not supported.	pdadmin config modify keyvalue remove *config_file stanza key* [*value*]	Not supported.
ivadmin_cfg_renewservercert()	**PDAppSvrConfig.replaceAppSvrCert**	`svrsslcfg -chgcert -f cfg_file -n server_name [-A admin_ID] -P admin_pwd`	Not supported.
ivadmin_cfg_rmvreplica2()	**PDAppSvrConfig.removePDServer**	`svrsslcfg -rmv_replica -f cfg_file -h host_name [-p port] [-k rank]`	Not supported.
ivadmin_cfg_setapplicationcert2()	Not supported.	`svrsslcfg -modify -f cfg_file [-t timeout] [-C cert_file] [-l listening_mode]`	Not supported.
ivadmin_cfg_setkeyringpwd2()	Not applicable.	`svrsslcfg -chgpwd -f cfg_file -n server_name [-A admin_ID] [-P admin_pwd]`	Not supported.
ivadmin_cfg_setlistening2()	**PDAppSvrConfig.setAppSvrListening**	`svrsslcfg -f cfg_file -modify -l yes`	Not supported.
ivadmin_cfg_setport2()	**PDAppSvrConfig.setAppSvrPort**	`svrsslcfg -config -f cfg_file -d kdb_dir_name -n server_name ...`	Not supported.
ivadmin_cfg_setssltimeout2()	Not supported.	`svrsslcfg -modify -f cfg_file -t timeout [-C cert_file] [-l listening_mode]`	Not supported.
ivadmin_cfg_setsvrpwd()	Not supported.	pdadmin config modify svrpassword *config_file password*	Not supported.
ivadmin_cfg_setvalue()	Not supported.	pdadmin config modify keyvalue { set | append }[–obfuscate] *config_file stanza key* [*value*]	Not supported.
ivadmin_cfg_unconfigureserver()	**PDAppSvrConfig.unconfigureAppSvr**	`svrsslcfg -unconfig -f cfg_file -n server_name [-A admin_ID] -P admin_pwd`	Not supported.
ivadmin_context_cleardelcred()	*PDContext object*.**clearDelegatedCred**	Not applicable.	Not applicable.
ivadmin_context_create3()	**PDContext** constructor	Not applicable.	Not applicable.
ivadmin_context_createdefault2()	**PDContext** constructor	Not applicable.	Not applicable.
ivadmin_context_createlocal()	Not supported.	Not applicable.	Not applicable.
ivadmin_context_delete()	*PDContext object*.**close**	Not applicable.	Not applicable.
ivadmin_context_ domainismanagement()	*PDContext object*.**domainIsManagement**	`pdadmin context show`	Not supported.
ivadmin_context_getaccexpdate()	*PDPolicy object*.**getAcctExpDate**	`pdadmin policy get account-expiry-date`	**User** > **Show Global User Policy** > view `Account Expiration Date` section
ivadmin_context_getcodeset()	*PDContext object*.**getLocale**	Not applicable.	Not applicable.

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_context_getdisabletimeint()	*PDPolicy object*.**getAcctDisableTimeInterval**	`pdadmin policy get disable-time-interval`	**User** > **Show Global User Policy** > view `Disable Time Interval` section
ivadmin_context_getdomainid()	*PDContext object*.**getDomainid**	`pdadmin context show`	Not supported.
Not supported.	Not supported.	`pdadmin errtext` *error_number*	Not supported.
Not supported.	Not supported.	`pdadmin exit`	Not supported.
ivadmin_context_getmax concurwebsess()	**PDPolicy.getMaxconcurrentWebSessions** *PDPolicy object*.**getMaxconcurrent WebSessions**	`pdadmin policy get max-concurrent-web-sessions`	**User** > **Show Global User Policy** > view `Max Concurrent Web Sessions` section
ivadmin_context_getmaxlgnfails()	*PDPolicy object*.**getMaxFailedLogins**	`pdadmin policy get max-login-failures`	**User** > **Show Global User Policy** > view `Max Login Failures` section
ivadmin_context_getmaxpwdage()	*PDPolicy object*.**getMaxPwdAge**	`pdadmin policy get max-password-age`	**User** > **Show Global User Policy** > view `Max Password Age` section
ivadmin_context_getmaxpwdrepchars()	*PDPolicy object*.**getMaxPwdRepChars**	`pdadmin policy get max-password-repeated-chars`	**User** > **Show Global User Policy** > view `Max Password Repeated Characters` section
ivadmin_context_getmgmtdomainid()	**PDDomain.getMgmtDomainName**	`pdadmin login –m`	Initial login.
ivadmin_context_getmgmtsvrhost()	Not supported.	Not supported.	Not available.
ivadmin_context_getmgmtsvrport()	Not supported.	Not supported.	Not available.
ivadmin_context_getminpwdalphas()	*PDPolicy object*.**getMinPwdAlphas**	`pdadmin policy get min-password-alphas`	**User** > **Show Global User Policy** > view `Minimum Password Alphas` section
ivadmin_context_getminpwdlen()	*PDPolicy object*.**getMinPwdLen**	`pdadmin policy get min-password-length`	**User** > **Show Global User Policy** > view `Minimum Password Length` section
ivadmin_context_getmin pwdnonalphas()	*PDPolicy object*.**getMinPwdNonAlphas**	`pdadmin policy get min-password-non-alphas`	**User** > **Show Global User Policy** > view `Minimum Password Non-Alphas` section
ivadmin_context_getpwdspaces()	*PDPolicy object*.**pwdSpacesAllowed**	`pdadmin policy get password-spaces`	**User** > **Show Global User Policy** > view `Password Spaces Allowed` section
ivadmin_context_gettodaccess()	*PDPolicy object*.**getAccessibleDays** *PDPolicy object* **.getAccessStartTime** *PDPolicy object*.**getAccessEndTime** *PDPolicy object*.**getAccessTimezone**	`pdadmin policy get tod-access`	**User** > **Show Global User Policy** > view `Time of Day Access` section
ivadmin_context_getuserid()	*PDContext object*.**getUserid**	`pdadmin context show`	Not supported.
ivadmin_context_getuserreg()	**PDUser.getUserRgy**	`pdadmin admin show configuration`	Not supported.
ivadmin_context_hasdelcred()	*PDContext object*.**hasDelegatedCred**	Not applicable.	Not applicable.
ivadmin_context_setaccexpdate()	**PDPolicy.setAcctExpDate** *PDPolicy object*.**setAcctExpDate**	`pdadmin policy set account-expiry-date {unlimited | absolute_time | unset}`	**User** > **Show Global User Policy** > set `Account Expiration Date` > **Apply**
ivadmin_context_setdelcred()	*PDContext object*.**setDelegatedCred**	Not applicable.	Not applicable.
ivadmin_context_setdisabletimeint()	**PDPolicy.setAcctDisableTime** *PDPolicy object*.**setAcctDisableTime**	`pdadmin policy set disable-time-interval {number | unset | disable}`	**User** > **Show Global User Policy** > set `Account Disable Time Interval` > **Apply**

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_context_setmax concurwebsess()	**PDPolicy.setMaxconcurrentWebSessions** *PDPolicy object*.**setMaxconcurrent WebSessions**	`pdadmin policy set max-concurrent-web-sessions {`*number*`|` displace `|` unlimited `|` unset`}` -user *user_name*	**User** > **Show Global User Policy** > set `Max Concurrent Web Sessions` > **Apply**
ivadmin_context_setmaxlgnfails()	**PDPolicy.setMaxFailedLogins** *PDPolicy object*.**setMaxFailedLogins**	`pdadmin policy set max-login-failures {`*number* `|` unset`}`	**User** > **Show Global User Policy** > set `Max Login Failures` > **Apply**
ivadmin_context_setmaxpwdage()	**PDPolicy.setMaxPwdAge** *PDPolicy object*.**setMaxPwdAge**	`pdadmin policy set max-password-age {`*relative_time* `|` unset`}`	**User** > **Show Global User Policy** > set `Max Password Age` > **Apply**
ivadmin_context_setmax pwdrepchars()	**PDPolicy.setMaxPwdRepChars** *PDPolicy object*.**setMaxPwdRepChars**	`pdadmin policy set max-password-repeated-chars [`*number* `|` unset`]`	**User** > **Show Global User Policy** > set `Max Password Repeated Characters` > **Apply**
ivadmin_context_setminpwdalphas()	**PDPolicy.setMinPwdAlphas** *PDPolicy object*.**setMinPwdAlphas**	`pdadmin policy set min-password-alphas {`*number* `|` unset`}`	**User** > **Show Global User Policy** > set `Minimum Password Alphas` > **Apply**
ivadmin_context_setminpwdlen()	**PDPolicy.setMinPwdLen** *PDPolicy object*.**setMinPwdLen**	`pdadmin policy set min-password-length {`*number* `|` unset`}`	**User** > **Show Global User Policy** > set `Minimum Password Length` > **Apply**
ivadmin_context_setmin pwdnonalphas()	**PDPolicy.setMinPwdNonAlphas** *PDPolicy object*.**setMinPwdNonAlphas**	`pdadmin policy set max-password-non-alphas {`*number* `|` unset`}`	**User** > **Show Global User Policy** > set `Minimum Password Non-Alphas` > **Apply**
ivadmin_context_setpwdspaces()	**PDPolicy.setPwdSpacesAllowed** *PDPolicy object*.**setPwdSpacesAllowed**	`pdadmin policy set password-spaces {yes	no	unset}`	**User** > **Show Global User Policy** > set `Password Spaces Allowed` > **Apply**
ivadmin_context_settodaccess()	**PDPolicy.setTodAccess** *PDPolicy object*.**setTodAccess**	`pdadmin policy set tod-access` *todaccess_value*	**User** > **Show Global User Policy** > set `Time of Day Access` > **Apply**
ivadmin_domain_create()	**PDDomain.createDomain**	`pdadmin domain create` *domain domain_admin_id domain_admin_password* `[–desc` *description*`]`	**Secure Domain** > **Create Secure Domain** > fill in form > **Create**
ivadmin_domain_delete()	**PDDomain.deleteDomain**	`pdadmin domain delete` *domain* `[–registry]`	**Secure Domain** > **List Secure Domain** > select secure domain names > **Delete**
ivadmin_domain_get()	**PDDomain** constructor	`pdadmin domain show` *domain*	**Secure Domain** > **List Secure Domain** > click secure domain name
ivadmin_domain_getdescription()	*PDDomain object*.**getDescription**	`pdadmin domain show` *domain*	**Secure Domain** > **List Secure Domain** > click secure domain name
ivadmin_domain_getid()	*PDDomain object*.**getId**	`pdadmin domain show` *domain*	**Secure Domain** > **List Secure Domain** > click secure domain name
ivadmin_domain_list()	**PDDomain.listDomains**	`pdadmin domain list`	**Secure Domain** > **List Secure Domain**
ivadmin_domain_setdescription()	**PDDomain.setDescription** *PDDomain object*.**setDescription**	`pdadmin domain modify` *domain* `description` *description*	**Secure Domain** > **List Secure Domain** > click secure domain name > modify description > **Apply**
ivadmin_free()	Not applicable.	Not applicable.	Not applicable.

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent

ivadmin_group_addmembers()	**PDGroup.addMembers**
	PDGroup object.**addMembers**
		`pdadmin group modify group_name add user`
		
		`pdadmin group modify group_name add (user_1 user_2 [ ... user_n])`
			Group > **Search Groups** > type pattern and maximum results > **Search** > click group name > **Members** tab > select users > **Add**

ivadmin_group_create2()	**PDGroup.createGroup**	`pdadmin group create group_name dn cn [group_container]`	**Group** > **Create Group** > fill in form > **Create**
ivadmin_group_delete2()	**PDGroup.deleteGroup**	`pdadmin group delete [-registry] group_name`	**Group** > **Search Groups** > type pattern and maximum results > **Search** > select group names > **Delete**
ivadmin_group_get()	**PDGroup** constructor	`pdadmin group show group_name`	**Group** > **Search Groups** > type pattern and maximum results > **Search** > click group name
ivadmin_group_getbydn()	**PDGroup** constructor	`pdadmin group show-dn dn`	Not supported.
ivadmin_group_getcn()	Not supported.	`pdadmin group show group_name`	**Group** > **Search Groups** > type pattern and maximum results > **Search** > click group name
ivadmin_group_getdescription()	*PDGroup object*.**getDescription**	`pdadmin group show group_name`	**Group** > **Search Groups** > type pattern and maximum results > **Search** > click group name
ivadmin_group_getdn()	*PDGroup object*.**getRgyName**	`pdadmin group show group_name`	**Group** > **Search Groups** > type pattern and maximum results > **Search** > click group name
ivadmin_group_getid()	*PDGroup object*.**getId**	`pdadmin group show group_name`	**Group** > **Search Groups** > type pattern and maximum results > **Search** > click group name
ivadmin_group_getmembers()	*PDGroup object*.**getMembers**	`pdadmin group show-members group_name`	**Group** > **Search Groups** > type pattern and maximum results > **Search** > click group name > **Members** tab
ivadmin_group_import2()	*PDGroup object*.**importGroup**	`pdadmin group import group_name dn [group_container]`	**Group** > **Import Group** > fill in form > **Import**
ivadmin_group_list()	**PDGroup.listGroups**	`pdadmin group list pattern max_return`	**Group** > **Search Groups** > type pattern and maximum results > **Search**
ivadmin_group_listbydn()	**PDGroup.listGroups**	`pdadmin group list-dn pattern max_return`	Not supported.

ivadmin_group_removemembers()	**PDGroup.removeMembers**
	PDGroup object.**removeMembers**
		`pdadmin group modify group_name remove user`
		
		`pdadmin group modify group_name remove (user_1 user_2 [ ... user_n])`
			Group > **Search Groups** > type pattern and maximum results > **Search** > click group name > **Members** tab > select user names > **Remove**

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_group_setdescription()	**PDGroup.setDescription** *PDGroup object*.**setDescription**	`pdadmin group modify` *`group_name`* `description` *`description`*	**Group** > **Search Groups** > type pattern and maximum results > **Search** > click group name > type description > **Apply**
Not supported	Not supported	`pdadmin help {`*`topic`* | *`command`*`}`	Not supported
Not supported	Not supported	`pdadmin login –a admin_id –p` *`password`* `[–d` *`domain`* | `–m]`	Not supported
Not supported	Not supported	`pdadmin login –l`	Not supported
Not supported	Not supported	`pdadmin logout`	Not supported
ivadmin_objectspace_create()	**PDProtObjectSpace.create ProtObjectSpace**	`pdadmin objectspace create` *`objectspace_name`*	**Object Space** > **Create Object Space** > fill in form > **Create**
ivadmin_objectspace_delete()	**PDProtObjectSpace.delete ProtObjectSpace**	`pdadmin objectspace delete` *`objectspace_name`*	**Object Space** > **Browse Object Space** > click object space name > **Delete**
ivadmin_objectspace_list()	**PDProtObjectSpace.listProtObjectSpaces**	`pdadmin objectspace list`	**Object Space** > **Browse Object Space**
ivadmin_pop_attach()	**PDProtObject.attachPop** *PDProtObject object*.**attachPop**	`pdadmin pop attach` *`object_name pop_name`*	**POP** > **List POPss** > click POP name > **Attach** tab > **Attach** > type protected object path > **Attach**
ivadmin_pop_attrdelkey()	**PDPop.deleteAttribute** *PDPop object* **.deleteAttribute**	`pdadmin pop modify` *`pop_name`* `delete attribute` *`attribute_name`*	**POP** > **List POPs** > click POP name > **Extended Attributes** tab > select attributes > **Delete**
ivadmin_pop_attrdelval()	**PDPop.deleteAttributeValue** *PDPop object*. **deleteAttributeValue**	`pdadmin pop modify` *`pop_name`* `delete attribute` *`attribute_name attribute_value`*	Not supported
ivadmin_pop_attrget()	*PDPop object*.**getAttributeValues**	`pdadmin pop show` *`pop_name`* `attribute`	**POP** > **List POPs** > click POP name > **Extended Attributes** tab
ivadmin_pop_attrlist()	*PDPop object*.**getAttributeNames**	`pdadmin pop list` *`pop_name`* `attribute`	**POP** > **List POPs** > click POP name > **Extended Attributes** tab
ivadmin_pop_attrput()	**PDPop.setAttributeValue***PDPop object* **.setAttributeValue**	`pdadmin pop modify` *`pop_name`* `set attribute` *`attribute_name attribute_value`*	**POP** > **List POPs** > click POP name > **Extended Attributes** tab > **Create** > fill in form > **Apply**
ivadmin_pop_create()	**PDPop.createPop**	`pdadmin pop create` *`pop_name`*	**POP** > **Create POP** > fill in form > **Create**
ivadmin_pop_delete()	**PDPop.deletePop**	`pdadmin pop delete` *`pop_name`*	**POP** > **List POPs** > select POP names > **Delete**
ivadmin_pop_detach()	**PDProtObject.detachPop** *PDProtObject object*.**attachPop**	`pdadmin pop detach` *`object_name`*	**POP** > **List POPs** > click POP name > **Attach** tab > select object > **Detach**
ivadmin_pop_find()	**PDProtObject.listProtObjectsByPop**	`pdadmin pop find` *`pop_name`*	**POP** > **List POPs** > click POP name > **Attach** tab
ivadmin_pop_get()	**PDPop** constructor	`pdadmin pop show` *`pop_name`*	**POP** > **List POPs** > click POP name
ivadmin_pop_getanyothernw2()	*PDPop object*.**getIPAuthInfo**	`pdadmin pop show` *`pop_name`*	**POP** > **List POPs** > click POP name
ivadmin_pop_getauditlevel()	*PDPop object*.**getAuditLevel**	`pdadmin pop show` *`pop_name`*	**POP** > **List POPs** > click POP name
ivadmin_pop_getdescription()	*PDPop object*.**getDescription**	`pdadmin pop show` *`pop_name`*	**POP** > **List POPs** > click POP name

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_pop_getid()	*PDPop object*.**getId**	`pdadmin pop show pop_name`	**POP** > **List POPs** > click POP name
ivadmin_pop_getipauth2()	*PDPop object*.**getIPAuthInfo**	`pdadmin pop show pop_name`	**POP** > **List POPs** > click POP name
ivadmin_pop_getqop()	*PDPop object*.**getQOP**	`pdadmin pop show pop_name`	**POP** > **List POPs** > click POP name
ivadmin_pop_gettod()	*PDPop object*.**getTodAccessInfo**	`pdadmin pop show pop_name`	**POP** > **List POPs** > click POP name
ivadmin_pop_getwarnmode()	*PDPop object*.**getWarningMode**	`pdadmin pop show pop_name`	**POP** > **List POPs** > click POP name
ivadmin_pop_list()	**PDPop.listPops**	`pdadmin pop list`	**POP** > **List POPs**
ivadmin_pop_list()	**PDPop.listPops**	`pdadmin pop list pop_name`	**POP** > **List POPs** > click POP name
ivadmin_pop_removeipauth2()	**PDPop.removeIPAuthInfo** *PDPop object*.**removeIPAuthInfo**	`pdadmin pop modify pop_name set ipauth remove network netmask`	**POP** > **List POPs** > click POP name > **IP Auth** tab > select IP authorization entries > **Delete**
ivadmin_pop_setanyothernw2()	**PDPop.setIPAuthInfo**	`pdadmin pop modify pop_name set ipauth anyothernw authentication_level`	**POP** > **List POPs** > click POP name > **IP Auth** tab > **Create** > select **Any Other Network** check box. and type the authentication level > **Create**
ivadmin_pop_setanyothernw _forbidden2()	**PDPop.setIPAuthInfo**	`pdadmin pop modify pop_name set ipauth anyothernw forbidden`	**POP** > **List POPs** > click POP name > **IP Auth** tab > **Create** > select **Any Other Network** and **Forbidden** check boxes > **Create**
ivadmin_pop_setauditlevel()	**PDPop.setAuditLevel***PDPop object* **.setAuditLevel**	`pdadmin pop modify pop_name set audit-level {all | none | audit_level_list}`	**POP** > **List POPs** > click POP name > select or clear appropriate check boxes > **Apply**
ivadmin_pop_setdescription()	**PDPop.setDescription***PDPop object* **.setDescription**	`pdadmin pop modify pop_name set description description`	**POP** > **List POPs** > click POP name > modify description > **Apply**
ivadmin_pop_setipauth2()	**PDPop.setIPAuthInfo***PDPop object* **.setIPAuthInfo**	`pdadmin pop modify pop_name set ipauth add network netmask authentication_level`	**POP** > **List POPs** > click POP name > **IP Auth** tab > **Create** > type the network, net mask, and authentication level > **Create**
ivadmin_pop_setipauth_forbidden2()	**PDPop.setIPAuthInfo***PDPop object* **.setIPAuthInfo**	`pdadmin pop modify pop_nameset ipauth add network netmask forbidden`	**POP** > **List POPs** > click POP name > **IP Auth** tab > **Create** > type network and net mask and select **Forbidden** check box > **Apply**
ivadmin_pop_setqop()	**PDPop.setQOP***PDPop object***.setQOP**	`pdadmin pop modify pop_name set qop {none | integrity | privacy}`	**POP** > **List POPs** > click POP name > select appropriate quality of protection > **Apply**
ivadmin_pop_settod()	**PDPop.setTodAccessInfo***PDPop object* **.setTodAccessInfo** .	`pdadmin pop modify pop_name set tod-access {anyday | weekday | day_list}:{anytime | time_spec-time_spec} [:utc | local]`	**POP** > **List POPs** > click POP name > define time of day access > **Apply**
ivadmin_pop_setwarnmode()	**PDPop.setWarningMode***PDPop object* **.setWarningMode**	`pdadmin pop modify pop_name set warning {yes | no]`	**POP** > **List POPs** > click POP name > select or clear **Warn Only On Policy Violation** check box > **Apply**

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
Not supported	Not supported	`pdadmin quit`	Not supported
ivadmin_protobj_access()	**PDProtObject.access**	`pdadmin object access object_name permissions`	Not supported.
ivadmin_protobj_attachacl()	**PDProtObject.attachAcl** *PDProtObject object*.**attachAcl**	`pdadmin acl attach object_name acl_name`	**ACL** > **List ACL** > click ACL name > **Attach** tab > **Attach** > type protected object path > **Attach**
ivadmin_protobj_attachauthzrule()	**PDProtObject.attachAuthzRule** *PDProtObject object*.**attachAuthzRule**	`pdadmin authzrule attach object_name ruleid`	**AuthzRule** > **List AuthzRule** > click authorization rule name > **Attach** tab > **Attach** > type protected object path > **Attach**
ivadmin_protobj_attrdelkey()	**PDProtObject.deleteAttribute** *PDProtObject object*.**deleteAttribute**	`pdadmin object modify object_name delete attribute_name`	**Object Space** > **Browse Object Space** > expand and click object name > **Extended Attributes** tab > select attribute > **Delete**
ivadmin_protobj_attrdelval()	**PDProtObject.deleteAttributeValue** *PDProtObject object*.**deleteAttributeValue**	`pdadmin object modify object_name delete attribute_name attribute_value`	Not supported
ivadmin_protobj_attrget()	*PDProtObject object*.**getAttributeValues**	`pdadmin object show object_name attribute attribute_name`	**Object Space** > **Browse Object Space** > expand and click object name > **Extended Attributes** tab
	PDProtObject.listProtObjects	`pdadmin object list`	**Object Space** > **Browse Object Space**
ivadmin_protobj_attrlist()	*PDProtObject object*.**getAttributeNames**	`pdadmin object list object_name attribute`	**Object Space** > **Browse Object Space** > expand and click object name > **Extended Attributes** tab
		`pdadmin object listandshow object_name`	Not supported
ivadmin_protobj_attrput()	**PDProtObject.setAttributeValue** *PDProtObject object*.**setAttributeValue**	`pdadmin object modify object_name set attribute attribute_name attribute_value`	**Object Space** > **Browse Object Space** > expand and click object name > **Extended Attributes** tab > **Create** > fill in form > **Apply**
ivadmin_protobj_create()	**PDProtObject.createProtObject**	`pdadmin object create object_name description type ispolicyattachable {yes|no}`	**Object Space** > **Create Object** > fill in form > **Create** The **type** field is not supported. You can select the **Can Policy be attached to this object** check box on the Protected Object Properties page.
ivadmin_protobj_delete()	**PDProtObject.deleteProtObject**	`pdadmin object delete object_name`	**Object Space** > **Browse Object Space** > expand and click object name > **Delete**
ivadmin_protobj_detachacl()	**PDProtObject.detachAcl** *PDProtObject object*.**detachAcl**	`pdadmin acl detach object_name`	**ACL** > **List ACL** > click ACL name > **Attach** tab > select protected object > **Detach**

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_protobj_detachauthzrule()	**PDProtObject.detachAuthzRule** *PDProtObject object*.**detachAuthzRule**	`pdadmin authzrule detach` *`object_name`*	**AuthzRule** > **List AuthzRule** > click authorization rule name > **Attach** tab > select object names > **Detach**
ivadmin_protobj_exists()	**PDProtObject.exists**	`pdadmin object exists` *`object_name`*	Not supported.
ivadmin_protobj_get3()	**PDProtObject** constructor **Note:** If the protected object name specified does not exist, default values are shown. To determine that a protected object exists, use the **`PDProtObject.exists`** command.	`pdadmin object show` *`object_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_getaclid()	*PDProtObject object*.**getAcl**	`pdadmin object show` *`object_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_getauthzruleid()	*PDProtObject object*.**getAuthzRule**	`pdadmin object show` *`object_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_getdesc()	*PDProtObject object*.**getDescription**	`pdadmin object show` *`object_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_geteffaclid()	*PDProtObject object*.**getEffectuveAclId**	`pdadmin object show` *`object_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_geteffauthzruleid()	*PDProtObject object*.**getEffectuveAuthzRuleId**	`pdadmin object show` *`object_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_geteffpopid()	*PDProtObject object*.**getEffectuvePopId**	`pdadmin object show` *`object_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_getid()	*PDProtObject object*.**getId**	`pdadmin object show` *`object_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_ getpolicyattachable()	*PDProtObject object*.**isPolicyAttachable**	`pdadmin object show` *`object_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_getpopid()	*PDProtObject object*.**getPopId**	`pdadmin object show` *`object_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_gettype()	Not supported	`pdadmin object show` *`object_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_list3()	**PDProtObject.listProtObjects** **Note:** Before using this information and the product it supports, read the information in "Notices" on page 319.	`pdadmin object list` *`directory_name`*	**Object Space** > **Browse Object Space** > expand and click object name
ivadmin_protobj_listbyacl()	**PDProtObject.listProtObjectsByAcl**	`pdadmin acl find` *`acl_name`*	**ACL** > **List ACL** > click ACL name > **Attach** tab
ivadmin_protobj_listbyauthzrule()	**PDProtObject.listProtObjects ByAuthzRule**	`pdadmin authzrule find` *`ruleid`*	**AuthzRule** > **List AuthzRule** > click authorization rule name > **Attach** tab
ivadmin_protobj_multiaccess()	**PDProtObject.multiAccess**	`pdadmin object access` *`object_name permissions`*	Not supported.
ivadmin_protobj_setdesc()	**PDProtObject.setDescription** *PDProtObject object*.**setDescription**	`pdadmin object modify` *`object_name`* `set description` *`description`*	**Object Space** > **Browse Object Space** > expand and click object name > modify description > **Apply**
ivadmin_protobj_setname()	Not supported	`pdadmin object modify` *`object_name`* `name` *`name`* `conflict_resolution resolution_modifier`	Not supported

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_protobj_ setpolicyattachable()	**PDProtObject.setPolicyAttachable** *PDProtObject object*.**setPolicyAttachable**	`pdadmin object modify` *`object_name`* `isPolicyAttachable {yes|no}`	**Object Space** > **Browse Object Space** > expand and click object name > select or clear check box> **Apply**
ivadmin_protobj_settype()	Not supported.	`pdadmin object modify` *`object_name`* `type` *`type`*	Not supported.
ivadmin_response_getcode()	Not applicable.	Not applicable.	Not applicable.
ivadmin_response_getcount()	Not applicable.	Not applicable.	Not applicable.
ivadmin_response_getmessage()	Not applicable.	Not applicable.	Not applicable.
ivadmin_response_getmodifier()	Not applicable.	Not applicable.	Not applicable.
ivadmin_response_getok()	Not applicable.	Not applicable.	Not applicable.
Not supported.	Not supported.	`pdadmin server list`	Not supported.
ivadmin_server_gettasklist()	**PDServer.getTaskList**	`pdadmin server listtasks` *`server_name`*	Not supported.
Not supported	Not supported	`pdadmin server show` *`server_name`*	Not supported
ivadmin_server_performtask()	**PDServer.performTask**	`pdadmin server task` *`server_name server_task`*	Not supported. For more information about the WebSEAL server tasks and junction points, see the *IBM Security Access Manager for Web: WebSEAL Administration Guide*.
		`pdadmin server task` *`server_name`* `{help | stats | trace}`	Not supported.
ivadmin_server_replicate()	**PDServer.serverReplicate**	`pdadmin server replicate` *`server_name`*	Not supported.
ivadmin_ssocred_create()	**PDSSOCred.createSSOCred**	`pdadmin rsrccred create` *`resource_name`* `rsrcuser` *`resource_userid`* `rsrcpwd` *`resource_pwd`* `rsrctype {web | group} user` *`user_name`*	**User** > **Search Users** > **Search** > click user name > **GSO Credentials** tab > **Create** > fill in form > **Create**
ivadmin_ssocred_create()	**PDSSOCred.createSSOCred**	`pdadmin rsrccred create` *`resource_group_name`* `rsrcuser` *`resource_userid`* `rsrcpwd` *`resource_pwd`* `rsrctype {web | group} user` *`user_name`*	**User** > **Search Groups** > **Search** > click user name > **GSO Credentials** tab > **Create** > fill in form > **Create**
ivadmin_ssocred_delete()	**PDSSOCred.deleteSSOCred**	`pdadmin rsrccred delete` *`resource_name`* `rsrctype {web | group} user` *`user_name`*	**User** > **Search Users** > **Search** > click user name > **GSO Credentials** tab > select credentials > **Delete**
ivadmin_ssocred_delete()	**PDSSOCred.deleteSSOCred**	`pdadmin rsrccred delete` *`resource_group_name`* `rsrctype {web | group} user` *`user_name`*	**User** > **Search Groups** > **Search** > click user name > **GSO Credentials** tab > select credentials > **Delete**
ivadmin_ssocred_get()	**PDSSOCred** constructor	`pdadmin rsrccred show` *`resource_name`* `rsrctype {web | group} user` *`user_name`*	**User** > **Search Users** > **Search** > click user name > **GSO Credentials** tab
ivadmin_ssocred_getid()	*PDSSOCred object*.**getResourceName**	`pdadmin rsrccred show` *`resource_name`* `rsrctype {web | group} user` *`user_name`*	**User** > **Search Users** > **Search** > click user name > **GSO Credentials** tab
ivadmin_ssocred_getssopassword()	*PDSSOCred object*.**getResourcePassword**	Not applicable.	Not applicable.

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_ssocred_getssouser()	*PDSSOCred object*.**getResourceUser**	Not applicable.	Not applicable.
ivadmin_ssocred_gettype()	*PDSSOCred object*.**getResourceType**	`pdadmin rsrccred show` *resource_name* `rsrctype {web | group} user` *user_name*	**User** > **Search Users** > **Search** > click user name > **GSO Credentials** tab
ivadmin_ssocred_getuser()	*PDSSOCred object*.**getUser**	`pdadmin rsrccred show` *resource_name* `rsrctype {web | group} user` *user_name*	**User** > **Search Users** > **Search** > click user name > **GSO Credentials** tab
ivadmin_ssocred_get()	**PDSSOCred** constructor	`pdadmin rsrccred show` *resource_group_name* `rsrctype {web | group} user` *user_name*	**User** > **Search Groups** > **Search** > click user name > **GSO Credentials** tab
ivadmin_ssocred_list()	*PDSSOCred object*.**listAndShowSSOCreds** *PDSSOCred object* **.listSSOCreds**	`pdadmin rsrccred list user` *user_name*	**User** > **Search Users** > **Search** > click user name > **GSO Credentials** tab
ivadmin_ssocred_set()	**PDSSOCred.setSSOCred** *PDSSOCred object*	`pdadmin rsrccred modify` *resource_name* `rsrctype {web | group} [-rsrcuser` *resource_userid*`] [-rsrcpwd` *resource_pwd*`] user` *user_name*	**User** > **Search Users** > **Search** > click user name > **GSO Credentials** tab > **Create** > modify form > **Create**
ivadmin_ssocred_set()	**PDSSOCred.setSSOCred .setSSOCred***PDSSOCred object*	`pdadmin rsrccred modify` *resource_group_name* `rsrctype {web | group} [-rsrcuser` *resource_userid*`] [-rsrcpwd` *resource_pwd*`] user` *user_name*	**User** > **Search Groups** > **Search** > click user name > **GSO Credentials** tab > **Create** > modify form > **Create**
ivadmin_ssogroup_addres()	**PDSSOResourceGroup .addSSOResource***PDSSOResourceGroup object*	`pdadmin rsrcgroup modify` *resource_group_name* `add rsrcname` *resource_name*	**GSO Resource** > **List GSO Groups** > select resource group > select members > **Add**
ivadmin_ssogroup_create()	**PDSSOResourceGroup .createSSOResourceGroup**	`pdadmin rsrcgroup create` *resource_group_name*	**GSO Resource** > **Create GSO Group** > fill in form > **Create**
ivadmin_ssogroup_create()	**PDSSOResourceGroup .createSSOResourceGroup**	`pdadmin rsrcgroup create` *resource_group_name* `-desc` *description*	**GSO Resource** > **Create GSO Group** > fill in form and modify the description > **Create**
ivadmin_ssogroup_delete()	**PDSSOResourceGroup .deleteSSOResourceGroup**	`pdadmin rsrcgroup delete` *resource_group_name*	**GSO Resource** > **List GSO Groups** > select resource groups > **Delete**
ivadmin_ssogroup_get()	**PDSSOResourceGroup** constructor	`pdadmin rsrcgroup show` *resource_group_name*	**GSO Resource** > **List GSO Groups** > select resource group
ivadmin_ssogroup_getdescription()	*PDSSOResourceGroup object*.**getDescription**	`pdadmin rsrcgroup show` *resource_group_name*	**GSO Resource** > **List GSO Groups** > select resource group
ivadmin_ssogroup_getid()	*PDSSOResourceGroup object*.**getId**	`pdadmin rsrcgroup show` *resource_group_name*	**GSO Resource** > **List GSO Groups** > select resource group
ivadmin_ssogroup_getresources()	*PDSSOResourceGroup object*.**getSSOResources**	`pdadmin rsrcgroup show` *resource_group_name*	**GSO Resource** > **List GSO Groups** > select resource group
ivadmin_ssogroup_list()	**PDSSOResourceGroup .listSSOResourceGroups**	`pdadmin rsrcgroup list`	**GSO Resource** > **List GSO Groups**
ivadmin_ssogroup_removeres()	**PDSSOResourceGroup .removeSSOResource** *PDSSOResourceGroup object* **.removeSSOResource**.	`pdadmin rsrcgroup modify` *resource_group_name* `remove rsrcname` *resource_name*	**GSO Resource** > **List GSO Groups** > select resource group > select members > **Remove**
ivadmin_ssoweb_create()	**PDSSOResource.createSSOResource**	`pdadmin rsrc create` *resource_name*	**GSO Resource** > **Create GSO** > fill in form > **Create**

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_ssoweb_create()	**PDSSOResource.createSSOResource**	`pdadmin rsrc create` *resource_name* `-desc` *description*	**GSO Resource** > **Create GSO** > fill in form and modify the description > **Create**
ivadmin_ssoweb_delete()	**PDSSOResource.deleteSSOResource**	`pdadmin rsrc delete` *resource_name*	**GSO Resource** > **List GSO** > select resources > **Delete**
ivadmin_ssoweb_get()	**PDSSOResource** constructor	`pdadmin rsrc show` *resource_name*	**GSO Resource** > **List GSO** > click resource
ivadmin_ssoweb_getdescription()	*PDSSOResource object*.**getDescription**	`pdadmin rsrc show` *resource_name*	**GSO Resource** > **List GSO** > click resource
ivadmin_ssoweb_getid()	*PDSSOResource object*.**getId**	`pdadmin rsrc show` *resource_name*	**GSO Resource** > **List GSO** > click resource
ivadmin_ssoweb_list()	**PDSSOResource.listSSOResources**	`pdadmin rsrc list`	**GSO Resource** > **List GSO**
ivadmin_user_create3()	**PDUser.createUser**	`pdadmin user create [-gsouser] [-no-password-policy]` *user_name dn cn sn password* [*group1* [*group2* ...]]	**User** > **Create User** > fill in form > **Create**
ivadmin_user_delete2()	**PDUser.deleteUser**	`pdadmin user delete [-registry]` *user_name*	**User** > **Search Users** > type pattern and maximum results > **Search** > select user names > **Delete**
ivadmin_user_get()	**PDUser** constructor	`pdadmin user show` *user_name*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name
ivadmin_user_getaccexpdate()	*PDPolicy object*.**getAcctExpDate**	`pdadmin user get account-expiry-date -user` *user_name*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab
ivadmin_user_getaccountvalid()	*PDUser object*.**isAccountValid**	`pdadmin user show` *user_name*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name
ivadmin_user_getbydn()	**PDUser** constructor	`pdadmin user show-dn` *dn*	Not supported
ivadmin_user_getcn()	*PDUser object*.**getFirstName**	`pdadmin user show` *user_name*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name
ivadmin_user_getdescription()	*PDUser object*.**getDescription**	`pdadmin user show` *user_name*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name
ivadmin_user_getdisabletimeint()	*PDPolicy object*.**getAcctDisableTimeInterval**	`pdadmin policy get disable-time-interval -user` *user_name*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab
ivadmin_user_getdn()	*PDUser object*.**getRgyName**	`pdadmin user show` *user_name*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name
ivadmin_user_getid()	*PDUser object*.**getId**	`pdadmin user show` *user_name*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name
ivadmin_user_getmaxconcurwebsess()	*PDPolicy object*.**getMaxconcurrentWebSessions**	`pdadmin policy get max-concurrent-web-sessions -user` *user_name*	**User** > **Show Global User Policy** > view `Max Concurrent Web Sessions`

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_user_getmaxlgnfails()	*PDPolicy object*.**getMaxFailedLogins**	`pdadmin policy get max-login-failures -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab
ivadmin_user_getmaxpwdage()	*PDPolicy object*.**getMaxPwdAge**	`pdadmin policy get max-password-age -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab
ivadmin_user_getmaxpwdrepchars()	*PDPolicy object*.**getMaxPwdRepChars**	`pdadmin policy get max-password-repeated-chars -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab
ivadmin_user_getmemberships()	*PDUser object*.**getGroups**	`pdadmin user show-groups user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Groups** tab
ivadmin_user_getminpwdalphas()	*PDPolicy object*.**getMinPwdAlphas**	`pdadmin policy get min-password-alphas -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab
ivadmin_user_getminpwdlen()	*PDPolicy object*.**getMinPwdLen**	`pdadmin policy get min-password-length -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab
ivadmin_user_ getminpwdnonalphas()	*PDPolicy object*.**getMinPwdNonAlphas**	`pdadmin policy get min-password-non-alphas -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab
ivadmin_user_getpasswordvalid()	*PDUser object*.**isPasswordValid**	`pdadmin user show user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name
ivadmin_user_getpwdspaces()	*PDPolicy object*.**pwdSpacesAllowed**	`pdadmin policy get password-spaces -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab
ivadmin_user_getsn()	*PDUser object*.**getLastName**	`pdadmin user show user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name
ivadmin_user_getssouser()	*PDUser object*.**isSSOUser**	`pdadmin user show user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name
ivadmin_user_gettodaccess()	*PDPolicy object*.**getAccessibleDays** *PDPolicy object*.**getAccessStartTime** *PDPolicy object*.**getAccessEndTime**	`pdadmin policy get tod-access -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab
ivadmin_user_import2()	**PDUser.importUser**	`pdadmin user import [-gsouser] user_name dn [group_name]`	**User** > **Import User** > fill in form > **Create**
ivadmin_user_list()	**PDUser.listUsers**	`pdadmin user list pattern max_return`	**User** > **Search Users** > type pattern and maximum results > **Search**

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_user_listbydn()	**PDUser.listUsers**	`pdadmin user list-dn pattern max_return`	Not supported.
ivadmin_user_setaccexpdate()	**PDPolicy.setAcctExpDate** *PDPolicy object*.**setAcctExpDate**	`pdadmin policy set account-expiry-date {unlimited | absolute_time | unset} -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab > modify value > **Apply**
ivadmin_user_setaccountvalid()	**PDUser.setAccountValid** *PDUser object*.**setAccountValid**	`pdadmin user modify user_name account-valid {yes|no}`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > select or clear check box > **Apply**
ivadmin_user_setdescription()	**PDUser.setDescription** *PDUser object*.**setDescription**	`pdadmin user modify user_name description description`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > modify description> **Apply**
ivadmin_user_setdisabletimeint()	**PDPolicy.setAcctDisableTime** *PDPolicy object*.**setAcctDisableTime**	`pdadmin policy set disable-time-interval {number | unset | disable} -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab > modify value > **Apply**
ivadmin_user_ setmaxconcurwebsess()con_l	**PDPolicy.Max ConcurrentWebSessionsDisplaced** **PDPolicy.MaxConcurrent WebSessionsUnlimited** **PDPolicy.MaxConcurrent WebSessionsEnforced** **PDPolicy.setMaxconcurrent WebSessions** *PDPolicy object*.**setMaxconcurrent WebSessions**	`pdadmin policy set max-concurrent-web-sessions {number | displace | unlimited | unset} -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab > modify value > **Apply**
ivadmin_user_setmaxlgnfails()	**PDPolicy.setMaxFailedLogins** *PDPolicy object*.**setMaxFailedLogins**	`pdadmin policy set max-login-failures {number | unset} -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab > modify value > **Apply**
ivadmin_user_setmaxpwdage()	**PDPolicy.setMaxPwdAge** *PDPolicy object*.**setMaxPwdAge**	`pdadmin policy set max-password-age {unset | relative_time} -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab > modify value > **Apply**
ivadmin_user_setmaxpwdrepchars()	**PDPolicy.setMaxPwdRepChars** *PDPolicy object*.**setMaxPwdRepChars**	`pdadmin policy set max-password-repeated-chars {number | unset} -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab > modify value > **Apply**
ivadmin_user_setminpwdalphas()	**PDPolicy.setMinPwdAlphas** *PDPolicy object*.**setMinPwdAlphas**	`pdadmin policy set min-password-alphas {number | unset} -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab > modify value > **Apply**
ivadmin_user_setminpwdlen()	**PDPolicy.setMinPwdLen** *PDPolicy object*.**setMinPwdLen**	`pdadmin policy set min-password-length {number | unset} -user user_name`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab > modify value > **Apply**

Table 39. Mapping of the administration C API to the Java methods, the pdadmin interface, and Web Portal Manager (continued)

C API	Java class and method	pdadmin command equivalent	Web Portal Manager equivalent
ivadmin_user_setminpwdnonalphas()	**PDPolicy.setMinPwdNonAlphas** *PDPolicy object*.**setMinPwdNonAlphas**	`pdadmin policy set min-password-non-alphas {`*`number`*` | unset} -user `*`user_name`*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab > modify value > **Apply** tab
ivadmin_user_setpassword()	**PDUser.setPassword** *PDUser object*.**setPassword**	`pdadmin user modify `*`user_name`*` password `*`password`*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > modify password> **Apply**
ivadmin_user_setpasswordvalid()	**PDUser.setPasswordValid** *PDUser object*.**setPasswordValid**	`pdadmin user modify `*`user_name`*` password-valid {yes|no}`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > select or clear check box > **Apply**
ivadmin_user_setpwdspaces()	**PDPolicy.setPwdSpacesAllowed** *PDPolicy object*.**setPwdSpacesAllowed**	`pdadmin policy set password-spaces {yes | no | unset} -user `*`user_name`*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab > modify value > **Apply**
ivadmin_user_setssouser()	**PDUser.setSSOUser** *PDUser object*.**setSSOUser**	`pdadmin user modify `*`user_name`*` gsouser {yes|no}`	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > select or clear check box > **Apply**
ivadmin_user_settodaccess()	**PDPolicy.setTodAccess** *PDPolicy object*.**setTodAccess**	`pdadmin policy set tod-access `*`tod_value`* `-user `*`user_name`*	**User** > **Search Users** > type pattern and maximum results > **Search** > click user name > **Policy** tab > modify value > **Apply**

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features contained in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM might have patents or pending patent applications that cover subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it to enable: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information might be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments might vary significantly. Some measurements might have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements might have been estimated through extrapolation. Actual results might vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding the future direction or intent of IBM are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing, or distributing application programs that conform to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample

programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2004, 2012. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations might not appear.

Privacy Policy Considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both: http://www.ibm.com/legal/copytrade.shtml

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

The Oracle Outside In Technology included herein is subject to a restricted use license and can only be used in conjunction with this application.

Index

G

I

J

L

M

N

O

P

R

S

T

U

W

Printed in USA