IBM Security Access Manager
Version 9.0.5
June 2018

Access Manager Platform and Supporting Components Administration Topics

IBM

IBM Security Access Manager
Version 9.0.5
June 2018

Access Manager Platform and Supporting Components Administration Topics

Contents

Figures

Tables

Chapter 1. Security Access Manager overview

Security Access Manager is an authentication and authorization solution for corporate web, client/server, and existing applications. Use Security Access Manager to control user access to protected information and resources. By providing a centralized, flexible, and scalable access control solution, Security Access Manager builds secure and easy-to-manage network-based applications and infrastructure.

Security Access Manager supports authentication, authorization, data security, and resource management capabilities. You use Security Access Manager in conjunction with standard Internet-based applications to build highly secure and well-managed intranets.

Security Access Manager provides the following frameworks:

Authentication framework
The Security Access Manager authentication service uses a wide range of built-in authenticators and supports external authenticators.

Authorization framework
The authorization service, accessed through a standard authorization application programming interface (API), provides permit and deny decisions on access requests for native Security Access Manager servers and other applications.

The authorization service, together with resource managers, provides a standard authorization mechanism for business network systems.

Security Access Manager can be integrated into existing and emerging infrastructures to provide secure, centralized policy management capability.

The following resource managers are some of the existing resource managers:

IBM Security Access Manager for Web WebSEAL
Manages and protects web-based information and resources. WebSEAL is included with Security Access Manager.

IBM Security Access Manager for Web for Operating Systems
Provides a layer of authorization policy enforcement on Linux and UNIX operating systems in addition to that provided by the native operating system.

Existing applications can take advantage of the Security Access Manager authorization service and provide a common security policy for the entire enterprise.

Core technologies

The Security Access Manager network security management solution provides and supports several core technologies and features.

See the following topics for information:

Authentication

Authentication is the first step a user must take when making a request for a resource that is protected by Security Access Manager. During authentication, a user identity is validated.

The authentication process depends on the specific requirements of the service-providing application. Security Access Manager allows a highly flexible approach to authentication through the use of the authorization API.

Security Access Manager provides built-in support of user name and password authentication through the authorization API. Applications can build any custom authentication mechanism that uses the authorization API.

Authorization

Authorization enforces the security policy.

- The authorization process determines which objects a user can access and which actions a user can take on those objects.
- The authorization process grants appropriate access to the user.

Security Access Manager handles authorization by using the following methods:

- Security Access Manager authorization service.
- Access control lists (ACLs), protected object policies (POPs), and authorization rules for fine-grained access control.
- Standards-based authorization API, which uses the aznAPI for C language applications, and the Java™ Authentication and Authorization Service (JAAS) for Java language applications.
- External authorization service capability.

Quality of Protection

Quality of Protection (QoP) is the degree to which Security Access Manager protects any information that is transmitted between a client and a server.

The quality of data protection is determined by the combined effect of encryption standards and modification-detection algorithms. The resource manager is responsible for ensuring that the quality of data protection is enforced.

Security Access Manager supports the following levels of Quality of Protection:

- Standard Transmission Control Protocol (TCP) communication (no protection)
- Data integrity protects messages (data stream) from being modified during network communication
- Data privacy protects messages from being modified or inspected during network communication

Supported encryption ciphers

Security Access Manager uses encryption ciphers from GSKit and Java Secure Socket Extension (JSSE).

To learn about these encryption ciphers, see the GSKit and JSSE documentation.

Secure communication

Security Access Manager supports the data integrity and data privacy provided by the Secure Socket Layer (SSL) communication protocol and the Transport Layer Security (TLS) communication protocol.

The SSL handshake protocol provides security and privacy over the Internet. SSL works with public key for authentication and secret key to encrypt data that is transferred over the SSL connection.

The TLS protocol meets the Federal Information Processing Standards (FIPS) 140-2 standard. The FIPS standard describes the requirements of the United States federal government for handling sensitive, but unclassified, use of information technology products. When FIPS mode is enabled in Security Access Manager, TLS version 1 (TLSv1) is used instead of SSL version 3 (SSLv3).

Security Access Manager generates keys and certificates with FIPS-approved operations. The client- and server-side keys and certificates are always FIPS approved.

To switch from SSL to TLS, you must change all server and remote run time configurations. In Security Access Manager, the protocol configuration specifies the FIPS mode. When FIPS mode is enabled, it uses the TLS protocol. When FIPS mode is disabled, it uses the SSL protocol.

Note: SSL and TLS protocols cannot be mixed in a Security Access Manager environment. Previous releases of IBM Security Access Manager runtime that did not support TLS cannot communicate with a server that is enabled for FIPS.

Security standards configurations (compliance types)

You can configure Security Access Manager Base components to work with various security standards, including FIPS 140-2, SP 800-131, and Suite B. These security standards meet information security requirements that are required by the government.

Security Access Manager uses cryptography in the following areas:

- To create and replace internal, self-signed certificates. These certificates are used by Security Access Manager Runtime and Security Access Manager server to authenticate with each other.
- Secure communication between the runtime and servers.
- Secure communication to LDAP.
- Secure communication to Syslog servers.

The Security Access Manager Base components integrate cryptographic modules, which include IBM Global Security Kit (GSKit) 8, Java Secure Socket Extension (JSSE), and Java Cryptography Extension (JCE). Most of the requirements in the standards are handled in GSKit, JSSE, and JCE, which must undergo the certification process to meet government standards. Security Access Manager Base components must be configured to run with GSKit, JSSE, and JCE that are enabled for a particular standard.

FIPS 140-2

The Federal Information Processing Standards (FIPS) specify federal government requirements for cryptographic modules. FIPS 140-2 is a National Institute of Standards and Technology standard.

The Security Access Manager Base components use certificates generated by the policy server to communicate securely in accordance with FIPS 140-2. The key strength and algorithms that generate FIPS 140-2 certificates are also used when Security Access Manager is not configured for a particular security mode. You can convert between these two modes without completely regenerating all the Security Access Manager certificates.

The FIPS 140-2 certificates are compatible with previous releases of Security Access Manager. Previous releases of Security Access Manager can communicate with Security Access Manager 8.0 policy servers.

Federal Information Processing Standards (FIPS) that specify requirements on cryptographic modules. For more information, see the National Institute of Standards and Technology website http://csrc.nist.gov/publications/PubsFIPS.html.

For more information about FIPS 140-2, see http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

SP 800-131a

Special Publication 800-131a (SP 800-131a) is an information security standard of the National Institute of Standards and Technology (NIST). SP 800-131a requires longer key lengths and stronger cryptography than other standards.

You can run SP 800-131a in two modes: transition and strict. Use the transition mode to move gradually towards a strict enforcement of SP 800-131a. The transition mode allows the use of weaker keys and algorithms than strict enforcement allows. The transition mode also allows the use of Transport Layer Security (TLS) v1.0 and v1.1.

A strict enforcement of SP 800-131a of the Security Access Manager Base components requires the following configuration:

- TLS v1.2 protocol for the Secure Sockets Layer (SSL) context
- Certificates must have a minimum length of 2048
- Elliptical Curve (EC) certificates must have a minimum size of 244-bit curves
- Certificates must be signed with a signature algorithm of SHA256, SHA384, or SHA512. Valid signature algorithms include:
  - `SHA256withRSA`
  - `SHA384withRSA`
  - `SHA512withRSA`
  - `SHA256withECDSA`
  - `SHA384withECDSA`
  - `SHA512withECDSA`
- `SP 800-131a` approved cipher suites

The Security Access Manager Base component communication uses certificates that are generated by the policy server. The policy server uses the same key strength and algorithms to create certificates for both the transition and strict versions of the SP 800-131a security mode. As a result, you can convert between the transition and strict modes without completely regenerating all Security Access Manager certificates.

The SP 800-131a certificates are not compatible with previous releases of Security Access Manager. Previous release Security Access Manager clients cannot communicate with the policy server of previous releases of the Security Access Manager in SP 800-131a mode.

For more information about SP 800-131a, see http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf.

Suite B

Suite B is a security standard that is developed by the National Security Agency (NSA) that establishes a cryptographic interoperability strategy. Suite B is similar to SP 800-131a, but it has tighter restrictions.

Suite B can run in two modes: 128-bit and 192-bit. To use the 192-bit mode, you must apply the unrestricted policy file to the JDK in the Security Access Manager Java components. When you apply the unrestricted policy, the JDK uses the stronger cipher that is required for the 192-bit mode.

Applying Suite B on the Security Access Manager Base components has the following prerequisites:

- TLS version 1.2 protocol for the SSL context
- Suite B-approved cipher suites
- Certificates:
  - 128-bit mode certificates must be signed with `SHA256withECDSA`.
  - 192-bit mode certificates must be signed with `SHA384withECDSA`.
- Ciphers:
  - SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  - `SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`

The Security Access Manager Base component communication uses certificates that are generated by the policy server. The strength and algorithms to create these certificates differ for each Suite B security mode. You cannot convert from the 128-bit mode to the 192-bit mode (or any other security mode) without completely regenerating all the Security Access Manager certificates. The certificates are not compatible with previous releases of Security Access Manager. Previous release Security Access Manager clients cannot communicate with the Security Access Manager 8.0 policy server in this mode.

For more information about the Suite B specifications, see http://www.nsa.gov/ia/programs/suiteb_cryptography/.

A requirement from the National Security Agency (NSA) to specify a cryptographic interoperability strategy. This standard is similar to SP800-131 with some tighter restrictions. Suite B can run in two modes: 128-bit or 192-bit. If you are using 192-bit mode with Security Access Manager Java applications, you must apply the unrestricted policy file to the JDK to use the stronger cipher that mode requires. Suite B requirements are:

- The use of TLSv1.2 protocol.
- Suite B approved Cipher suites
- Certificates:
  - 128-bit mode certificates must be signed with SHA256withECDSA
  - 192-bit mode certificates must be signed with SHA384withECDSA
- Ciphers:

- SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

Java properties that enable the security standards

The IBM® virtual machine for Java (JVM) runs in a specific security mode based on system properties.

When Security Access Manager Base components are run in a JVM other than WebSphere®, Security Access Manager automatically enables the appropriate system properties for the security configuration settings.

Table 1. JVM system properties enabled by Security Access Manager

Security standard	System property to enable	Valid values
FIPS 140-2	`com.ibm.jsse2.usefipsprovider` (for newer JVMs that support it) `com.ibm.jsse2.JSSEFIPS` (for older JVMs such as Java 5 and pre Java 1.6SR10)	`true` or `false`
SP 800-131a	`com.ibm.jsse2.sp800-131`	`transition` or `strict`
Suite B	`com.ibm.jsse2.suiteB`	`128` or `192`

Scalability

Scalability is the ability to respond to increasing numbers of users who access resources in the domain. Security Access Manager uses several techniques to provide scalability.

Security Access Manager provides the following scalability methods:

- Replication of services
  - Authentication services
  - Authorization services
  - Security policies
  - Data encryption services
  - Auditing services
- Front-end replicated servers
  - Mirrored resources for high availability
  - Load balancing client requests
- Back-end replicated servers
  - Back-end servers can be Security Access Manager WebSEAL or other application servers
  - Mirrored resources (unified object space) for high availability
  - Additional content and resources
  - Load balancing of incoming requests
- Optimized performance by allowing for the offloading of authentication services and authorization services to separate servers
- Scaled deployment of services without increasing management processor usage

Accountability

Security Access Manager provides several logging and auditing capabilities to increase accountability for server activity.

Log files capture any error and warning messages generated by Security Access Manager servers. Audit trail files monitor Security Access Manager server activity.

Centralized management

Security Access Manager uses two user interfaces to manage security policies and the Security Access Manager servers.

- **pdadmin** command-line interface
- Web Portal Manager graphical user interface (GUI). However, some tasks cannot be done with the Web Portal Manager.
- Administration API

You can accomplish most tasks by using any of these methods.

pdadmin command-line interface

The **pdadmin** command-line interface administers Security Access Manager.

This interface provides commands for managing users, groups, roles, permissions, policies, domains, and servers, and other tasks. This interface can be used in scripts or batch files to automate processing.

This interface is installed as part of the IBM Security Access Manager runtime package.

For specific task information, see the task-specific chapters in this guide. For detailed syntax information about the **pdadmin** command-line interface, see the Reference topics in the IBM Knowledge Center.

Web Portal Manager

Web Portal Manager is an optional web-based interface for administering or performing administrative tasks with Security Access Manager.

You can use Web Portal Manager to perform administrative tasks, such as managing users, groups, roles, permissions, policies, domains, and servers. A key advantage to using Web Portal Manager is that you can do these tasks remotely from any supported web browser. You do not need any special network configuration.

Administration API

You can use the administration API to write applications to manage users, groups, roles, permissions, policies, domains, and servers. Both C and Java language versions of these functions are available.

See details about the administration API:

- *Administration C API Developer Reference*: http://www.ibm.com/support/docview.wss?uid=swg21665268
- *Administration Java Classes Developer Reference*: http://www.ibm.com/support/docview.wss?uid=swg21665270

Security policy overview

The goal of any security policy is to adequately protect business assets and resources with a minimal amount of administrative effort. High-level steps include determining which resources to protect and the level of access that users get to those resources.

1. Define what resources need to be protected. Protected resources might be any type of data object, such as files, directories, network servers, messages, databases, or web pages.
2. Determine what users and groups of users can access to these protected resources. Also consider what type of access to these resources is permitted.
3. Apply the appropriate security policy on these resources to ensure that only the right users can access them.

The enforcement of the security policy is the job of the resource manager. The resource manager calls the Security Access Manager authorization service with the credentials of the user that makes the request. The call includes the type of access wanted and the object to be accessed. The credential provides detailed information, acquired during authentication, that describes the user, any group associations, and other security-related identity attributes. Credentials can be used to do a multitude of services, such as authorization, auditing, and delegation.

The authorization service is also called the authorization engine. The authorization service uses the security policy to determine whether the request is allowed or denied. The request might also be conditionally allowed pending additional verification by the resource manager. The resource manager takes the recommendation of the authorization service. The resource manager does any additional verification actions and ultimately either denies the request or permits the request to be processed.

For example, suppose that John wants to access a particular web page that is on a website protected by Security Access Manager WebSEAL. WebSEAL is a resource manager that manages and protects web-based information and resources. It must decide whether "John" can access that page. The resource manager obtains the credentials for John, and then asks the authorization service whether John has read access to the web page. The authorization service checks the security policy and determines that John is permitted access. The service responds to the resource manager that the request is granted. The resource manager then directs the request to the appropriate back-end web server, which provides the web page.

The security policy in Security Access Manager is defined through the use of access control lists (ACLs), protected object policies (POPs), and authorization rules.

Authorization API standard

Authorization services are a critical part of the security architecture of an application. After a user passes the authentication process, authorization services proceed to enforce the business policy by determining what services and information the user can access.

For example, a user might access a web-based retirement fund. The user can view personal account information after an authorization server verifies the identity, credentials, and privilege attributes of that user.

The standards-based authorization API (aznAPI) allows applications to call the centralized authorization service. The authorization API eliminates the necessity for developers to write authorization code for each new application.

The authorization API allows businesses to standardize all applications on a trusted authorization framework. With the authorization API, businesses can provide more control over access to resources on their networks.

Authorization: conceptual model

In security systems, authorization is distinct from authentication. *Authorization* determines whether an authenticated client has the required permissions to do a task on a specific resource in a domain. *Authentication* ensures that the individual is who that individual claims to be.

When servers enforce security in a domain, each client must provide proof of its identity. In turn, security policy determines whether that client has the required permission to do a task on a requested resource. Access to every resource in a domain is controlled by a server. The demands on the server for authentication and authorization can provide comprehensive network security.

In the Security Access Manager authorization model, authorization policy is implemented independently of the mechanism for user authentication. Users can authenticate their identity with a public/private key, secret key, or customer-defined mechanisms.

Part of the authentication process involves the creation of a credential that describes the identity of the client. Authorization decisions made by an authorization service are based on user credentials.

The resources in a domain receive a level of protection that is dictated by the security policy for the domain. The security policy defines the legitimate participants of the domain. It also defines the degree of protection that surrounds each resource that requires protection.

The authorization process, as shown in Figure 1 on page 10, includes the following basic components:

resource manager

Implements the requested operation when authorization is granted.

A component of the resource manager is a *policy enforcer* that directs the request to the authorization service for processing.

authorization service

Processes the decision-making action on the request.

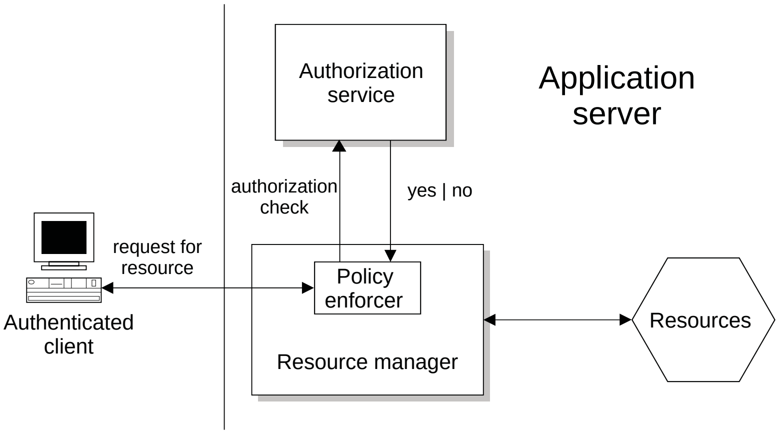


Figure 1. General authorization model

Traditional applications bundle the policy enforcer and resource manager into one process. An example of this structure is Security Access Manager WebSEAL.

The independent functions of these authorization components allow flexibility in the design of the security enforcement strategy.

For example, such independence allows the security administrator to control:

- Where the processes are located
- Who writes the code for the processes
- How the processes do their tasks

Benefits of a standard authorization service

A standard authorization service offers several benefits over an assortment of proprietary authorization implementations. Benefits can include reduced development cost and the ability to share information securely.

Authorization in most systems, both existing and new, is tightly coupled to individual applications. Companies typically build applications over time to serve their business needs. Many of these applications require some specific form of authorization.

The result is often a wide variety of applications with differing authorization implementations. These proprietary authorization implementations require separate administration, are difficult to integrate, and result in higher costs of ownership.

A distributed authorization service can provide these independent applications with a standard authorization decision-making mechanism. Benefits of such a standard authorization service include:

- Reduced cost of developing and managing access to applications
- Reduced total cost of ownership and management of separate authorization systems
- Use the existing security infrastructure
- Allow new businesses to open applications more securely
- Enable newer and different kinds of applications
- Allow shorter development cycles

- Share information securely

Security Access Manager authorization service overview

You can integrate Security Access Manager into existing and emerging infrastructures to provide a secure, centralized policy management capability. The authorization service, together with resource managers, provides a standard authorization mechanism for business network systems.

The Security Access Manager authorization service is illustrated in Figure 2.

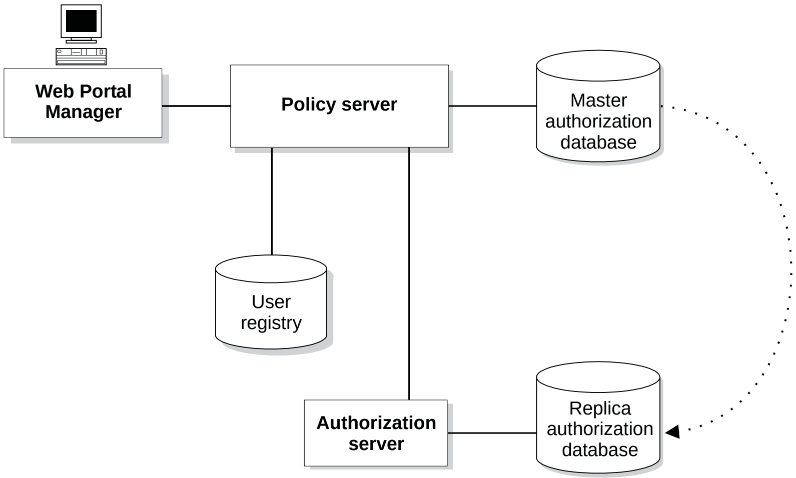


Figure 2. Security Access Manager server components

Existing applications can take advantage of the authorization service. An authorization policy is based on user or group roles. It can be applied to network servers, individual transactions, database requests, specific web-based information, management activities, and user-defined objects.

The authorization API allows existing applications to call the authorization service, which bases its decision on the corporate security policy. For more information about the authorization API, see "Security Access Manager authorization API" on page 17.

The Security Access Manager authorization service is also extensible. It can be configured to call on other authorization services for additional processing by using the external authorization service plug-in interface.

The authorization service provides the following benefits:

- The service is application independent.
- The service uses a standard authorization coding style that is language independent (the authorization API).
- The service is centrally managed and therefore easy to administer. The addition of a new employee, for example, requires modifying the privilege database in one central location, rather than across multiple systems.
- The service addresses the application of security services in a heterogeneous cross-platform environment.
- The service integrates existing non-Security Access Manager authorization systems through an external authorization service capability.

- The service has a scalable and flexible architecture that can be easily integrated with existing infrastructure.
- The service enables multi-tiered authorization. A credentials packet can be passed through the multiple layers of an application process or transaction.
- The service uses a common and effective auditing model.
- The service is independent of any authentication mechanism.

Security Access Manager authorization service

The authorization service is responsible for the authorization decision-making process that enforces a network security policy.

Authorization decisions made by the authorization service result in the approval or denial of client requests to do operations on protected resources in a domain.

Components

The authorization service is made up of the following basic components: the master authorization policy database, the policy server, and the authorization decision-making evaluator.

Policy database

The *policy database* contains the security policy information for all resources in a domain. The policy database is also called the master authorization policy database or the master authorization database.

Each domain has its own policy database. The contents of this database are manipulated by using the **pdadmin** command-line interface, Web Portal Manager, and the administration API.

Policy server

The policy server maintains the policy databases and replicates this policy information throughout the domains. The policy server also updates the database replicas whenever a change is made to the master.

The policy server also maintains location information about the other Security Access Manager and non-Security Access Manager resource managers that operate in the domain.

Authorization evaluator

The authorization evaluator is the decision-making process that determines the ability of the client to access a protected resource based on the security policy. The evaluator makes its recommendation to the resource manager, which responds.

User registry replication parameters are configurable for each evaluator.

Figure 3 on page 13 illustrates the main components of the authorization service:

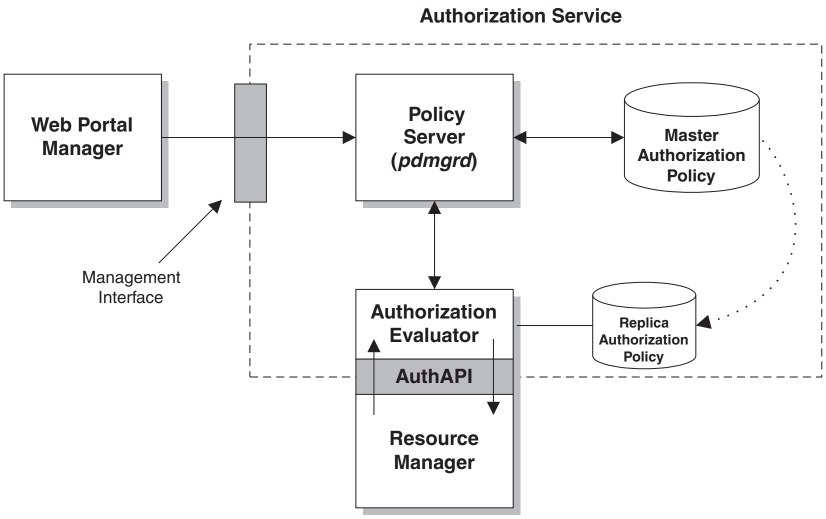


Figure 3. Authorization service components

Authorization service interfaces

Interaction in the *authorization service* occurs in the management interface and the authorization API.

Management interface

The security administrator manages the security policy with the Web Portal Manager or **pdadmin** command-line interface to apply policy rules to resources in a domain. The security policy is managed in the policy database by the policy server.

This interface is complex and involves detailed knowledge of the object space, policies, and credentials.

Authorization API

The authorization API passes requests for authorization decisions from the resource manager to the authorization evaluator. The authorization evaluator provides feedback on whether to grant or deny the request.

Replication for scalability and performance

You can replicate authorization service components to increase availability in a heavy-demand environment.

You can configure the master authorization policy database, containing policy rules and credential information, to automatically replicate. Resource managers that call the authorization service have two options for referencing this database information:

- The application, when configured to work seamlessly with the authorization evaluator, uses a local cache of the database.

  The database is replicated for each resource manager that uses the authorization service in local cache mode.
- The application uses a shared replica cached by the remote authorization server component.

The database is replicated for each instance of the authorization server. Many applications can access a single authorization server.

The update notification from the policy server occurs whenever a change is made to the master authorization policy database. The update notification triggers the caching process to update all replicas, as shown in Figure 4:

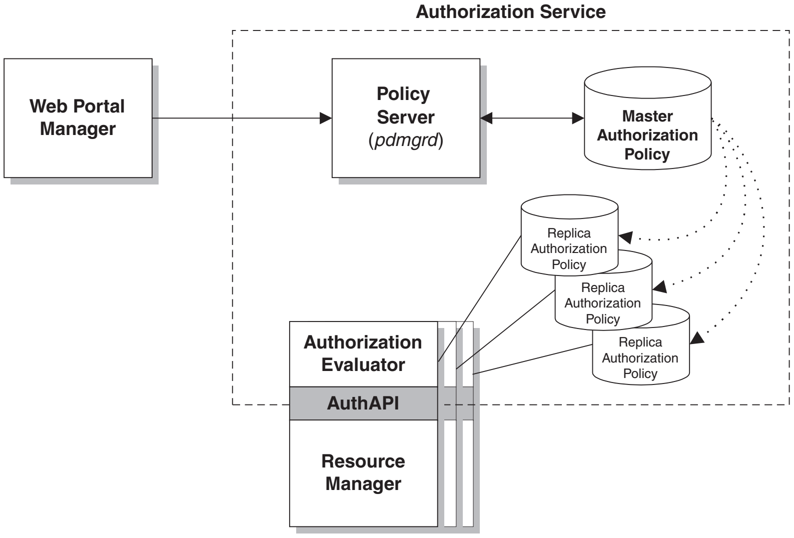

Authorization Service
Web Portal Manager
Policy Server (pdmgrd)
Master Authorization Policy
Replica Authorization Policy
Replica Authorization Policy
Replica Authorization Policy
Authorization Evaluator
AuthAPI
Resource Manager

Figure 4. Replicated authorization service components

Performance notes

Consider these performance issues:

- You can update notifications directly from the policy server. You can also configure the resource managers to verify the version of the master authorization policy database every few minutes. This verification ensures that the update notifications are not missed. Such a mechanism is called polling and is not enabled by default.

  If an update notification fails to reach a server, a log entry is created. In both cases, a retry mechanism also ensures that the update happens in the future.
- The cached authorization policy information results in high system performance. For example, when WebSEAL does an authorization check, it checks the policy in its own cached version of the database. WebSEAL does not have to access the network to obtain this information from the master database. The result is fast response times (performance) for authorization checks.
- Individual authorization results are not cached by the calling application server.

Implementation of a network security policy

Controlling user and group participation in the domain and applying rules to resources that require protection determine the security policy for a domain. These rules are defined by access control lists (ACLs), protected object policies (POPs), and authorization rules.

The authorization service enforces these policies by matching the credentials of a user with the permissions in the policy assigned to the requested resource. The resulting recommendation is passed to the resource manager, which completes the response to the original request.

Definition and application of security policy

You can protect system resources by defining a security policy. You define a security policy with access control lists (ACLs), protected object policies (POPs), and authorization rules. You apply the security policy to the object representations of those resources in the object space.

You can apply ACLs, POPs, and authorization rules to the same object. The Web Portal Manager, **pdadmin** command-line interface and the administration API are used to define this policy.

The authorization service makes authorization decisions based on the policies applied to these objects. When a requested operation on a protected object is permitted, the resource manager responsible for the resource implements this operation.

One policy can dictate the protection parameters of many objects. Any change to the security policy affects all objects to which the policy is attached.

Explicit and inherited policies

A security policy can be explicitly applied or inherited. The administrator can apply explicit policies only at points in the hierarchy where the rules must change.

The Security Access Manager protected object space supports inheritance of ACLs, POPs, and authorization rules. This factor is an important consideration for the security administrator who manages the object space. The administrator needs to apply explicit policies only at points in the hierarchy where the rules must change, as shown in Figure 5.

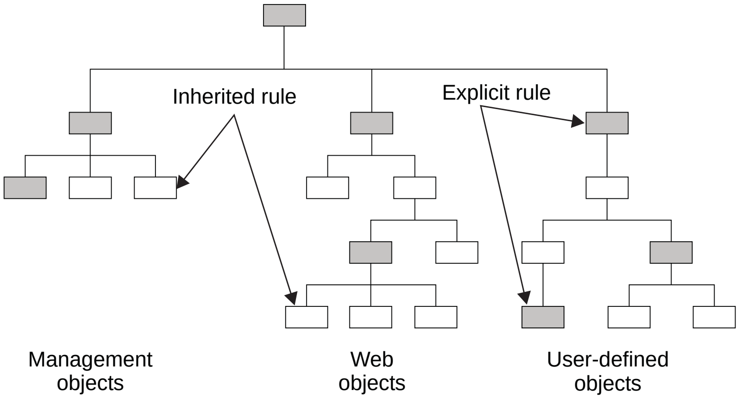


Figure 5. Explicit and inherited policies

Examples of policy types include:

- Hardcoded rules
- External authorization capability
- Special secure labeling
- Access control lists (ACLs), protected object policies (POPs), and authorization rules

Access control lists

An *access control list (ACL) policy* is a set of actions, controls, or permissions. The ACL policy specifies the necessary conditions for a user or group to do operations on a resource.

ACL policy definitions are important components of the security policy established for a domain.

An ACL policy specifically determines what operations can be done on a resource, and who can do those operations. An ACL policy is made up of one or more entries that include user and group designations and either their specific permissions or rights.

Protected object policies

Protected object policies (POPs) contain additional conditions that must be met before granting access to a user or group.

Unlike ACLs, which are dependent on what user or group is attempting the action, POPs affect all users and groups. POPs also indicate whether requests must be audited. It is the responsibility of Security Access Manager and the resource manager to enforce the POP conditions.

Authorization rules

Define authorization rules to specify additional conditions that must be met before granting access to a resource.

You can use rules to make authorization decisions based on the context and the environment that surround a request. You can also use rules to make authorization decisions based on who is attempting the access and what type of action is being attempted. These conditions are evaluated as a Boolean expression to determine whether the request must be allowed or denied.

The authorization process: step-by-step

This example illustrates how the authorization process works.

Figure 6 illustrates the complete authorization process.

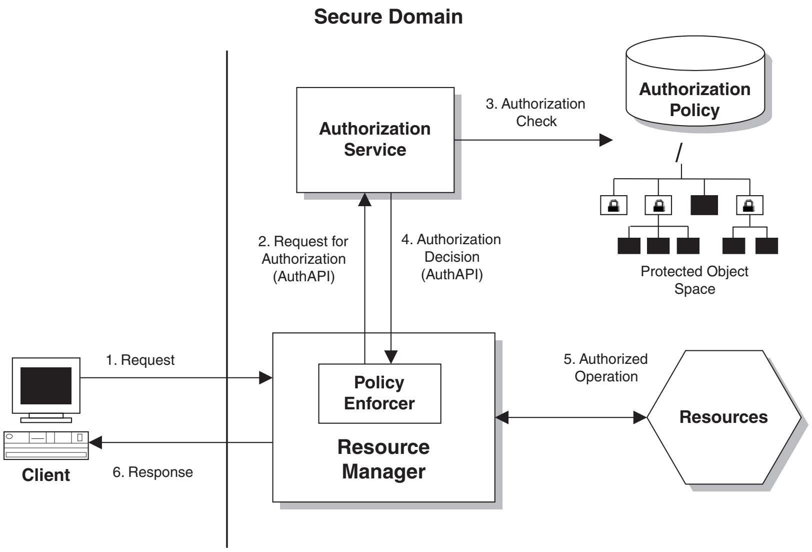


Figure 6. The Security Access Manager authorization process

1. An authenticated client request for a resource is directed to the resource manager server and intercepted by the policy enforcer process. For example, the resource manager can be WebSEAL for Hypertext Transfer Protocol (HTTP), HTTPS access, or another application.
2. The policy enforcer process uses the authorization API to call the authorization service for an authorization decision. For more information about the authorization API, see “Security Access Manager authorization API.”
3. The authorization service does an authorization check on the resource. See Authorization Algorithm for details on the algorithm used.
4. The decision to accept or deny the request is returned as a recommendation to the resource manager through the policy enforcer.
5. If the request is finally approved, the resource manager passes the request on to the application responsible for the resource.
6. The client receives the results of the requested operation.

Security Access Manager authorization API

The Security Access Manager authorization application programming interface (API) is the interface between the resource manager requesting the authorization check and the authorization service itself.

The authorization API allows Security Access Manager applications and other applications to query the authorization service to make authorization decisions. At the same time, the authorization API shields the application from the complexities of the actual decision-making process, including issues of management, storage, caching, replication, credential formats, and authentication methods.

The authorization API provides a standard programming model for coding authorization requests and decisions. You can use the authorization API to make standardized calls to the centrally managed authorization service from any existing or newly developed application.

The authorization API can be used in one of the following modes:

Remote cache mode

In this mode, the API is initialized to call the remote authorization server to do authorization decisions on behalf of the application. The authorization server maintains its own cache of the replica authorization policy database. This mode is best suited for handling authorization requests from application clients.

For more information about remote cache mode, see “Authorization API: remote cache mode” on page 19.

Local cache mode

In this mode, the API is initialized to download and maintain a local replica of the authorization database for the application. Local cache mode provides better performance because the application does all authorization decisions locally instead of across a network. However, the processor usage of database replication and the security implications of using this mode make it best suited for use by trusted application servers.

For more information about local cache mode, see “Authorization API: local cache mode” on page 19.

The authorization API also works independently from the underlying security infrastructure, the credential format, and the evaluating mechanism. The authorization API makes it possible to request an authorization check and get a simple `yes` or `no` recommendation in return. The details of the authorization check mechanism are invisible to the user.

Authorization API examples

Applications can use the authorization API to do access control on specific and specialized processes.

Example 1
You can design a graphical interface to dynamically show interface controls as active or inactive, according to the results of the authorization check.

Example 2
Figure 7 illustrates a request for a Common Gateway Interface (CGI) transaction by a web application.

The lowest level of authorization, as illustrated in Figure A of Figure 7, involves an "all-or-nothing" access control on the Uniform Resource Locator (URL). This coarse-grained level of authorization determines only whether the client can run the CGI program. If access is allowed to the CGI application, no further control is available to resources manipulated by the CGI application.

As illustrated in Figure B of Figure 7, access controls were set on resources that the CGI program manipulates. The web application is configured to use the authorization API. The CGI program can call the authorization service to make authorization decisions on the resources it manipulates based on the identity of the requesting client.

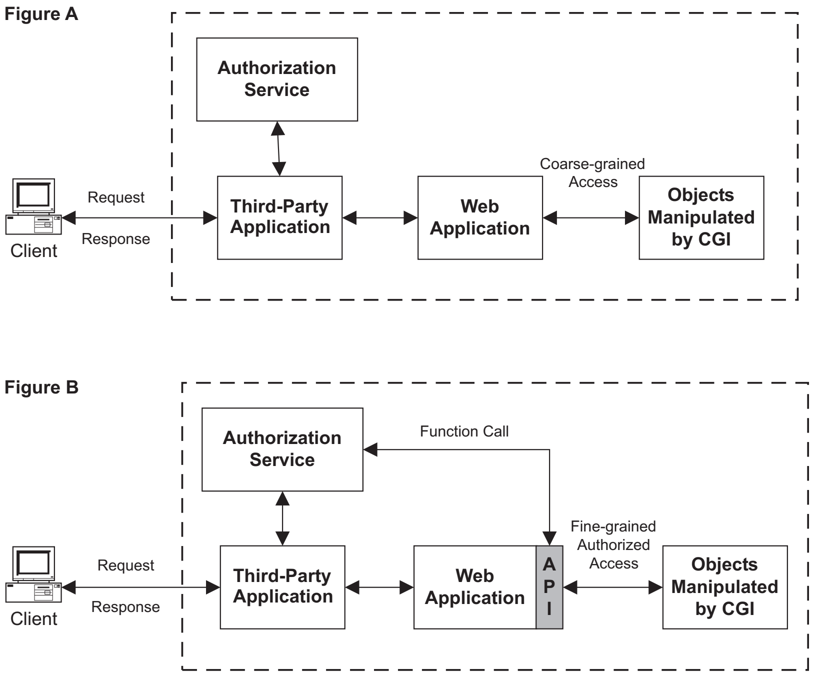


Figure 7. Example use of the authorization API

Authorization API: remote cache mode

In remote cache mode, resource managers use the function calls from the authorization API to communicate to the remote authorization server.

The authorization server functions as the authorization decision-making evaluator and maintains its own replica authorization policy database.

The authorization server decides and returns a recommendation to the application through the API. The server can also write an audit record that contains the details of the authorization decision request.

The remote cache mode requires an authorization server that runs in a domain, as shown in Figure 8. The authorization server can be on the same system as the application or on another system. You also can install the authorization server on more than one system in a domain for high availability. The authorization API transparently performs failover when a particular authorization server fails.

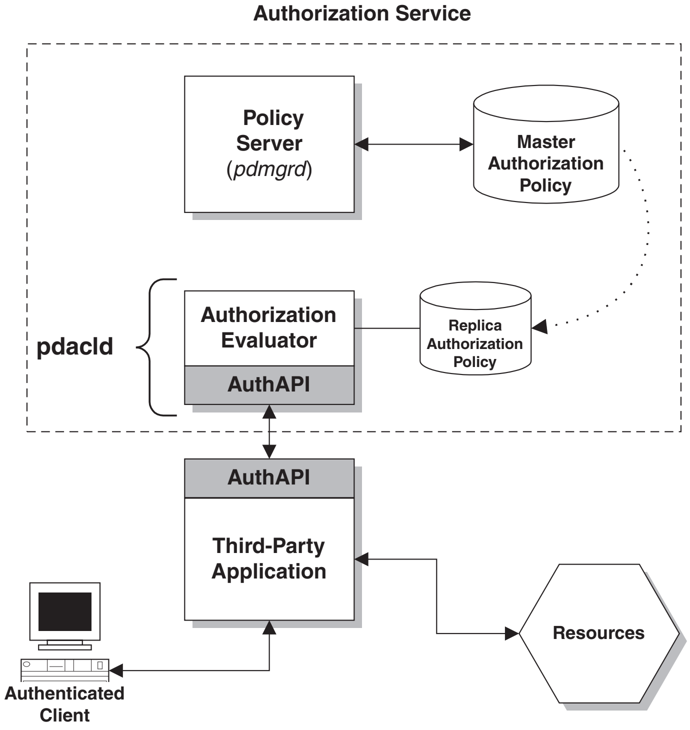


Figure 8. Authorization API: remote cache mode

Authorization API: local cache mode

In local cache mode, the API downloads and maintains a replica of the authorization policy database on the local file system of the resource manager. It makes all authorization decisions in-memory, which results in higher performance and better reliability.

The local replica is persistent across invocations of the application. When the API starts in replica mode, it checks for updates to the master authorization policy database that were made after the local replica was built.

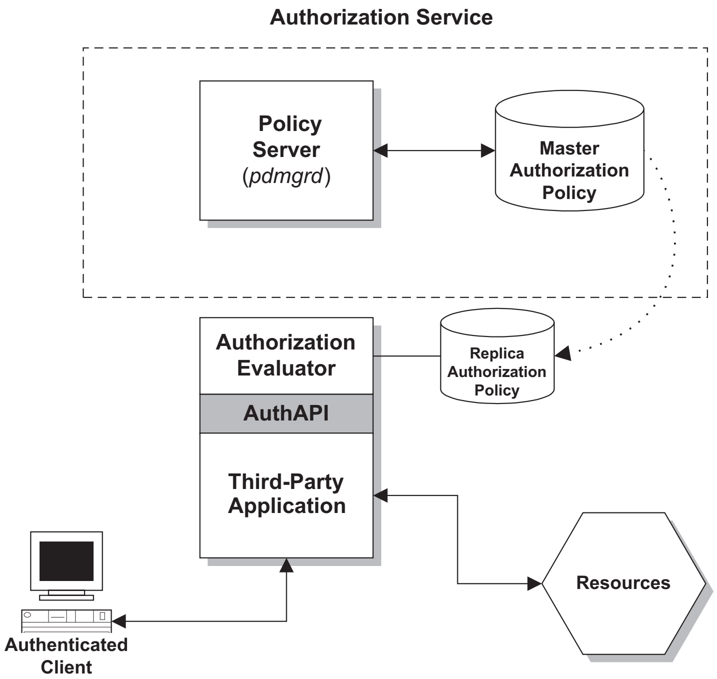


Figure 9. Authorization API: local cache mode

External authorization capability

Security Access Manager provides an optional external authorization capability to accommodate any additional authorization requirements.

In some cases, it might not be possible for the standard Security Access Manager policy implementations to express all conditions required by a security policy. The standard policy implementation uses ACLs, POPs, and authorization rules to manage access to resources.

You can use the external authorization service to impose additional authorization controls and conditions that are dictated by a separate, external, authorization service module.

External authorization service

External authorization capability is automatically built into the Security Access Manager authorization service. Resource managers can use an external authorization service to provide a seamless experience.

If you configure an external authorization service, the Security Access Manager authorization service incorporates the access decision paths into its evaluation process.

Resource managers such as WebSEAL benefit from the additional, seamless contribution of an external authorization service. Applications that use the authorization API also benefit. Any addition to the security policy through an external authorization service is not apparent to these applications. The addition requires no change to the applications.

The external authorization service architecture provides the full integration of an existing security service. An external authorization service preserves your investment in security infrastructure by incorporating existing servers into the Security Access Manager authorization decision-making process.

Application of specific conditions on resource requests

An external authorization service can impose more specific conditions or system-specific side effects on a successful or unsuccessful access attempt.

Examples of such conditions include:

- Causing an external auditing mechanism to record the successful or unsuccessful access attempt
- Actively monitoring the access attempt and causing an alert or alarm whenever unacceptable behavior is detected
- Conducting billing or micro-payment transactions
- Imposing access quotas on a protected resource

Authorization evaluation process

Security Access Manager uses a multi-step process to evaluate authorization requests.

An authorization decision that incorporates an external authorization server takes place in the following manner:

1. If a trigger condition is met during an access decision, the external authorization services that were configured for that condition are each called in turn. The external authorization services evaluate their own external authorization constraints.

   Invocation of the external authorization service occurs regardless of whether the necessary permission is granted to the user by the Security Access Manager authorization service.
2. Each external authorization service returns a decision of permitted, denied, or indifferent.

   When `indifferent` is returned, the external authorization service determined that its functionality is not required for the decision process and that it does not participate.
3. Each external authorization service decision is weighted according to the level of importance that its decision carries in the process.

   The weighting of individual external authorization services is configured when the service plug-in is loaded.
4. All authorization decision results are summed and combined with the decision made by the Security Access Manager authorization service. The resulting decision is returned to the caller.

Example of an external authorization service

In this example, the external authorization service imposes a quota restriction on how often a photo-quality printer resource can be accessed.

Figure 10 on page 22 illustrates an authorization decision that involves an application server and an external authorization service.

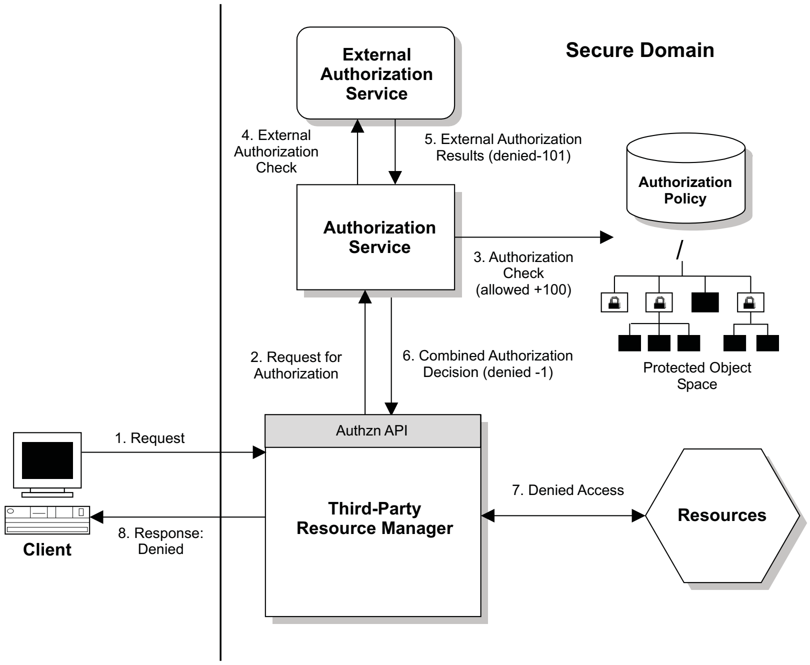

External Authorization Service
Secure Domain
4. External Authorization Check
5. External Authorization Results (denied-101)
Authorization Policy
Authorization Service
3. Authorization Check (allowed +100)
/
2. Request for Authorization
6. Combined Authorization Decision (denied -1)
Protected Object Space
Authzn API
1. Request
Third-Party Resource Manager
7. Denied Access
Resources
8. Response: Denied
Client

Figure 10. External authorization service with an application server

The service implementation imposes a limit on the number of job submissions that any one person can make to this printer in one week. A trigger condition is attached to the photo printer resource so that the external authorization service is called whenever the photo printer is accessed.

The external authorization service is loaded with the default decision weighting of 101. The default decision weighting overrides any decision made by the Security Access Manager authorization service if required.

1. The resource manager server receives a request from a client for access to an online photo printing resource. The client is a member of the appropriate group `GraphicArtists` and so is typically permitted to submit jobs to the printer.
2. The application server first consults the Security Access Manager authorization service to determine whether the requesting user has permission to submit jobs to the printer.
3. The authorization service verifies the access permissions on the target requested object and compares the permissions against the capabilities of the requesting user:

   ```
   group GraphicArtists rx
   ```

   In the ACL on the printer resource, the **x** permission grants any user in the `GraphicArtists` group access to the resource. Therefore, the authorization service grants the user permission to submit the job.
4. The photo printer resource is being accessed and an external authorization service trigger condition is attached to this object. A request is also made to the external authorization service configured for that trigger condition.

   The external authorization service receives all the Access Decision Information (ADI) that was passed in with the original access decision check by the resource manager server.
5. The external authorization service consults a record of previous accesses made by this user. If the requesting user is within the quota for the week, it returns an access decision of `indifferent`.

The external authorization service is indifferent to the request. The service does not participate in the access decision because the conditions for denying access are not present.

If the user exceeds the quota, then the external authorization service returns a decision of `access denied`.

For this example, the requester exceeds the quota. The external authorization service detects this problem and returns an `access denied` decision.

6. The Security Access Manager authorization service receives the `access denied` result from the external authorization service. It then takes this decision and weights it with the default external authorization service weighting value of 101.

   The results of the external authorization service decision and the decision made by the Security Access Manager authorization service are combined. The result is `access denied` because the result of the external authorization service (–101) outweighs that of the Security Access Manager authorization service (100).
7. The resource manager server rejects the job submission to the photo printer resource.
8. The resource manager server returns a response to the caller to indicate that the job was rejected.

The process of implementing an external authorization service

This process requires you to write a plug-in module for the external resource manager. You then register the external authorization service with the resource manager.

1. Write an external resource manager service plug-in module with an authorization interface that can be referenced during authorization decisions.
2. Register the external authorization service with the resource manager so that the resource manager can load the plug-in service at initialization time.

   Registering the service sets a trigger condition for the invocation of the external authorization service. When the trigger condition is encountered during an authorization check, the external authorization service interface is called to make an additional authorization decision.

Deployment strategies

You can deploy an external authorization service in several ways.

- Any number of external authorization services can be registered with resource manager applications. Applications that can load external authorization services include the authorization server, other Security Access Manager resource managers, and any other resource manager applications that you create.
- Remote-mode authorization API clients make requests to the authorization server for authorization decisions. These clients automatically use any external authorization service that is loaded by the authorization server.
- More than one external authorization service can be called for any single trigger condition. First, the result of each external authorization service is weighted. Then the results are combined with the result of the Security Access Manager authorization service.
- Trigger conditions can be placed on objects with the use of a POP trigger. If a trigger condition is placed on an object, then any request to that object instigates a call to the appropriate external authorization services.
- Trigger conditions can also be placed on the operations requested by a user. For example, an external authorization service can be triggered only when a user

requests a Write operation to a protected resource. In this case, the external authorization service is not triggered for any other operation. It is then possible to develop sets of operations for which one or more external authorization services are triggered according to the requested set of operations.

- The external authorization services are implemented as dynamically loadable library (dynamic link library (DLL) modules. This feature greatly simplifies the task of external authorization service development. There is no requirement to make remote requests to the external authorization service. The load of the call is equivalent to the load of a function call.
- The combination of the authorization API and an external authorization service provides a highly extensible and flexible solution for implementing a complex security policy.

Chapter 2. Web Portal Manager

Security Access Manager has both command-line and graphical interface interfaces for managing domains, users, groups, permissions, policies, and other resources in your enterprise.

The command line interface, **pdadmin**, is installed as part of the runtime package. The graphical user interface is a management console, called Web Portal Manager.

Although you can manage your enterprise through either interface, only a subset of the management tasks can be completed through Web Portal Manager. To compare the mapping between the **pdadmin** utility and Web Portal Manager tasks, see Appendix E, “pdadmin to Web Portal Manager equivalents,” on page 299.

Another difference between these interfaces is that when you use the **pdadmin** utility, you can specify a file. With Web Portal Manager, you cannot specify a file name. In some cases, however, you can copy and paste the contents of the file.

You can automate certain management tasks by writing scripts that use the **pdadmin** utility.

For more information about **pdadmin**, see the Command Reference topics in the Knowledge Center.

Online help

Instructions for completing tasks with Web Portal Manager are documented in the online help system. Use the help system if you have questions when you enter information in fields or select or clear choices.

To access the online help:

1. Use Web Portal Manager to log on to the domain.
2. Select a task such as **Group** > **Import Group**.
3. In the task title bar, click the question mark icon on the right side of the page. A help window contains the online information for completing the task.

Mitigating cross-site request forgery attacks

To help mitigate cross-site request forgery (CSRF) attacks in Web Portal Manager, a token has been added to certain Web Portal Manager requests.

This token modifies the URL to the Web Portal Manager web pages. An error is returned if the token is missing from the request or does not match the real session token.

A CSRF attack is a type of malicious web site attack that is sometimes called a one-click attack or session riding. This type of attack sends unauthorized requests from a user that the website trusts. CSRF uses the trust that a site has in the browser of an authenticated user for malicious attacks. CSRF uses links or scripts to send involuntary HTTP requests to a target site where the user is authenticated.

Chapter 3. Security Access Manager administration

Administering Security Access Manager includes tasks such as installing and configuring resource managers, defining users and groups, and implementing security policies.

The administration of Security Access Manager involves the following high-level tasks.

1. Create domains and subdomains for management purposes, as necessary. See Chapter 5, “Domain management,” on page 55.
2. Install and configure resource managers. During configuration, Security Access Manager resource managers and other components create a protected object space and protected resources, also known as protected objects.
3. Create additional object spaces, as needed, for management purposes. See Chapter 6, “Object space management,” on page 61.
4. Define protected objects in the object space, as needed, to represent the resources that are to be protected. For protected objects, you can define the following characteristics:
   - Who is allowed access.
   - What type of access is allowed.
   - When that access is allowed.
   - What other conditions that must be met before Security Access Manager allows access.
   - Whether the access request is audited.

   See Chapter 7, “Manage protected objects,” on page 67.
5. Define users and groups that require access to the protected resources. See Chapter 11, “Manage users and groups,” on page 145.
6. Implement your *security policy* by attaching the following elements to objects that are in the protected object space:
   - An access control list (ACL)
   - A protected object policy (POP)
   - An authorization rule

Domains

A *domain* consists of all the resources that require protection and the associated security policy used to protect those resources.

The resources that you can protect depend on the resource managers that are installed. These resources depend on which resource managers are installed. The resources can be any physical or logical entity, including objects such as files, directories, web pages, printer and network services, and message queues. Any security policy that is implemented in a domain affects only the objects in that domain. Users with authority to do tasks in one domain do not necessarily have the authority to do those tasks in other domains.

Security Access Manager creates a domain, called the *management domain*, as part of its initial configuration. The default name of this management domain is `Default`. It is in a stand-alone naming context, with a suffix called `secAuthority=Default`.

This domain is used by Security Access Manager to manage the security policy of all domains and is available for managing other protected resources as well. The administrator can rename the management domain and change its location when the policy server is configured.

For small and moderately sized enterprises, one domain is typically sufficient. If only one domain is needed, no explicit action needs to be taken.

In large enterprises, however, you might want to define two or more domains. Each domain is given a name and is established with a unique set of physical and logical resources. The security administrator can define the resources in a domain based on geographical area, business unit, or major organizational division within the enterprise. The security policy defined in the domain affects only the resources in that domain, which allows data to be partitioned and managed independently.

A multiple domain environment can be invaluable when there is a business need to keep a physical separation between different sets of data. The following other benefits are associated with using multiple domains:

Increased security

Security policy data for each domain is mutually exclusive. You cannot associate users, groups, and resources that are defined in a domain with another domain. For example, suppose that a user named John Doe is identified as `JohnDoe` in the `Sales` domain and as `JDoe` in the `Advertising` domain. Although the same person, each user ID is unique for each domain. As a result, resources that are available to user `JohnDoe` can be granted access by the unique ID by which the user is defined in that domain (`Sales`). In addition, user `JohnDoe` can be granted access in the `Sales` domain by the unique ID in the groups of which `JohnDoe` is a member. Likewise, user `JDoe` can be granted access only by the unique ID by which the user is defined in the `Advertising` domain.

Simplified administration

You can assign independent administrators to handle policy management tasks for each domain. For example, assume that you are an IT specialist for a large corporation. You are assigned to deploy Security Access Manager from a single data center. You can create a separate domain with a unique policy database and an administrator for each organization, division, or geographic area in your company. As users, groups, or resources change, the assigned administrator is responsible for updating the security policy for that particular domain. This domain administrator can also delegate administration tasks to others in that domain.

An administrator assigned to a specific domain has authority only in that domain. By default, an administrator can view users and groups defined in the user registry that are not necessarily Security Access Manager users or groups. This feature is beneficial if, for example, an administrator wants to import a user or group from a different domain. The administrator of the management domain can limit the registry data that a domain administrator can access. To do so, add the `allowed-registry-substrings` stanza entry to the `[domains]` stanza in the `ivmgrd.conf` configuration file for the policy server.

For more information about managing domains, see Chapter 5, "Domain management," on page 55.

Protected object space

Security Access Manager conceptualizes resources in a domain by showing a virtual representation called the *protected object space*. The protected object space is the logical and hierarchical portrayal of resources that belong to a domain.

The structure of the protected object space consists of the following types of objects:

Resource objects
The logical representation of actual physical resources in a domain, such as files, services, web pages, and message queues.

Container objects
Structural components that group resource objects hierarchically into distinct functional regions.

Security policy can be applied to both types of objects. Figure 11 shows a logical representation of a protected object space with multiple container and resource objects. This illustration shows container objects as white boxes and resource objects as gray boxes.

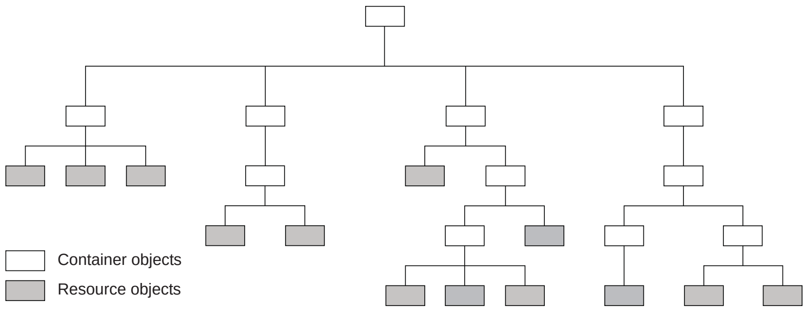


Figure 11. Security Access Manager protected object space

The structural top of the protected object space is the *root container object*. Below the root container object are one or more container objects. Each container object represents an object space that consists of a related set of resources. These resources can be resource objects or container objects.

The installation of Security Access Manager creates the `/Management` object space. This object space consists of the objects that are used to manage Security Access Manager itself. Under the `/Management` object space, the installation creates the following container objects:

- `/Users`
- `/Groups`
- `/POP`
- `/Action`
- `/ACL`
- `/GSO`
- `/Server`
- `/Config`
- `/Replica`

Figure 12 shows the complete /Management object space that is created during the installation of Security Access Manager.

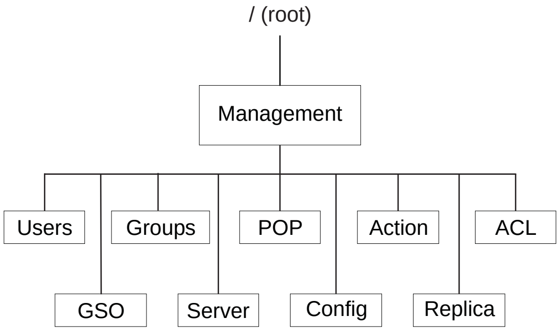


Figure 12. Regions of the Security Access Manager protected object space

Each resource manager that protects a related set of resources creates its own object space. For example, the installation of the WebSEAL component creates the /WebSEAL object space.

Users and groups

Security Access Manager maintains information about its users and groups in the user registry.

If you have a user registry that maintains users and groups for another application, you can import this user registry information into Security Access Manager. If a required user or group was not in the user registry before it was imported into Security Access Manager or a new user or group needs to be added to the Security Access Manager user registry, you can create it using Security Access Manager.

Basic users, or users in the registry that are not imported to the Security Access Manager, are supported. For more information, see Configuring the runtime to authenticate basic users.

Security Access Manager supports two types of group definitions. The most common type of group maintains the group membership as an explicit list of members (users). This type of group is sometimes called a static group, because the membership is listed and maintained.

For Active Directory and LDAP registry users, Security Access Manager also supports the use of dynamic groups. *Dynamic groups* are groups whose members are automatically resolved when the group is accessed. This resolution is based on the results of a defined search filter. For example, you create a dynamic group for members of department XYZ. If you import a new user whose data matches an entry in the search filter, the user is automatically added to the group. If an existing employee switches departments, the user is automatically removed from the group. Manual intervention is not required.

The creation and management of a dynamic group can be complex and is specific to the vendor implementation. It requires a search-like filter to be specified and used for group membership resolution. Because of these variables, dynamic groups cannot be created or maintained with Security Access Manager utilities or user interfaces. The vendor-specific tools must be used to create and maintain dynamic groups. Security Access Manager, however, can import and use these dynamic groups after they are created.

Security Access Manager supports different types of users. When a domain is created, a special user known as the *domain administrator* is created. For the management domain, the domain administrator is **sec_master**. The **sec_master** user and associated password are created during the configuration of the Security Access Manager policy server. For other domains, the user ID and password of the domain administrator are established when the domain is created. The domain administrator has nearly complete control of the domain. Think of the domain administrator as the Security Access Manager equivalent to the Linux or UNIX root account or the Microsoft Windows Administrator user.

The domain administrator is added as a member of the Security Access Manager **iv-admin** group within the domain. The **iv-admin** group represents those users with domain administration privileges. When adding users to the **iv-admin** group, ensure that you do not compromise the security of your domain.

Security policy

Attaching a *security policy* to objects in the protected object space controls access to objects in a domain. After attaching a security policy to an object, any change to the security policy is reflected immediately throughout the domain.

Each security policy can be defined with a combination of the following controls:

Access control list policies
An access control list (ACL) policy specifies the set of predefined actions that a set of users and groups can do on an object. For example, a specific set of groups or users can be granted read access to an object.

Protected object policies
A protected object policy (POP) specifies access conditions that are associated with an object. A POP affects all users and groups. For example, a time-of-day restriction can be placed on an object that excludes all users and groups from accessing that object during the specified time.

Authorization rules
An authorization rule specifies a complex condition that is evaluated to determine whether access is permitted. The data that determines whether access is permitted can be based on the context of the request, the current environment, or other external factors. For example, it can deny a request to modify an object more than five times in an eight-hour period.

A security policy can be explicitly applied to an object or can be inherited by an object that is above it in the hierarchy. Apply an explicit security policy in the protected object space only at those points in the hierarchy where the security policy must change.

You can implement a security policy by strategically attaching ACL policies, POPs, and authorization rules to objects that require protection. The Security Access Manager authorization service decides whether to allow or deny access to objects based on several criteria. One criterion is the credentials of the user that is making the request. Other criteria include the specific permissions and conditions that are specified in the ACL policies, POPs, and authorization rules.

The authorization service uses the following algorithm to process the security policy that is attached to a protected object:

1. Check permissions in the ACL policy to determine whether the user can override the attached POP or authorization rule. See "ACL policies" for information about the evaluation process.
2. When there is an authorization rule attached and the user cannot override it, gather the Access Decision Information (ADI).
3. When there is a POP attached:
   a. Check the Internet Protocol (IP) endpoint authentication method policy.
   b. Check the time-of-day policy.
   c. Check the audit-level policy and audit the access decision.
4. When an authorization rule is attached and the user cannot override the authorization rule, check the authorization rule policy.
5. When an external authorization service (EAS) operation or a POP trigger applies to this access decision, call the EAS.

If any of the ACL policy, POP, or authorization rule evaluations fail, then the access request is denied. The EAS can override this decision on its own if it is configured to do so.

ACL policies

The *ACL policy* defines who has access to and what operations can be performed on the object.

Each ACL policy has a unique name and can be applied to multiple objects within a domain.

An ACL policy consists of one or more of the following entry descriptions:

- The names of users and groups whose access to the object is explicitly controlled
- The specific operations that each user, group, or role can do
- The specific operations that the special **any-other** and **unauthenticated** user categories can do

Use of ACL policies with the authorization service

Security Access Manager relies on ACL policies to specify the conditions for a particular user to do an operation on a protected object. When you attach an ACL policy to an object, entries in the ACL specify what operations are allowed on it. The entries in the ACL policies also specify who can do the operations.

Web Portal Manager can group ACL policies based on:

- Use of policies in a particular part of the object space such as WebSEAL
- Use across the entire object space such as Base or Generic

Security Access Manager uses a default set of actions that cover a wide range of operations. Actions, or permissions, are represented by single alphabetic ASCII characters (a-z, A-Z). Each permission is displayed by the **pdadmin** utility with a label that describes the operation it governs.

A resource manager software typically contains one or more operations that are done on protected resources. Security Access Manager requires that resource managers make calls to the authorization service before the requested operation is allowed to progress. This call is made through the authorization application programming interface (authorization API) for both Security Access Manager services and other applications.

The authorization service uses the information contained in the ACL entry to make a simple “yes” or “no” response to the following question:

> Does this user or group have the appropriate permission to do the requested operation on the requested object? For example, does the user have the view (**r**) permission to view an object?

The authorization service has no knowledge about the operation that requires the read (**r**) permission. It merely notes the presence or absence of the **r** action bit in the ACL entry of the requesting user or group.

The authorization service is independent of the requested operations. This independence is why you can extend the benefits of the authorization service to other applications.

Evaluation of ACL policies

Security Access Manager follows specific steps to evaluate the permissions granted to a particular user by an ACL policy. Understanding how Security Access Manager evaluates ACL policies can help you determine how best to prevent unauthorized users from gaining access to resources.

- Evaluate authenticated requests

  Security Access Manager evaluates an authenticated user request by matching attributes of the user that is requesting access with criteria defined in the ACL entries.

  Security Access Manager evaluates an authenticated user request in the following order:

  1. Match the user ID with the user ACL entries. The permissions granted are the permissions in the matching entry.
     Successful match
         Evaluation stops here.
     Unsuccessful match
         Continue to the next step.
  2. Determine the groups to which the user belongs and match group ID with the group ACL entries. If more than one group entry is matched, the resulting permissions are a logical “or” operation (most permissive) of the permissions granted by each matching entry.
     Successful match
         Evaluation stops here.
     Unsuccessful match
         Continue to the next step.
  3. Grant the permissions of the **any-other** entry, if it exists.
     Successful match
         Evaluation stops here.
     Unsuccessful match
         Continue to the next step.
  4. An implicit **any-other** entity exists when there is no **any-other** ACL entry. This implicit entry grants no permissions.
     Successful match
         No permissions granted. End of evaluation process.

- Evaluate unauthenticated requests

  Security Access Manager evaluates an **unauthenticated** user by granting the permissions from the **unauthenticated** ACL entry.

  The **unauthenticated** entry is a mask (a bit-wise "and" operation) against the **any-other** entry when permissions are determined. A permission for **unauthenticated** is granted only if the permission also is defined in the **any-other** entry.

  Because **unauthenticated** depends on **any-other**, it makes little sense for an ACL entry to contain **unauthenticated** without **any-other**. If an ACL entry contains **unauthenticated** without **any-other**, the default response is to deny permissions to **unauthenticated**.

Protected object policies

A *protected object policy (POP)* specifies a security policy that applies to an object regardless of which user or which operation is done. A POP imposes access conditions on an object based on the time of the access. A POP also indicates whether the access request must be audited.

Each POP has a unique name and can be applied to multiple objects within a domain.

You can apply the following conditions on an object:

- POP attributes, such as warning mode, audit level, and time-of-day.

  More details about these attributes are in "Configure POP attributes" on page 110.
- Authentication strength POP (step-up).

  More details about this policy are in "Step-up authentication" on page 114.
- Quality of Protection POP.

  More details about this policy are in "Set a Quality of Protection level" on page 113.
- Network-based authentication POP.

  More details about this policy are in "Network-based authorization policy" on page 109.

Authorization rules

An *authorization rule policy* specifies which security policy applies to an object based on various conditions, such as context and environment.

Each authorization rule policy has a unique name and can be applied to multiple objects within a domain.

You define authorization rules in a way similar to definitions of ACL policies and POPs. You specify conditions that must be met before access to a protected object is permitted. You create an authorization rule with a number of conditions. These conditions are based on data supplied to the authorization engine in the user credential from several sources. The conditions can be based on data from the resource manager application and from the encompassing business environment. These conditions are evaluated as a Boolean expression to determine whether access to the object must be granted or denied.

You can work with complex, structured data by using the language of an authorization rule. You can examine values in the rule data and make informed access decisions. You can define the data for an access decision statically within the system or during a business process. Authorization rules provide the flexibility of a policy defined by an external authorization service. Unlike an external authorization service, you do not have to build an external authorization service into a shared library plug-in to use the authorization rules.

How authorization rules differ

ACL policies use a predefined set of operations to control which users and groups have permission to do operations on a protected object. Rules decide whether to grant access based on the attributes of a user or object and the context and environment.

For example, the ability of a user to read data associated with an object is either granted or denied by an ACL policy. POPs apply to all users and groups and control conditions that are specific to a particular protected object. For example, time-of-day access excludes all users and groups from accessing an object outside of the times set in the time-of-day policy.

Unlike ACL policies, authorization rules determine whether to allow access based on the attributes of a person or object. The authorization rules also take into account the context and environment that surrounds the access decision. For example, you can use a rule to implement a time-of-day policy that depends on the user or group. You can use an authentication rule to extend the controls provided by the ACL policies to implement a more advanced policy. For example, you can develop a policy based on quotas.

An ACL policy can grant a group permission to write to a resource. A rule can extend the policy. For example, a rule can evaluate whether a group exceeds a specified quota before it permits the group to write to a resource.

When to use authorization rules

In the authorization process, the entire security policy (ACL policies, POPs, and authorization rules) must permit access to the protected object before access is granted. Authorization rules provide the flexibility to extend an ACL policy or POP by tailoring the security policy to your needs.

Authorization rules can extend a policy implemented by other Security Access Manager policy types. These rules are not merely extensions of the existing policy types. An authorization rule is a policy type that is robust enough to replace the ACL policy and POP. Using ACL policies and POPs generally provides better performance. Use an authorization rule to complement these policies instead of replacing them.

Guidelines for a secure object space

To configure a secure object space, use these guidelines.

- Set high-level security policy on container objects at the top of the object space. Set exceptions to this policy with explicit ACL policies, POPs, and authorization rules on objects that are lower in the hierarchy.
- Arrange your protected object space so that most objects are protected by inherited, rather than explicit, ACL policies, POPs, and authorization rules.

Reduce the risk of an error that might compromise your network by simplifying the maintenance of your tree. An inherited security policy lowers maintenance because it reduces the number of ACL policies, POPs, and authorization rules that you must maintain.

- Position new objects in the tree where they inherit the appropriate permissions.

  Arrange your object tree into a set of subtrees, where each subtree is governed by a specific access policy. You determine the access policy for an entire subtree by setting explicit ACL policies, POPs, and authorization rules at the root of the subtree.
- Create a core set of ACL policies, POPs, and authorization rules, and reuse these policies wherever necessary.

  ACL policies, POPs, and authorization rule policies are a single-source definition. Any modification to the policy impacts all objects associated with the ACL policy, POP, or authorization rule.
- Control user access through the use of groups.

  It is possible for an ACL policy to consist of only group entries. Individual user entries are not required in the ACL policy when the users can be categorized into groups instead. Authorization rules can also be written to consider any group memberships of an individual rather than the individual specifically. This feature can reduce the complexity of the rule logic considerably.

  Access to an object by individual users can be efficiently controlled by adding users to or removing users from these groups.

Chapter 4. Default security policy

Security Access Manager establishes a default security policy to protect all objects in a domain. A set of administrative users and groups is established and granted a predefined set of permissions. This chapter describes the default security policy.

Default administration users and groups

At installation, Security Access Manager provides several important administration groups. By default, these users and groups are given special permissions to control and manage all operations in a domain. The access control lists (ACLs) created during configuration define this default security policy.

The following sections detail the specific roles assigned to each of these users and groups at installation time. The sections explain how to create administration users.

iv-admin group

This group represents the administrator group. All members of this group are considered administrators of the domain by the default policy.

You can easily place users into an administration role by adding them to the **iv-admin** group. There is a danger in this procedure when a user becomes a member of this group with the default ACLs. The user immediately has full rights to do administration operations on any object in the protected object space.

When the policy server is configured, the administrator (**sec_master**) user is created and added to the **iv-admin** group. It is the combination of group memberships that grants **sec_master** complete rights for all operations within the management domain but only within the default policy. The **sec_master** user does not have rights to new groups created outside of the default policy unless it is added as a user or a member of a group.

sec_master user

The **sec_master** user is created when Security Access Manager is initially installed and configured. The default policy makes the **sec_master** user a member of the **iv-admin** group, permitting it to do all actions within Security Access Manager.

Think of this account as the equivalent of the Linux or UNIX root account, or a member of the Microsoft Windows Administrator group.

ivmgrd-servers group

The **ivmgrd-servers** group contains the policy servers and the policy proxy servers. By default, members of this group are authorized to delegate requests to other Security Access Manager servers on behalf of the requestor.

Administration users

You can create administration accounts with varying degrees of responsibility. Responsibility is delegated to administrators through strategically placed administration ACLs.

The following list illustrates possible administration roles:

Security policy administrator

Security policy administrators are responsible for defining and organizing security policy in a domain. The administrator needs to be able to create, modify, and delete security policy. To do these tasks, these administrators need the following permissions on the `/Management/ACL`, `/Management/POP`, and `/Management/Rule` resources:

- Traverse (**T**)
- Browse (**b**)
- View (**v**)
- Modify (**m**)
- Delete (**d**)

These administrators need the following permissions to navigate their subtree of protected resources:

- Traverse (**T**)
- Browse (**b**)
- View (**v**)

These administrators need the following permission to ability to attach and detach a security policy to the same subtree:

- Attach (**a**)

These administrators must have the following permissions so as not to be affected by security policies that apply to all users for the same subtree.

- Bypass POP (**B**)
- Bypass rule (**R**)

Protected resource administrator

Protected resource administrators are responsible for adding and removing user access to one or more protected resources. These tasks include:

- Adding users to and removing users from groups that are defined in the security policy
- Adding permissions to and removing permissions from resources

These administrators need the following permissions on the `/Management/Groups` protected resource or on the individual groups that are defined in the `/Management/Groups` subtree:

- Traverse (**T**)
- Browse (**b**)
- View (**v**)
- Add (**A**)

Deployment administrator

Deployment administrators are responsible for installation and configuration of the resource managers in the domain.

These administrators need the following permissions on the `/Management/Server` protected resource:

- Traverse (**T**)
- Browse (**b**)
- View (**v**)
- Modify (**m**)
- Delete (**d**)

These permissions give the ability to configure resource managers into and out of the domain and update their configuration. See "Permissions attribute" on page 75.

Definition and application of security policy

Security administrators protect system resources by defining a security policy. A security policy consists of the access control list (ACL) policies, protected object policies (POPs), and authorization rules. You can apply these policies and rules to the object representations of the system resources to be protected in the object space. You can apply ACL policies, POPs, and authorization rules to the same object.

The authorization service makes authorization decisions based on the policies applied to these objects. When a requested operation on a protected object is permitted, the resource manager responsible for the resource implements this operation.

One policy can dictate the protection parameters of many objects. Any change to an ACL policy, POP, or authorization rule affects all objects to which the policy is attached.

ACL policies

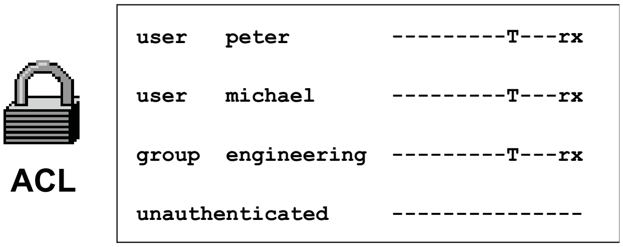
An *ACL policy* is the set of controls (permissions) that specifies the necessary conditions to do certain operations on that resource.

ACL policies are important components of the security policy that is established for the domain. ACL policies, like all policies, are used to stamp the set of security standards for an organization on the resources that are represented in their protected object spaces.

An ACL policy provides the following controls:

- What operations can be done on an object or resource
- Who can do an operation

An ACL policy is made up of one or more entries that include user and group designations and their specific permissions.



```
user   peter        ---------T---rx

user   michael      ---------T---rx

group  engineering  ---------T---rx

unauthenticated     ---------------
```

Figure 13. ACL policy

Protected object policies

ACL policies provide the authorization service with information that results in a `yes` or `no` answer on a request to access a protected object and do some operation on that object.

In contrast to ACL policies, *protected object policies (POPs)* contain additional conditions on the request. The conditions are passed back to Security Access Manager and the resource manager. These conditions are passed along with the `yes` ACL policy decision from the authorization server. It is the responsibility of Security Access Manager and the resource manager to enforce the POP conditions.

The following table lists the available attributes for a POP that are provided by Security Access Manager.

Table 2. POP attributes that are enforced by Security Access Manager

POP attribute	Description
Name	Name of the policy. This attribute relates to the *pop-name* variable in the **pop** command documentation.
Description	Descriptive text for the policy. This attribute appears in the **pop show** command.
Warning mode	Provides administrators a means to test ACLs, POPs, and authorization rules. Warning mode provides a way to test security policy before they are made active.
Audit level	Specifies the type of auditing: all, none, successful access, denied access, or errors. Audit level informs the authorizations service that extra services are required when permitting access to the object.
Time-of-day access	Day and time restrictions for successful access to the protected object. Time-of-day places restrictions on the access to the object.
IP endpoint authorization method policy	Specifies authorization requirements for access from members of external networks. IP endpoint authorization method policy places restrictions on the access to the object.
EAS trigger attributes	Specifies an External Authorization Service (EAS) plug-in that is started to make an authorization decision with the externalized policy logic of the customer.
Quality of Protection	Specifies degree of data protection: none, integrity, or privacy. Quality of Protection informs the authorizations service that extra services are required when permitting access to the object.

Although Security Access Manager provides these POP attributes, it enforces only the following attributes:
- Name
- Description
- Warning mode
- Audit level
- Time-of-day access

Each resource manager or plug-in can optionally enforce one or more of the following attributes:
- IP endpoint authorization method policy
- EAS trigger attributes
- Quality of Protection

The concept of inherited, or sparse ACLs as described in "Sparse security policy model" on page 41 also applies to POPs.

Authorization rules

An *authorization rule* specifies the policy that applies to an object and that is based on various conditions, such as context and environment. Each authorization rule has a unique name and can be applied to multiple objects in a domain.

Like ACL policies and POPs, authorization rules are defined to specify conditions that must be met before access to a protected object is permitted. An authorization rule is created with a number of Boolean conditions. The conditions are based on data that is supplied to the authorization service in the user credential. Data might also be supplied from the resource manager or from the encompassing business environment. The language of an authorization rule allows customers to work with complex, structured data, by examining the values in that data, and making informed access decisions. This information can be defined statically in the system or defined during a business process. Authorization rules can be used to implement extensible attribute-based authorization policy with attributes in the business environment or attributes from trusted external sources.

The authorization 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 similar constraints as ACL policies and POPs.

Sparse security policy model

To secure network resources in a protected object space, each object must be protected by security policy.

You can assign security policy to an object in one of following ways:

- Attach an explicit security policy on the object.
- Allow the object to inherit its security policy from a preceding container object in the hierarchy.

Adopting an inherited security scheme can greatly reduce the administration tasks for a domain. This section describes the concepts of inherited, or sparse security policies.

Security policy inheritance

Security policy inheritance simplifies the task of setting and maintaining access controls on a large protected object space.

The power of security policy inheritance is based on the following principle:

> Any object without an explicitly attached security policy inherits the policy of its nearest container object with an explicitly set security policy. The inheritance chain is broken when an object has an explicitly attached security policy.

In a typical object space, you need to attach only a few security policies at key locations to secure the entire object space. Therefore, it is called a sparse security policy model.

A typical object space begins with a single explicit security policy attached to the root container object. The root ACL must always exist and can never be removed. Normally, the root ACL is an ACL with little restriction. All objects in the object space inherit this ACL.

When a region or subtree in the object space requires different access control restrictions, you attach an explicit security policy at the root of that subtree. This attachment interrupts the flow of inherited security policies from the primary object space root to that subtree. A new chain of inheritance begins from this newly created explicit security policy.

default-root ACL policy

During the installation and initial configuration of Security Access Manager, the ACL policy for the entire object space is created and explicitly set.

This ACL policy is the default-root ACL policy and includes the following users and permissions:

```
group iv-admin              TcmdbvaBR
any-other                   T
unauthenticated             T
```

Security Access Manager checks inheritance beginning with the root of the protected object space. If you do not explicitly set an ACL policy on any other object in the tree, the entire tree inherits this root ACL policy.

There is always an explicit ACL policy set at the root of the protected object space. An administrator can replace this ACL policy with another ACL policy that contains different entries and permission settings. However, the administrator cannot completely remove the root ACL policy. See “Permissions attribute” on page 75.

Control permission

The control (**c**) permission gives you ownership of an ACL policy. As owner, you can modify entries in the ACL policy. Being able to modify entries in the ACL policy means that you can create entries, delete entries, grant permissions, and take away permissions.

The administrator who wants to delete a permission from an ACL policy must have an entry in that ACL policy. The administrator must also have the control permission set in that entry.

With control permission, you can grant administration powers to another user, such as the ability to attach or detach that ACL policy to objects. You must use the control permission with great care because of its powerful ownership properties.

Traverse permission

The traverse permission (**T**) specifies that a user or group that is identified in the ACL entry has permission to pass through this container object to gain access to a protected resource.

Security Access Manager access control depends on the following conditions:

- The permission that controls the requested object must contain appropriate access permissions for the requesting user.
- The requested object must be accessible to the requesting user. Accessibility to protected objects is controlled by the traverse (**T**) permission.

The traverse permission is applied only to container objects in the protected object space.

If there are no permissions defined for a user, that user cannot even traverse the root container object. This user cannot gain access at all to the protected object space, regardless of any permissions that might be granted lower in the tree.

A protected object is accessible if the requester possesses the traverse permission on each ACL attached to container objects above the requested resource on the path towards root and including root.

Figure 14 illustrates how the traverse permission works. Within the fictional ACME Corporation, there is an Engineering container object (directory), which contains a TechPubs directory. Kate (user kate) is a member of the Sales department and requires traversing to the Engineering/TechPubs/ directory tree to review a release note file (release_note). The administrator provides traverse for **any-authenticated** at the root. The administrator provides traverse permission for group sales on the Engineering directory. The TechPubs directory inherits the ACL from the Engineering directory. Although Kate has no other permissions in these two directories, she can pass (traverse) through these directories to access the required file. Because this file has read permission for Kate, she can view the file.

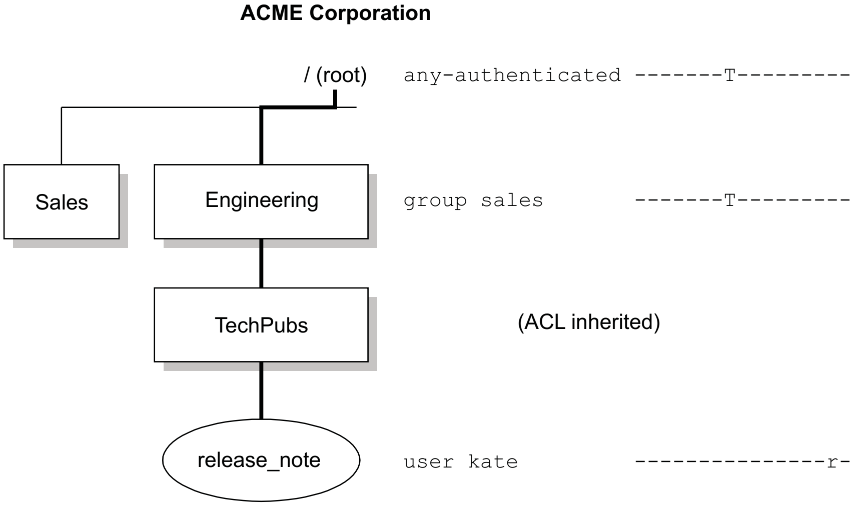

ACME Corporation
/ (root)
any-authenticated -------T---------
Sales
Engineering
group sales -------T---------
TechPubs
(ACL inherited)
release_note
user kate ---------------r-

Figure 14. Traverse permission

You can easily restrict access to the hierarchy below a specified container object without resetting individual permissions on these objects. Delete the traverse permission from the appropriate ACL policy. Deleting traverse permission on a directory object protects all objects lower in the hierarchy, even if those objects have other less restrictive ACL policies.

For example, sales group does not have the traverse permission on the Engineering directory. User kate cannot access the release_note file even though the user has read permission for that file.

Resolution of an access request

Inheritance begins at the root of the protected object space. Inheritance impacts all objects in the object space until it reaches an object with an explicit ACL policy. At this point, a new chain of inheritance begins.

Objects below an explicitly set ACL policy inherit the new ACL policy. If you delete an explicit ACL policy, permission for all objects reverts to the nearest container object with an explicitly set ACL policy.

When a user tries to access a protected object such as a document, Security Access Manager checks whether that user has the permissions to access the object. Security Access Manager checks each object along the object hierarchy for the inherited or explicitly set permissions. A user is denied access to an object if any container object in the hierarchy above the protected object does not include the traverse permission for that user. Access is denied if the target object does not contain sufficient permissions to do the requested operation.

To succeed an access check, the requester must have both of the following permissions:

- Permission to traverse the path to the requested object
- Appropriate permissions on the requested object

For example, to determine whether a user can read the `report.html` resource in the `/acme/engineering/project_Y/current/` object, Security Access Manager does the following checks:

1. Whether traverse permission is set on the root (/).
2. Whether traverse permission is set on the `acme`, `engineering`, `project_Y`, and `current` directories.
3. Whether read permission is set on the `report.html` file.

If any of these checks fail, the user is denied access.

Application of ACL policies to different object types

You can set permissions for various operations in an ACL policy. Only a subset of these possible operations might be relevant for a specific object to which the ACL policy is attached.

The reason for this behavior is related to the following Security Access Manager features that are designed to make administration easier:

- ACL policies
- ACL inheritance

Use ACL policies to use the same set of permissions to multiple objects in the protected object space. The ACL policy contains enough permissions to meet the requirements of all objects to which the ACL applies. However, each individual object might be affected by only a few of these permissions.

In an ACL inheritance model, any object might not have an explicitly attached ACL policy. The object inherits the policy definitions from the nearest attached ACL policy to an object above it in the hierarchy.

In summary, an ACL policy describes the necessary permissions for all object types to which it can apply, and the object to which it is attached.

ACL policy inheritance example

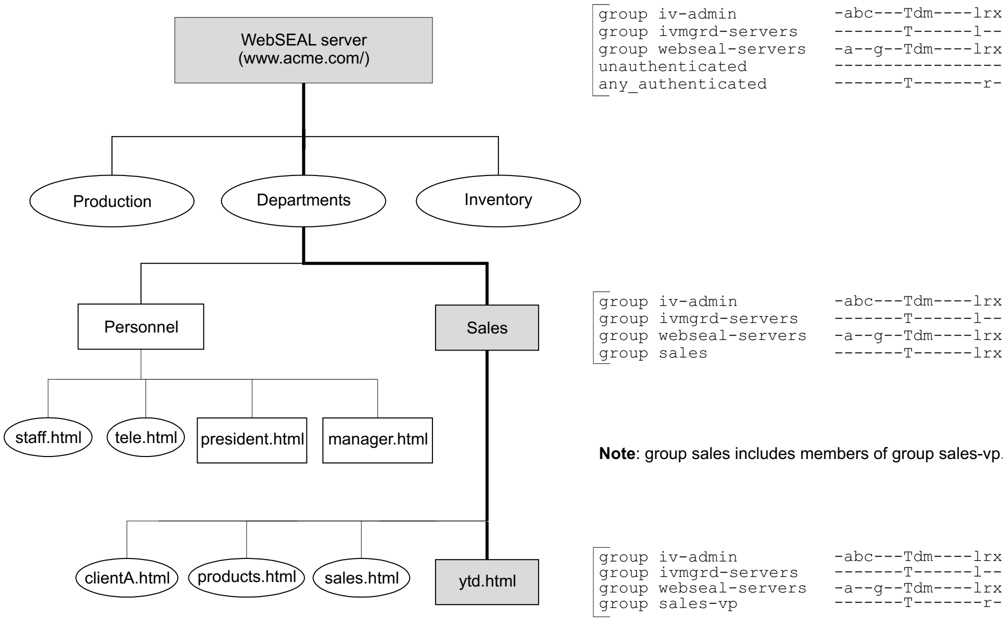
This example illustrates the impact of a mixture of inherited and explicit ACL policies in the fictional ACME corporate object space.

A corporate object space has a general security policy set at the root object. Root is followed by the `/WebSEAL` container object and individually controlled departmental subtrees.

In this example, the `sales` group is given ownership of its departmental subtree. The ACL policy on this subtree no longer acknowledges the **unauthenticated** or **any-other** entry types.

The `ytd.html` file has an attached ACL policy that grants read permission to members of the `sales-vp` group (who are also members of the `sales` group).

Note: This ACL policy scheme does not need to be changed when users are added to or removed from the domain. Users can be added to or removed from the existing groups.



```
group iv-admin         -abc---Tdm----lrx
group ivmgrd-servers   -------T------l--
group webseal-servers  -a--g--Tdm----lrx
unauthenticated        -----------------
any_authenticated      -------T-------r-
```

```
group iv-admin         -abc---Tdm----lrx
group ivmgrd-servers   -------T------l--
group webseal-servers  -a--g--Tdm----lrx
group sales            -------T------lrx
```

Note: group sales includes members of group sales-vp.

```
group iv-admin         -abc---Tdm----lrx
group ivmgrd-servers   -------T------l--
group webseal-servers  -a--g--Tdm----lrx
group sales-vp         -------T-------r-
```

Figure 15. ACL inheritance example

Default ACL policies

You can add entries for users, groups, **any-other** (**any-authenticated**), and **unauthenticated** to provide a broader range of control. These entries can better meet the requirements of your protected object space.

Users and groups with the control (**c**) permission own the ACL and have the power to modify the ACL entries.

A detailed description of permissions can be found in "Default permissions in the primary action group" on page 76.

The following default ACL policies are suggested starting points for securing management operations in a domain:

default-root ACL policy

The ACL policy for the entire object space is the default-root ACL policy.

This ACL policy includes the following users and permissions:

```
group iv-admin              TcmdbvaBR
any-other                   T
unauthenticated             T
```

The default-root ACL policy is a basic policy that enables everyone to traverse the object space, but they cannot do any other actions. Typically, you would not need to change this setting.

Use the default-root ACL policy to quickly deny access to the entire object space for an individual user or group. Consider the following entry in the default-root ACL policy:

```
user john                    -----------------
```

The user john has no permissions. This user cannot even traverse the root container object. The user cannot access the protected object space regardless of any permissions that are granted lower in the tree.

default-management ACL policy

The default ACL policy of the /Management container object is the default-management ACL policy.

At installation, this ACL policy is attached to the /Management container object in the object space. This ACL policy includes the following users and permissions:

```
group iv-admin               TcmdbsvaBtNWAR
group ivmgrd-servers         Ts
any-other                    Tv
```

default-replica ACL policy

The default ACL policy for the /Management/Replica container object is the default-replica ACL policy.

This ACL policy includes the following users and permissions:

```
group iv-admin               TcbvaBR
group ivmgrd-servers         m
group secmgrd-servers        mdv
group ivacld-servers         mdv
```

default-config ACL policy

The default ACL policy for the /Management/Config container object is the default-config ACL policy.

This ACL policy includes the following users and permissions:

```
group iv-admin               TcmdbsvaBR
any-other                    Tv
unauthenticated              Tv
```

default-gso ACL policy

The default ACL policy for the /Management/GSO container object is the default-gso ACL policy.

This ACL policy includes the following users and permissions:

```
group iv-admin               TcmdbvaBNR
any-other                    Tv
unauthenticated              Tv
```

default-policy ACL policy

The default ACL policy for the /Management/Policy container object is the default-policy ACL policy.

This ACL policy includes the following users and permissions:

```
group iv-admin              TcmdbvaBNR
any-other                   Tv
unauthenticated             Tv
```

default-domain ACL policy

The default ACL policy for the /Management/Domain container object is the default-domain ACL policy.

This ACL policy includes the following users and permissions:

```
group iv-admin       TcmdbvaBNR
group ivmgrd-servers  v
```

default-proxy ACL policy

The default ACL policy for the /Management/Proxy container object is the default-proxy ACL policy.

This ACL policy includes the following users and permissions:

```
group iv-admin       Tcbv
group ivmgrd-servers  Tg
```

/Management permissions

The /Management region of the protected object space contains multiple container objects.

The following security considerations apply for the /Management region of the protected object space:

- The /Management object begins the chain of permission inheritance for the entire /Management region of the object space.
- If you do not apply any other explicit permission, this object defines, through inheritance, the security ACL policy for the entire /Management object space.
- The traverse (T) permission is required for access to /Management.

The /Management region contains the following container objects that each requires a specific set of permissions:

/Management/ACL permissions

Use this object to do high-level ACL management tasks that can affect the security policy for the domain.

Table 3. /Management/ACL permissions

Permission	Operation
d (delete)	Delete an existing ACL policy.
m (modify)	Create an ACL policy.
v (view)	List and find view ACLs; show ACL details. This permission must be in an entry of an ACL attached to the /Management/ACL object.

The **acl find** command shows the list of protected resources where this ACL is attached. You must have the view (**v**) permission on those protected resources before they can be shown.

You must create ACL administrator entries in the effective ACL policy for the /Management/ACL object. The ACL entry of an administrator might contain any of the permissions listed in the table. These permissions give the administrator powers to create, view, and delete ACL policies.

An ACL administrator cannot modify an existing ACL unless there is an entry in that ACL for the administrator that contains the control (**c**) permission. Only the owner of an ACL can modify its entries.

The creator of a new ACL policy (**m** on /Management/ACL) becomes the first entry in that ACL with the TcmdbsvaBIR permissions set by default.

For example, if **sec_master** is an administrator entry in the default-management ACL, with **m** permission, **sec_master** can create an ACL policy. User **sec_master** becomes the first entry in the new ACL, with TcmdbsvaBIR permissions.

Ownership of the default-management ACL itself is given to the **iv-admin** group by default.

/Management/Action permissions

You might need to use the /Management/Action permissions to manage custom actions and action groups.

Action tasks and associated permissions include:

Table 4. /Management/Action permissions

Permission	Operation
d (delete)	Delete an existing action or action group.
m (modify)	Create an action or action group.

To view an action or action group, no special permissions are required.

Resource managers can call the authorization service through the authorization API. To integrate a resource manager with the authorization service:

1. Define the object space for the resource manager.
2. Define the action groups and actions for the resource manager.
3. Apply permissions on resources and objects that need protection.

The administrator of a resource manager object space can use the **pdadmin** utility to define new permissions and actions. Resource managers generally define the actions and action groups that are applicable to the resources that they are protecting.

The administrator must have the **m** and **d** permissions on the Management/Action object to create and delete these new permissions or actions.

/Management/POP permissions

Use this object to manage protected object policies.

All permissions must appear in entries for ACLs on /Management/POP. Action tasks and associated permissions include:

Table 5. /Management/POP permissions

Permission	Operation
d (delete)	Delete a POP.
m (modify)	Create POPs and modify POP attributes.
v (view)	Find and list POPs and show POP details.
B (bypass POP)	Override the POP on an object.

The **pop find** command shows the list of protected resources where this POP is attached. You must have the view (**v**) permission on those protected resources before they can be shown.

/Management/Server permissions

Administrators can do server tasks with the `/Management/Server` container object of the protected object space when the appropriate permissions are set.

Server management controls:

- Determine whether a user has permission to view configured resource managers.
- Initiate® a replication of one or more resource managers.
- Enable runtime tracing features on behalf of resource managers.

Resource managers become available in the list of resource managers after they are configured into the domain. Resource managers are removed when they are unconfigured.

The viewable resource manager information allows other Security Access Manager servers, particularly the policy server, to locate and communicate with that resource manager.

Table 6. /Management/Server permissions

Permission	Operation
s (server)	Replicate the resource manager or the authorization database.
v (view)	List registered servers and display server properties.
t (trace)	Enable dynamic trace or statistics administration.

/Management/Config permissions

Use the `/Management/Config` container object of the protected object space to do configuration tasks when the appropriate permissions are set.

Configuration management controls are used to determine whether a user has permission to configure, unconfigure, or update the configuration of a resource manager.

A server definition is created for a particular resource manager or the authorization server as part of the configuration process. The definition for a server is deleted when the server is unconfigured.

Server definitions contain information that allows other Security Access Manager servers, particularly the policy server, to locate and communicate with that resource manager.

Table 7. /Management/Config permissions

Permission	Operation
m (modify)	Configure a resource manager into a domain or update the configuration of a resource manager.
d (delete)	Unconfigure a resource manager from a domain.

/Management/Policy permissions

Use the `/Management/Policy` container object of the protected object space to authorize the **`policy get`** and **`policy set`** commands when the appropriate permissions are set.

Table 8. /Management/Policy permissions

Permission	Operation
v (view)	Required for **`policy get`** commands.
m (modify)	Required for **`policy set`** commands.

/Management/Replica permissions

Use the `/Management/Replica` container object of the protected object space to control the replication of the master policy database.

High-level controls on this object affect the operation of the policy server and the resource managers in the domain.

Replica management controls are used to determine which resource managers are allowed to download the master policy database to their local file system.

Table 9. /Management/Replica permissions

Permission	Operation
v (view)	Read the master policy database.

All Security Access Manager servers that maintain a local replica of the policy database must be granted view (**v**) permission on the `/Management/Replica` object. This group of servers includes all resource managers and the authorization servers. The replication process requires that these processes be allowed to view and access entries out of the master policy database.

The Security Access Manager installation automatically grants read permission to any server that requires access to the master policy database. When a resource manager is configured into the domain, it is automatically added as a member to the **ivacld-servers** group. This group, by default, is given permission to download the master policy database.

/Management/Users permissions

Use this object to manage user accounts.

Action tasks and associated permissions include:

Table 10. /Management/Users permissions

Permission	Operation
d (delete)	Delete a user account.
m (modify)	Modify the details of a user account.
N (create)	Create a user and optionally assign that user to one or more groups. Import group data from the user registry.
v (view)	List user accounts and show details for a user account.
W (password)	Reset and validate a user password.

The password (**W**) permission allows password resets. This permission is appropriate to give to help desk administrators so that they can assist users who forget their passwords. This permission allows an administrator to reset the password and then to use the **`user modify password-valid`** command to set a value of `no`. This action allows the user to log on and then forces the user to immediately apply a new password. Setting **`user modify password-valid`** to `no` for a user does not indicate whether the password is not valid due to the maximum password age policy, which is a global setting. The **`policy set max-password-age`** command sets the maximum time that must elapse before a password expires.

The ability for an administrator to manage all user accounts is controlled by permissions on the `/Management/Users` object. For example, if an administrator has view (**v**) permission on the `/Management/Users` object, that administrator can view information about all users.

To limit the scope of administrator control to a specific group, remove the administrator permissions from the `/Management/Users` object. Apply permissions to the `/Management/Groups` object that is associated with the group to be managed. For example, if an administrator is given view (**v**) permission on the `/Management/Groups/Accounting` object, that administrator can view only information about users in the `Accounting` group.

If an administrator has view (**v**) permission to any group that the user is a member of, the administrator can view the information for that user. Adding the view (**v**) permission to the `/Management/Groups` object itself allows an administrator to view information about any user who is a member of any group.

Access granted by the `/Management/Users` object overrides any access restrictions imposed by delegated administration policy ACLs under `/Management/Groups/`*`group_name`*. For information about delegated administration, see "[delegated-admin] stanza" on page 215.

/Management/Groups permissions

Use this object to manage groups and group membership.

Table 11. /Management/Groups permissions

Permission	Description
d (delete)	Delete a group.
m (modify)	Modify group descriptions. Remove one or more user members of a group.
N (create)	Create a group. Import group data from the user registry.
v (view)	List groups and show group details.

Table 11. /Management/Groups permissions (continued)

Permission	Description
A (add)	Add one or more users to a group.

The add (**A**) permission is required on your entry in the ACL on a group so that you can add existing users to your group. Use the **`user create`** command, which requires the **N** permission, to create new users and optionally place them in one or more existing groups.

The capability of adding existing users to your group is powerful because the owner of a group has control over all user members of the group. If you, as the owner of the group, also have the delete (**d**) permission, you can delete this user from the entire domain.

The ability for an administrator to manage all groups is controlled by permissions on the `/Management/Groups` object. For example, if an administrator has delete (**d**) permission on the `/Management/Groups` object, that administrator can delete any group.

To limit the scope of administrator control to a specific group, apply permissions to the object that is associated with the group. For example, if an administrator is given delete (**d**) permission on the `/Management/Groups/Travel/Europe` object, that administrator can delete any group within that object.

Permissions on `/Management/Groups` objects affect the ability of an administrator to manage users who are part of those groups. Giving an administrator delete (**d**) permission on a group allows that administrator to delete a user who is a member of the group. If an administrator has view (**v**) permission on a group, that administrator can view information about the users that are part of those groups.

/Management/GSO permissions

Use the `/Management/GSO` container object of the protected object space to do Global Sign-On (GSO) tasks when the appropriate permissions are set.

Table 12. /Management/GSO permissions

Permission	Operation
N (create)	Create a resource, resource group, or resource credential. Creating a resource, resource group, or resource credential also requires the **m** (modify) permission.
d (delete)	Delete a resource, resource group, or resource credential. Deleting a resource, resource group, or resource credential also requires the **m** (modify) permission.
m (modify)	Modify a resource group or resource credential.
v (view)	List or show resources, resource groups, and resource credentials.

/Management/Rule permissions

Use this object to manage authorization rule policies.

All permissions must occur in entries for ACLs on `/Management/Rule`.

Table 13. /Management/Rule permissions

Permission	Operation
R (bypass rule)	Override the authorization rule policy on an object.
d (delete)	Delete an authorization rule.
m (modify)	Create authorization rules and modify authorization rule attributes.
v (view)	Find and list authorization rules and show authorization rule details.

The **authzrule find** command shows the list of protected resources where this rule is attached. You must have the view (**v**) permission on those protected resources before they can be shown.

/Management/Domain permissions

Use the /Management/Domain container object of the protected object space to do domain tasks when the appropriate permissions are set.

Table 14. /Management/Domain permissions

Permission	Operation
m (modify)	Modify or create a domain.
v (view)	List and show domains.
d (delete)	Delete a domain.

/Management/Proxy permissions

Administrators or resource managers can use the /Management/Proxy container object of the protected object space to do delegated management tasks when the appropriate permissions are set.

Table 15. /Management/Proxy permissions

Permission	Operation
g (delegate)	Allows administrators and resource managers to act on the behalf of the specified credential.

Changing the mapping of HTTP request methods

You can change the mapping of HTTP request methods to ACL permission bits by specifying values in the **[http-method-perms]** stanza of the Web Reverse Proxy configuration file.

For example, to define that an HTTP method **POST** maps to the permission bits **A** and **x**, add the following entry to the **[http-method-perms]** stanza:

```
POST = Ax
```

You can also create custom permissions in custom action groups, for example, **[my-action-group]t**. See “Custom permissions in custom action groups” on page 77.

Here is an example **[http-method-perms]** stanza:

```
<default> = r
GET = r
HEAD = T
```

```
PUT = m
POST = Ax
DELETE = d
TRACE = [my-action-group]t
```

Note: The **`<default>`** entry is always mandatory when you add any entries to the **`[http-method-perms]`** stanza.

The **`[http-method-perms]`** stanza can be specified on a per-junction basis by creating a stanza in the form of `[http-method-perms:`*`junction`*`]`. The values in the global **`[http-method-perms]`** stanza apply to any junctions that do not have a junction-specific stanza. A junction-specific stanza **`[http-method-perms:junction]`** does not inherit values from a global stanza. For more information, see [http-method-perms] stanza.

The **`[http-method-perms]`** stanza is empty when the Web Reverse Proxy instance is first configured. If the **`[http-method-perms]`** stanza is empty, the Web Reverse Proxy defaults to the following ACL bits:

```
PUT => m
DELETE => d
All else (GET,POST .. ) => r
```

Configuration validation

The configuration that is specified in the **`[http-method-perms]`** stanza is validated each time that a junction is created. The configuration is also validated when the Web Reverse Proxy instance is restarted. This mechanism ensures that the specified ACL bits correspond to actions already defined in the policy database.

The validation process of the **`[http-method-perms]`** configuration might cause authorization audit events to be generated. Such audit events will appear as unauthenticated accesses to the Security Access Manager policy object**`/Permission-Configuration`**.

If you want to suppress these audit events, attach a protected object policy (POP) with its audit-level attribute set to **`none`** to an object with the name **`/Permission-Configuration`**. This object is not created in the object space by default. The following **`pdadmin`** commands illustrate the creation of policy that will suppress the generation of these audit events:

```
pdadmin> objectspace create /Permission-Configuration
"Permission configuration validation" 0
pdadmin> pop create permission-configuration
pdadmin> pop modify permission-configuration set audit-level none
pdadmin> pop attach /Permission-Configuration permission-configuration
```

Chapter 5. Domain management

An administrator in the management domain can create additional domains. You must specify a unique name and an administrator when you create the domain. Domain administrators can do administrative tasks only within their own domains and do not have the authority to do tasks in other domains.

In a domain, an administrator can create users, groups, and other objects. Users and groups are specific to their domain and cannot access resources in other domains. If users and groups are created outside of Security Access Manager, these users and groups can be imported into other domains. Resources that are defined and access controls for resources that are protected by Security Access Manager are maintained on a per domain basis. Resources and access controls for resources cannot be shared among domains.

Logging on to domains

You can log on to a domain with the Web Portal Manager or **pdadmin** utility.

Logging on to domains with Web Portal Manager

You can log on to domains with Web Portal Manager.

Procedure

1. From the login screen, type the domain name that you created. The default domain name is `Default`.
2. Type the user ID that was created for this domain. The default user ID is **sec_master**.
3. Type the password associated with the user ID.

Logging on to domains with pdadmin

You can log on to domains with the **pdadmin** utility.

Procedure

To log on to a domain, use the **login** command. Specify an administrator user ID and password, and a value for the domain.

Example

For example, the `myadmin_id` administrator can log on interactively to the `Domain-ABC` domain by entering the following command:

```
pdadmin login -a myadmin_id -p 12A345 -d Domain-ABC
```

See the *IBM Security Access Manager for Web: Command Reference*.

Creating a domain

You can create several domains in addition to the management domains with the Web Portal Manager or **pdadmin** utility.

Only an administrator who is logged on to the management domain is authorized to create additional domains. Only an administrator with the appropriate permissions in that management domain can create a domain.

Creating a domain with Web Portal Manager

You can create a domain with the Web Portal Manager.

Procedure

1. Log on to the Web Portal Manager management domain as a domain administrator.
2. Click **Secure Domain → Create Secure Domain**.
3. Type the **Secure Domain Name** that you want to create. For example, type `Domain-ABC`.

   The following restrictions apply to the domain name:
   - The maximum length is limited to 64 characters.
   - The name can contain a-z, A–Z, 0–9, hyphen (-), underscore (_), period (.), "at" symbol (@), or ampersand (&) characters.
   - The name can contain any character from a double-byte character set.
4. Optional: Type a **Description** of the domain, such as: `Test Domain`.
5. Type a **New Domain Administrator ID**. For example, type `myadmin_id`.

   Note: You must create an administrator ID for the domain.
6. Type a **New Administrator Password**. For example, type `12A345`. Passwords must adhere to the password policies set by the domain administrator.
7. Type the password again in **Confirm Password**.
8. Click **Create**.

Creating a domain with pdadmin

You can create a domain with the **pdadmin** utility.

Procedure

1. Log on to the management domain.
2. Use the **domain create** command.

Example

For example, to create a domain named `Domain-ABC`, enter the following command on a single line:

```
pdadmin sec_master> domain create Domain-ABC myadmin_id 12A345 -desc "Test Domain"
```

See the *IBM Security Access Manager for Web: Command Reference*.

Modifying the description for a domain

You can modify a domain description by using the Web Portal Manager or **pdadmin** utility.

Only an administrator who is logged on to the management domain is authorized to modify a domain description. A domain can be modified only by an administrator with the appropriate permissions within the management domain.

Modifying the description of a domain with Web Portal Manager

You can modify the description of a domain with the Web Portal Manager.

Procedure

1. Log on to the Web Portal Manager management domain as a domain administrator.
2. Click **Secure Domain → List Secure Domain**.
3. From the Manage Secure Domains page, click the name of the domain that you want to change. For example, click `Domain-ABC`.
4. From the Secure Domain Properties page, edit the **Description** field to add a description or change the existing description. For example, type `new test domain description` to change the existing description.
5. Click **Apply**.

Modifying the description of a domain with pdadmin

You can modify the description of a domain with the **pdadmin** utility.

Procedure

1. Log on to the management domain as a domain administrator.
2. Use the **domain modify** command.

Example

For example, to change the description of the domain named `Domain-ABC` to `new test domain description`, enter the following command on a single line:

```
pdadmin sec_master> domain modify Domain-ABC description "new test
domain description"
```

See the *IBM Security Access Manager for Web: Command Reference*.

Listing domains

You can list all domains, except for the management domain, by using the Web Portal Manager or **pdadmin** utility.

Only an administrator who is logged on to the management domain is authorized to list domains. The administrator must have the appropriate permissions to list domains within the management domain.

The Manage Secure Domains page displays all the domain names, except for the management domain, as links. You can filter the domain names to view only the domain names that meet the criteria you specify.

Listing domains with Web Portal Manager

You can list the domains, except for the management domain, with the Web Portal Manager.

Procedure

1. Log on to the Web Portal Manager management domain as a domain administrator.
2. Click **Secure Domain → List Secure Domain**.

Listing domains with pdadmin

You can list all domains, except for the management domain, with the **pdadmin** utility.

Procedure

1. Log on to the management domain as a domain administrator.
2. Use the **domain list** command.

   ```
   pdadmin sec_master> domain list
   ```

   See the *IBM Security Access Manager for Web: Command Reference*.

Deleting a domain

You can delete a domain by using the Web Portal Manager or **pdadmin** utility.

Only an administrator who is logged on to the management domain with the appropriate permissions is authorized to delete domains.

Deleting a domain deletes the specified Security Access Manager group. Specifying the optional registry entry option deletes all user and group information, including associated ACL entries, from the user registry when the domain is deleted.

Note: The delete operation cannot be reversed.

Deleting a domain with Web Portal Manager

You can delete a domain with the Web Portal Manager.

Procedure

1. Log on to the Web Portal Manager management domain as a domain administrator.
2. Click **Secure Domain → List Secure Domain**.
3. From the Domain List page, select the domain you want to delete.
4. From the Domain Properties page, click **Delete**.

   To permanently remove domain information from the user registry, click **Delete Registry Entry**. Otherwise, the user and group information for the domain remains in the user registry and can be used if the domain is created again.

Deleting a domain with pdadmin

You can delete a domain with the **pdadmin** utility.

About this task

Note: If you unconfigure the management domain with the **pdconfig** utility, any additional domain that exists is deleted.

Procedure

1. Log on to the management domain as a domain administrator.
2. Use the **domain delete** command.

   To permanently remove domain information from the user registry, use the **–registry** option. Otherwise, the user and group information for the domain remains in the user registry and can be used in case the domain is created again.

Example

For example, to delete the domain named `Domain-ABC` and permanently remove the domain information from the user registry, enter the following command:

```
pdadmin sec_master> domain delete Domain-ABC –registry
```

See the *IBM Security Access Manager for Web: Command Reference*.

Chapter 6. Object space management

Security Access Manager represents resources to be protected with a virtual representation of the object space that is called the *protected object space*.

An object space consists of resource objects and container objects. *Resource objects* are logical representations of resources to be protected. Use *Container objects* to group resource objects and other container objects hierarchically into logical groups or regions. Grouping similar objects makes it easier for you to administer a consistent security policy.

Security policy is applied by attaching access control list (ACL) policies, protected object policies (POPs), and authorization rules to the objects in the object space. These objects represent the physical resources you want to protect. The Security Access Manager authorization service evaluates user credentials and the conditions specified in the security policy. Then, Security Access Manager determines whether to permit or deny access to resources.

The following object spaces are created during the installation of Security Access Manager products:

- The `/Management` object space during the installation of any Security Access Manager product, if it does not exist
- The `/WebSEAL` object space during the installation of Security Access Manager

In the following sections, instructions are provided for using either Web Portal Manager or **pdadmin**, or both. For online help while usingWeb Portal Manager, click the question mark to open a separate help window for the current page.

Note: There are no equivalent **pdadmin** commands for importing, exporting, and copying object spaces.

Creating an object space

You can create an object space with the Web Portal Manager or **pdadmin** utility.

To do this task, the administrator must have the following permissions:

- Create (**N**)
- Modify (**m**)

Creating an object space with Web Portal Manager

You can create an object space with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space → Create Object Space**.
3. Type an **Object Name**. This field is required. For example: `/Test-Space`
4. Type a **Description** for the object space. For example: `New Object Space`
5. Click **Create**. To see the `/Test-Space` object space in the hierarchical structure, browse the object space. See "Listing object spaces" on page 63.

Results

Because an object space consists of resource objects and container objects, you do not have to specify an object type when using Web Portal Manager.

Creating an object space with pdadmin

You can create an object space with the **pdadmin** utility.

Procedure

1. Log on to the domain as a domain administrator.
2. Use the **objectspace create** command.

Results

Note: Do not use the **objectspace** command on object spaces that are created by or developed with Security Access Manager. The following object spaces are created by Security Access Manager:

- /Management
- /WebSEAL
- /OSSEAL

For example, to create the /Test-Space object space that is an application container object, which is object type 14, enter the following command:

```
pdadmin sec_master> objectspace create /Test-Space "New Object Space" 14
```

When creating an object space, an object type must be specified. This object space example assigns an object type of 14, which is for an application container object.

"Protected object space" on page 29 describes the two general types of objects: resource objects and container objects. You can select any of the listed object space types. Alternatively, use any unused category number listed in the following list to designate the object space type and assign a meaning to it.

The following object space types are valid for Security Access Manager:

0	Unknown
1	Secure domain
2	File
3	Executable program
4	Directory
5	Junction
6	WebSEAL server
7	Unused
8	Unused
9	HTTP server
10	Nonexistent object
11	Container object
12	Leaf object
13	Port
14	Application container object
15	Application leaf object
16	Management object
17	Unused

See the *IBM Security Access Manager for Web: Command Reference*.

Listing object spaces

You can list all object spaces with the Web Portal Manager or **pdadmin** utility.

To do this task, the administrator requires the following permissions:
- Browse (**b**)
- View (**v**)

Listing object spaces with Web Portal Manager

You can list object spaces with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space → Browse Object Space** to display the Browse Object Space page.

Results

The Browse Object Space page displays all the objects in the domain in a hierarchical structure. All object spaces appear at the same structural level as the default `/Management` object space. Each object space and the corresponding object are displayed as a link. When you select any link, the Protected Object Properties page for that object or object space is displayed.

Listing object spaces with pdadmin

You can list object spaces with the **pdadmin** utility.

Procedure

1. Log on to the domain as a domain administrator.
2. Use the **objectspace list** command.

   ```
   pdadmin sec_master> objectspace list
   ```

   See the *IBM Security Access Manager for Web: Command Reference*.

Copying an object space with Web Portal Manager

You can copy an object space only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space → Copy/Paste Object Space** to display the Copy/Paste Object Space page.
3. To select which objects to copy, navigate the object space and select the object-specific check boxes in the **Copy** column.
4. To select where these objects are to be pasted, navigate to the object space and select the object-specific check boxes in the **Paste** column.
5. Click **Copy/Paste** to copy the selected object space hierarchies to the designated locations.

Results

If successful, the copied object space is shown under the pasted location. To validate, click **Refresh**.

Importing object spaces with Web Portal Manager

You can import object spaces only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space** → **Import Object**.
3. From the Import Protected Object From File page, complete one of the following steps:
   - In the **Object File Name** field, type the name of the object to import. For example, type `objectImport.xml`.
   - Click **Browse** to select a file name.
4. Optional: Select the **Create Groups** check box to trigger the creation of a group for associated ACLs with entries with the type `Group`.
5. When the **Create Groups** box is selected, in the **Registry Container** text field, type the name of the registry container. For example, type `o=ibm,c=us`.
6. The file that contains the object space might be encrypted when it was exported. In the **Encryption String** text field, type the string that was used to encrypt the XML file.
7. Click **Import**.

Results

If successful, the imported object space is available when you browse the object space.

Exporting object spaces with Web Portal Manager

You can export object spaces only with Web Portal Manager

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space** > **Browse Object Space** to display the Browse Object Space page.
3. Navigate the hierarchy and select the object that you want to export.
4. From the Protected Object Properties page, click **Export** to display the Export Object to File page.
5. Optional: Select the **Export Object including Children** check box to descend the object hierarchy and export all child objects.
6. Optional: In the **Encryption String** text field, type the string to use to encrypt the XML file. If not specified, the exported file is in plain text.
7. When an Encryption String is provided, in the **Confirm Encryption String** text field, type the string again.
8. Click **Export** to display the File Download window.
9. Click **Save** to display the Save As window.
10. Click **Save** to create the file that contains the exported description. The default file name is `objectExport.xml`.

Results

If successful, the exported XML description file is available in the specified location.

Deleting an object space

You can delete an object space with the Web Portal Manager or **pdadmin** utility.

To do this task, the administrator requires the following permissions:
- Delete (**d**)
- Modify (**m**)

Deleting an object space with Web Portal Manager

You can delete an object space with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space** → **Browse Object Space**.
3. From the Browse Object Space page, expand and click the object space that you want to delete.
4. From the Protected Object Properties page, the name of the object space is displayed in the **Object Name** field. Click **Delete**.
5. To confirm the deletion, click **Delete** again.

Results

If successful, a message indicates that the object space was deleted.

Deleting an object space with pdadmin

You can delete an object space with the **pdadmin** utility.

Procedure

1. Log on to the domain as a domain administrator.
2. Use the **objectspace delete** command.

Example

For example, to delete the object space named /Test-Space, enter the following command:

```
pdadmin sec_master> objectspace delete /Test-Space
```

See the *IBM Security Access Manager for Web: Command Reference*.

Chapter 7. Manage protected objects

An *object* is a logical representation of a system resource. To protect objects, you must apply security policies. Security policies are the combination of access control list (ACL) policies, protected object policies (POPs), and authorization rules that you can attach to an object.

In the following sections, instructions are provided for using either Web Portal Manager or **pdadmin**, or both. For online help while usingWeb Portal Manager, click the question mark to open a separate help window for the current page.

Note: There are no equivalent **pdadmin** commands for importing, exporting, and copying object spaces.

After an object space is created, you can populate it with objects and then manage these objects. For information about creating an object space, see "Creating an object space" on page 61.

Create an object

You can create an object with the **pdadmin** utility.

Web Portal Manager provides two ways of creating objects:

- Specifying the fully qualified path of the new object, starting from root
- Specifying the new object from the provided path of the parent object

To do this task, the administrator requires the following permissions:

- Create (**N**)
- Modify (**m**)

Creating an object with Web Portal Manager, from root

You can create an object at the root level with Web Portal Manager. The object specifies the path from root.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space** > **Create Object** to display the Create Protected Object page.
3. Type the full path of the object in the **Object Name** text field. For example, type `/Management/Groups/test-object`.
4. Optional: In the **Description** text field, type a description for the object. For example, type `Test Object`.
5. Click **Create**.

What to do next

To be able to attach a policy to this protected object, click **Object Space** → **Browse Object Space**. The Browse Object Space page provides a hierarchical view of all the objects in the domain as links. Click the link for an object to go to its Protected Object Properties page. From this page, select the **Can Policy be attached to this object** check box and click **Apply**.

Creating an object with Web Portal Manager, from parent object

You can create an object that uses the parent object as the base path with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space → Browse Object Space** to display the Browse Object Space page.
3. Navigate the object hierarchy and select the link of the parent object to display the Protect Object Properties page. For example, select the link that is associated with the `/Management/Groups` object.
4. Click the **Create Child Object** link to display the Create Protected Object page where the **Object Name** and **Description** fields contain the values of the parent object.
5. In the **Object Name** field, append a slash and the name of the new object. For example, append `/test-object` to the provided parent path of `/Management/Groups`.
6. Optional: In the **Description** field, modify the description for the object. For example, type `Test Object`.
7. Click **Create**.

What to do next

After the object is created, a dialog is displayed with the link to this object. To attach a policy to this protected object, click this link to display its Protected Object Properties page. From this page, select the **Can Policy be attached to this object** check box and click **Apply**.

Creating an object with pdadmin

You can create an object in the domain with the **`pdadmin`** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **`object create`** command.

Results

For example, to create the object named `/Management/test-object` that is an application container object (14), enter the following command:

```
pdadmin object create /Management/test-object "Test Object" 14
        ispolicyattachable yes
```

The *type* can be one of the following object type categories:

0	Unknown
1	Secure domain
2	File
3	Executable program
4	Directory
5	Junction
6	WebSEAL server
7	Unused

8	Unused
9	HTTP server
10	Nonexistent object
11	Container object
12	Leaf object
13	Port
14	Application container object
15	Application leaf object
16	Management object
17	Unused

When creating an object, a type must be specified. You can select an appropriate category, or use any number to designate the object type and assign a meaning to it.

If the **ispolicyattachable** option is omitted from the **`object create`** command, this command assumes that you intended to use the **`objectspace create`** command. An object space is created rather than an object.

See the *IBM Security Access Manager for Web: Command Reference*.

List objects

You can list objects in the domain with the Web Portal Manager or **`pdadmin`** utility.

To do this task, the administrator requires the following permissions:
- Browse (**b**)
- View (**v**)

Listing objects with Web Portal Manager

You can list objects with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space** > **Browse Object Space**.

Results

The Browse Object Space page displays all the objects in the domain in a hierarchical structure. All object spaces are listed at the same structural level as the default `/Management` object space. Each object space and each object are displayed as a link. When you select any link, the Protected Object Properties page for that object or object space is displayed.

Listing objects with pdadmin

You can list all objects in the domain with the **`pdadmin`** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **`object list`** command.

Example

For example, to list the objects under the /Management object space, enter the following command:

```
pdadmin sec_master> object list /Management
```

See the *IBM Security Access Manager for Web: Command Reference*.

Importing objects with Web Portal Manager

You can import objects only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space** > **Import Object**.
3. From the Import Protected Object From File page, complete one of the following steps:
   - In the **Object File Name** field, type the name of the object to import. For example, type objectImport.xml.
   - Click **Browse** to select a file name.
4. Optional: Select the **Create Groups** check box to trigger the creation of a group for associated ACLs with the type Group.
5. When the **Create Groups** box is selected, in the **Registry Container** text field, type the name of the registry container. For example, type o=ibm,c=us.
6. The file that contains the object space might be encrypted when it was exported. In the **Encryption String** text field, type the string that was used to encrypt the XML file.
7. Click **Import**.

Results

If successful, the imported object is available when you browse the object space.

Exporting objects with Web Portal Manager

You can export an object only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space → Browse Object Space** to display the Browse Object Space page.
3. Navigate the hierarchy and select the object that you want to export.
4. From the Protected Object Properties page, click **Export** to display the Export Object to File page.
5. Optional: Select the **Export Object including Children** check box to descend the object hierarchy and export all child objects.
6. Optional: In the **Encryption String** text field, type the string to use to encrypt the XML file. If not specified, the exported file is in plain text.
7. When an encryption string is provided, in the **Confirm Encryption String** text field, type the string again.
8. Click **Export** to display the File Download window.

9. Click **Save** to display the Save As window.
10. Click **Save** to create the file that contains the exported description. The default file name is `objectExport.xml`.

Results

If successful, the exported XML description file is available in the specified location.

Delete an object

You can delete an object with the Web Portal Manager or **pdadmin** utility.

To do this task, the administrator requires the following permissions:
- Delete (**d**)
- Modify (**m**)

Deleting an object with Web Portal Manager

You can delete an object with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space → Browse Object Space**.

   The Browse Object Space page provides a hierarchical display of all objects in the domain as links.
3. Click the link for an object to see its properties. These properties include whether ACL policies, POPs, and authorization rules are attached to the object and whether the object has extended attributes. For example, click the `/Management/text-object` link to display its properties.
4. From the Protected Object Properties page, ensure the object named is the one you want to delete and click **Delete**.

Deleting an object with pdadmin

You can delete an object with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **object delete** command.

Example

For example, to delete the object named `/Management/test-object`, enter the following command:

```
pdadmin object delete /Management/test-object
```

See the *IBM Security Access Manager for Web: Command Reference*.

Chapter 8. Manage access control

A domain administrator can use access control list (ACL) policies to control access to objects.

ACL policies contain ACL entries that control who can access which domain resources and do which actions. A domain administrator manages the ACL policies by adding, removing, and modifying the ACL entries in the ACL policies. See "ACL policies." For details about the ACL policy tasks that a domain administrator can do, see "Manage ACL policies" on page 79.

An *ACL entry* defines a user or group and which actions each can do against a protected object. A domain administrator can manage these ACL entries before or after the ACL policy is attached to domain resources. Any change to the ACL entry affects only the access that these users and groups have against a specific domain resource to which the ACL policy is attached. See "ACL entries" on page 74.

To define ACL entries, a domain administrator adds or removes permissions (actions) for specific users or groups. A *permission* is an action that is defined by an action bit in an action group. An *action group* is a set of permissions. A domain administrator can add or remove action groups from an ACL entry.

When Security Access Manager is installed, the `primary` action group is created, and contains 17 permissions. These permissions are defined with action bits.

As additional resource managers are installed, additional action groups might be created. As needed, a domain administrator can create additional action groups and add new actions to previously created action groups. See "Action groups and actions" on page 76. For details about the action group tasks that a domain administrator can do, see "Manage action groups" on page 93. For details about the action tasks that a domain administrator can do, see "Manage actions" on page 95.

A domain administrator can assign administrative authority to another user. To define another administrative user, the domain administrator sets the ACL entries for that user to match the ACL entries of the domain administrator. In this situation, both the new administrative user and the domain administrator have the same authority.

ACL policies

In the protected object space, ACL policies can be attached to resource objects and container objects.

Each ACL policy contains one or more ACL entries that affect only that object. For example, the ACL policy that is attached to the `spooler` object might allow all requesters the following permissions:

- Execute
- List
- Read
- Write

However, the ACL policy that is attached to the `docs_repository` object might allow all requesters the following permissions:

- List
- Read

In this case, both ACL policies that are attached to these objects for all requesters. However, the permissions that are defined in the ACL entry for all requesters are different.

Container objects represent specific regions in the protected object space. After a domain administrator creates an ACL policy and attaches it to a container object, the ACL policy serves the following important security tasks:

- The root (/) container object begins the chain of ACL inheritance for the entire protected object space.
- Through inheritance, the root object defines the security policy for the entire object space.
- Unless an explicit ACL policy is attached to a contained object, the ACL policy for the container object defines the security policy for all resources in that container object.
- The traverse permission allows a requester to pass through a container object to the requested object. To deny access to all objects in a region, remove the traverse permission (**T** action bit) from the ACL entry.
- The traverse permission does not grant any other access controls to the container object.

ACL entries

Each ACL policy can contain one or more ACL entries. Each ACL entry contains attributes that identify the user or group and the actions that this user or group can perform.

The number of required attributes for an ACL entry depends on the ACL entry type. The general format of an ACL entry contains the following attributes:

Type Specifies the entity category (user, group, or special) for which the ACL entry was created. See “Type attribute” on page 75.

ID The unique identifier (name) of the user or group that is specified with the type attribute. The **any-other** and **unauthenticated** special entry types do not require the ID attribute. See “ID attribute” on page 75.

Permissions
Defines the set of permissions (actions) that are permitted on the resource by this user or group. Permissions are defined by using action bits. Action bits are defined in action groups. See “Permissions attribute” on page 75 and “Action groups and actions” on page 76.

Figure 16 shows the attributes of an ACL entry.

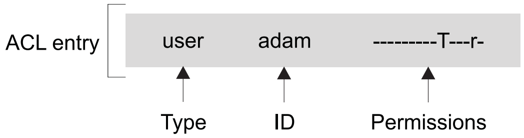


Figure 16. ACL entry attributes

Type attribute

The type attribute of an ACL entry type identifies the user, group, or special entity for a specific ACL entry.

The following types are supported:

user Sets permissions for a specific user in a domain. The user must be a member of the domain with an account in the registry. The user entry type requires a user name (ID). The entry format is `user` *`ID permissions`* as shown in the following example:

```
user anthony -------T-----r-
```

group Sets permissions for all members of a specific group in a domain. The group entry type requires a group name (ID). The entry format is `group` *`ID permissions`* as shown in the following example:

```
group engineering -------T-----r-
```

any-other

Sets permissions for all authenticated users. No ID designation is required. The entry format is `any-other` *`permissions`* as shown in the following example:

```
any-other   -------T-----r-
```

The **any-other** entry type is also known as **any-authenticated**.

unauthenticated

Sets permissions for those users who are not authenticated by the policy server. No ID attribute is required in the ACL entry. The entry format is `unauthenticated` *`permissions`* as shown in the following example:

```
unauthenticated -------T-----r-
```

This ACL entry is a mask (a bit-wise *and* operation) against the **any-other** ACL entry to determine the action set. A permission for **unauthenticated** is granted only if the permission also appears in the **any-other** entry.

For example, when **unauthenticated** has read and write permissions and **any-other** has transverse and read permissions, the resulting action set is read only. This example is shown in the following equation:

```
   unauthenticated -------------rw
+  any-other       -------T-----r-
----------------------------------
                   -------------r-
```

ID attribute

Each user ACL entry and each group ACL entry have unique identifiers (name).

These names must represent valid users or groups that are created in a domain and have an account in the registry.

The **any-other** and **unauthenticated** special entry types do not use the ID attribute.

Permissions attribute

Each ACL entry contains a set of permissions (actions) that describes the specific operations that are permitted on the object by the user or group. Permissions are context-sensitive.

The behavior of certain permissions varies according to where the permissions are applied. For example, the modify permission (**m** action bit) behaves differently for protected resources in the /WebSEAL object space than for protected resources in the /Management object space.

Permissions control protected resources in the following ways:

- Determine whether a user can do operations on protected objects
- Determine whether an administrator can change security policy on the object and any object that inherits permissions
- Determine whether Security Access Manager itself can delegate credentials for a user

Action groups and actions

A domain administrator defines the actions that requesters can perform on objects in the protected object spaces. An *action* is a permission in an action group that is defined in the action group by an action bit.

A domain administrator modifies the ACL entries in an ACL policy before or after the ACL policy is attached to an object. The actions that can be defined in an ACL entry must be previously defined in an action group.

When Security Access Manager is installed, the primary action group is created. The primary action group is an action group that is created during the installation of an application or resource manage. As additional applications and resource managers are installed, additional action groups might be created.

Independent of whether additional action groups are created during subsequent installations, a domain administrator can create additional action groups. A domain administrator can create custom permissions in a primary action group or a custom action group by defining new action bits.

Default permissions in the primary action group

Security Access Manager defines permissions with action bits. When you install Security Access Manager, the default primary action group is created. This action group contains 17 permissions.

Web Portal Manager divides these permissions into the following categories.

- Base
- Generic
- Application

Table 16 shows the action bit in the primaryaction group, a brief description of its associated permission, and its category as shown in Web Portal Manager

Table 16. Action bits and permissions

Action bit	Description of permission	Web Portal Manager category
a	Attach	Base
A	Add	Base
b	Browse	Base
B	Bypass protected object policy (POP)	Base

Table 16. Action bits and permissions (continued)

Action bit	Description of permission	Web Portal Manager category
c	Control	Base
d	Delete	Generic
g	Delegation	Base
l	List directory	Application
m	Modify	Generic
N	Create	Base
R	Bypass rule	Base
r	Read	Application
s	Server administration	Generic
t	Trace	Base
T	Traverse	Base
v	View	Generic
W	Password	Base
x	Execute	Application

Security Access Manager provides the capability to define additional permissions for use by resource managers. See “Manage action groups” on page 93.

Custom permissions in custom action groups

The default permissions in the `primary` action group are available to all applications. If a custom action group uses these default permissions, the associated actions must closely match that of the actual operation that is done by an action in the `primary` action group.

For example, the read permission (action bit **r**) must be used only by an action that requires read-only access to a protected object.

The authorization service does not know or care about the action. A custom action group can reuse an action bit from the `primary` action group to create an action in a custom action group for an unrelated operation. However, this situation might cause difficulty for a domain administrator who must be able to distinguish between two dissimilar uses of the same action bit.

A custom action group might use an action that is not appropriately represented by a default permission. A domain administrator can define a new action bit for a permission that can be used and be recognized by the authorization service. See “Manage action groups” on page 93.

When to create custom permissions

This example demonstrates how a domain administrator can protect a printer from unauthorized use by creating a custom action.

Figure 17 on page 78 shows an example of this requirement. A print spooling service is written with the authorization application programming interface (authorization API). The service can call the authorization service to do ACL checks on requests made to the printer.

The default permissions do not include a permission for protecting printers. However, the printer can be protected by a custom action bit (**p** in this example).

An ACL policy is attached to the printer object. If a user requests the use of this protected printer, that user must have an ACL entry that contains the **p** action bit. The authorization service returns a favorable response if the **p** action bit is present and the printing operation proceeds. If the authorization service returns an unfavorable response, the printing operation is not allowed to proceed.

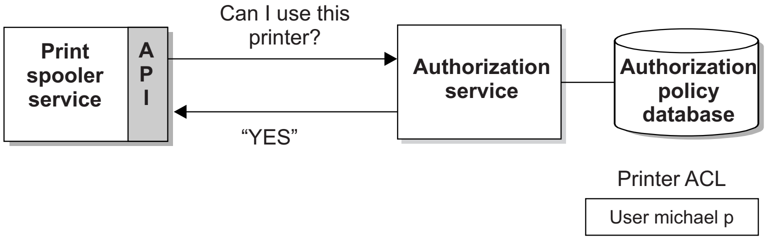


Figure 17. Permissions for a custom print spooler

Representation of custom actions and action groups

You must use a special syntax to identify custom action bits that belong to action groups other than the `primary` action group. The `primary` action group is the default action group.

As described in "ACL entries" on page 74, ACL entries contain an entry type, an ID for user and group types, and the set of permissions (action bits).

Permissions that represent the action bits from multiple action groups are presented in the following format:

```
bits[group_1]bits_1...[group_n]bits_n
```

The following example is an example of the permissions attribute:

```
abgTr[groupA]Pq[groupB]Rsy[groupC]ab
```

The previous permissions attribute has the following interpretation:

- The `primary` action group contains the **a**, **b**, **g**, **T**, and **r** action bits.
- The `groupA` action group contains the **P** and **q** action bits.
- The `groupB` action group contains the **R**, **s**, and **y** action bits.
- The `groupC` action group contains the **a** and **b** action bits.

Action group `groupC` contains action bits that use the same letters for action bits as used in the `primary` action group. The action bits are associated with a specific action group (`groupC`). The **a** and **b** action bits have unique identities and can represent different permissions from those action bits in the `primary` action group.

Scenario with custom actions

The following scenarios show how to add custom actions to an ACL policy that is attached to a protected object.

1. To show action groups, enter the following command:

   ```
   pdadmin sec_master> action group list

           primary
           test-group
   ```

2. To list permissions in the `test-group` action group, enter the following command:

```
pdadmin sec_master> action list test-group

      P  Test-Action   Special
      S  Test-Action2  Special
```

3. To list ACL policies, enter the following command:

```
pdadmin sec_master> acl list

      default-webseal
      default-root
      default-gso
      default-policy
      default-config
      test-acl
      default-replica
      default-management
```

4. To show details about the ACL name `test-acl`, enter the following command:

```
pdadmin sec_master> acl show test-acl

      ACL Name: test-acl
      Description:
      Entries:
          User sec_master Tcmdbva
          Group ivmgrd-servers Tl
          Any-other r
```

5. To add an ACL entry for the user named `Kathy` that contains permissions from the action groups named `primary` and `test-group`, enter the following command:

```
pdadmin sec_master> acl modify test-acl set user kathy brT[test-group]PS
```

6. To validate this addition, enter the following command:

```
pdadmin sec_master> acl show test-acl

      ACL Name: test-acl
      Description:
      Entries:
      User  sec_master  Tcmdbva
      Group  ivmgrd-servers  Tl
      Any-other  r
      User  kathy  Tbr[test-group]PS
```

Manage ACL policies

You can create and configure an ACL policy and attach it to objects in the protected object space. ACL policies are placed in the master policy database on a domain-by-domain basis. The master policy database is controlled by the policy server.

In the following sections, instructions are provided for using either Web Portal Manager or **`pdadmin`**, or both. For online help while usingWeb Portal Manager, click the question mark to open a separate help window for the current page.

Note: There are no equivalent **`pdadmin`** commands for importing, exporting, or cloning ACL policies.

In the following sections, instructions are provided for **`pdadmin`**.

Create an ACL policy

You can create an ACL policy with the Web Portal Manager or **`pdadmin`** utility.

An ACL policy contains an entry with all the defined permissions for the logged in user who created the ACL policy. You must modify this ACL policy. Add ACL entries for additional users and groups that need to manage this ACL policy and the objects to which this ACL policy is attached.

After adding the appropriate ACL entries, you might need to remove the ACL entry for the user who created the ACL policy.

Creating an ACL policy with Web Portal Manager

You can create an ACL policy with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** → **Create ACL**.
3. In the **ACL Name** file, type the name of the ACL policy. For example, type `Test-ACL`.
4. Optional: In the **Description** field, type a description of the ACL. For example, type `Test of new ACL`.
5. Click **Create**.

Results

If successful, a link for this ACL policy is available when you list all ACL policies. You can now add and remove ACL entries from the ACL policy. See "Create an ACL entry" on page 87 and "Remove ACL entries from an ACL policy" on page 89.

Creating an ACL policy with pdadmin

You can create an ACL policy in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **acl create** command.

Results

For example, to create an ACL policy named `Test-ACL`, enter the following command:

```
pdadmin sec_master> acl create Test-ACL
```

For more information, see the *IBM Security Access Manager for Web: Command Reference*.

Modify the description of an ACL policy

You can modify an ACL policy with Web Portal Manager or the **pdadmin** utility.

Modifying the description of an ACL policy with Web Portal Manager

You can modify the description of an ACL policy with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** → **List ACL**.

3. From the Manage ACLs page, click the link for the ACL policy that you want to change.
4. From the ACL Properties page, modify the text in the **Description** field, as appropriate.
5. Click **Set**.

Modifying the description of an ACL policy with pdadmin

You can modify the description of an ACL policy in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as a domain administrator.
2. Use the **acl modify** command with the **description** option.

Example

For example, to modify the description of the ACL named Test-ACL to be ACL for Test resources, enter the following command:

```
pdadmin sec_master> acl modify Test-ACL description "ACL for Test resources"
```

To show the modifications to the ACL, use the **acl show** command. For example, to show the ACL named Test-ACL, enter the following command:

```
pdadmin sec_master> acl show Test-ACL

          ACL Name:  Test-ACL
          Description: ACL for Test resources
          Entries:   User   maryj r
```

See the *IBM Security Access Manager for Web: Command Reference*.

List ACL policies

You can list all ACL policies with Web Portal Manager or the **pdadmin** utility.

Listing ACL policies with Web Portal Manager

You can list ACL policies in the domain with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** → **List ACL**.

Results

The Manage ACLs page displays all the ACL policies in the domains.

Listing ACL policies with pdadmin

You can list ACL policies in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as a domain administrator.
2. Use the **acl list** command.
   ```
   pdadmin sec_master> acl list
   ```
   See the *IBM Security Access Manager for Web: Command Reference*.

View an ACL policy

You can view an ACL policy with Web Portal Manager or the **pdadmin** utility.

Viewing an ACL policy with Web Portal Manager

You can view an ACL policy with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** → **List ACL**.
3. From the Manage ACLs page, click the link for the ACL policy that you want to view.

Viewing an ACL policy with pdadmin

You can view an ACL policy in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as a domain administrator.
2. Use the **acl show** command.

   ```
   pdadmin sec_master> acl show test-acl
   ```

   See the *IBM Security Access Manager for Web: Command Reference*.

Cloning an ACL policy

You can clone an ACL policy only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** > **List ACL**.
3. From the Manage ACLs page, select the ACL policy you want to clone.
4. From the ACL Properties page, click **Clone**.
5. From the Clone ACL page, type an **ACL Name**. For example, type `Test-ACL`. The default value is the name of the original ACL with the prefix `Clone`.
6. Optional: Type a **Description** of the ACL policy For example, type `Clone of new ACL`. The default value is the description of the original ACL.
7. Click **Clone**.

Results

If successful, a link for the cloned ACL policy is created and a success message is displayed.

Importing ACL policies

You can import an ACL policy in the domain only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** > **Import ACL**.
3. From the Import ACL page, complete one of the following steps:
   - In the **ACL File Name** field, type the name of the ACL to import. For example, type `aclImport.xml`.
   - Click **Browse** to select a file name.

4. Optional: Select the **Create Groups** check box to create a group for ACL entries with the type `Group`.
5. If you selected **Create Groups**, type the name of the registry container for the ACL in the **Registry Container** field. For example, type `o=ibm,c=us`.
6. The file that contains the ACL might be encrypted when it was exported. In the **Encryption String** field, type the string that was used to encrypt the XML file.
7. Click **Import**.

Results

If successful, the imported ACL policy is available when you list all the ACL policies.

Exporting all ACL policies

You can export the definitions of all ACL policies in the domain only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** > **Export All ACLs** to display the Export ACL to File page.
3. Optional: In the **Encryption String** text field, type the string to use to encrypt the XML file. If not specified, the exported file is in plain text.
4. When an encryption string is provided, in the **Confirm Encryption String** text field, type the string again.
5. Click **Export** to display the File Download window.
6. Click **Save** to display the Save As window.
7. Click **Save** to create the file that contains the exported description. The default file name is `aclExport.xml`.

Results

If successful, the exported XML description file is available in the specified location.

Exporting a single ACL policy

You can export the definition of a single ACL policy in the domain only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** > **List ACL**.
3. From the Manage ACLs page, select the ACL that you want to export.
4. From the ACL Properties page, click **Export** to display the Export ACL to File page.
5. Optional: In the **Encryption String** field, type the string to use to encrypt the XML file. If not specified, the exported file is in plain text.
6. When an encryption string is provided, type the string again in the **Confirm Encryption String** field.
7. Click **Export** to display the File Download window.
8. Click **Save** to display the Save As window.

9. Click **Save** to create the file that contains the exported description. The default file name is `aclExport.xml`.

Results

If successful, the exported XML description file is available in the specified location.

Exporting multiple ACL policies

You can export the definition of ACL policies in a domain from a list only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** > **List ACL**.
3. From the Manage ACLs page, select the ACLs that you want to export.
4. Click **Export** to display the Export ACL to File page.
5. Optional: In the **Encryption String** text field, type the string to use to encrypt the XML file. If not specified, the exported file is in plain text.
6. When an encryption string is provided, in the **Confirm Encryption String** text field, type the string again.
7. Click **Export** to display the File Download window.
8. Click **Save** to display the Save As window.
9. Click **Save** to create the file that contains the exported descriptions. The default file name is `aclExport.xml`.

Results

If successful, the exported XML description file is available in the specified location.

Attach an ACL policy to an object

You can attach an ACL to a protected object with Web Portal Manager or the **pdadmin** utility.

To do this task, the administrator requires the attach (**a**) permission.

Attaching an ACL policy to an object with Web Portal Manager

You can attach an ACL policy to a protected object with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** → **List ACL**.
3. From the Manage ACLs page, click the link for the name of the ACL that you want to attach to a protected object.
4. From the ACL Properties page, click the **Attach** tab.
5. Click **Attach**.
6. From the Attach ACL page, type a **Protected Object Path**. For example, type `/Management/test-object`.
7. Click **Attach**.

Results

If successful, the protected object is displayed as a protected object link for the named ACL.

Attaching an ACL policy to an object with pdadmin

You can attach an ACL policy to a protected object in a domain with the **pdadmin** utility.

Procedure

1. Log on to the domain.
2. Use the **acl attach** command.

Results

For example, to attach an ACL named Test-ACL to a protected object named /Management/test-object, enter the following command:

```
pdadmin sec_master> acl attach /Management/test-object Test-ACL
```

See the *IBM Security Access Manager for Web: Command Reference*.

Detach an ACL policy from an object

You can detach an ACL from an object with Web Portal Manager or the **pdadmin** utility.

To do this task, the administrator requires the attach (**a**) permission.

Detaching an ACL policy from an object with Web Portal Manager

You can detach an ACL policy from a protected object in the domain with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** → **List ACL**.
3. From the Manage ACLs page, click the link for the ACL policy to detach.
4. From the ACL Properties page, click the **Attach** tab.
5. If the ACL is attached to protected objects, select one or more check boxes for the protected objects from which you want to detach the ACL.
6. Click **Detach**. You are asked to confirm the detachment.

Detaching an ACL policy from an object with pdadmin

You can detach an ACL policy from a protected object in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain.
2. Use the **acl detach** command.

Example

For example, to detach the ACL from the protected object named /Management/test-object, enter the following command:

```
pdadmin sec_master> acl detach /Management/test-object
```

See the *IBM Security Access Manager for Web: Command Reference*.

Locate where an ACL policy is attached

You can find where an ACL is attached with Web Portal Manager or the **pdadmin** utility.

Locating where an ACL policy is attached with Web Portal Manager

You can locate where an ACL policy is attached with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** > **List ACL**. A list of ACL names is displayed.
3. From the Manage ACLs page, click the link for the name of the ACL.
4. From the ACL Properties page, click the **Attach** tab.

Locating where an ACL policy is attached with pdadmin

You can locate where an ACL policy is attached in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as a domain administrator.
2. Use the **acl find** command.

Example

For example, to find where the ACL named Test-ACL is attached, enter the following command:

```
pdadmin sec_master> acl find Test-ACL
```

See the *IBM Security Access Manager for Web: Command Reference*.

Delete an ACL policy

You can delete an ACL policy with Web Portal Manager or the **pdadmin** utility.

Deleting an ACL policy with Web Portal Manager

You can delete an ACL policy with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** → **List ACL**.
3. From the Manage ACLs page, select one or more check boxes of the ACL policies that you want to delete.
4. Click **Delete**, and then confirm the deletion by clicking **Delete** again on the Delete confirmation page.

Results

If successful, the ACL policy is no longer included in the list of ACL policies in the Manage ACLs page.

Deleting an ACL policy with pdadmin

You can delete an ACL policy in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as a domain administrator.
2. Use the **acl delete** command.

Results

For example, to delete the ACL named Test-ACL, enter the following command:

```
pdadmin sec_master> acl delete Test-ACL
```

See the *IBM Security Access Manager for Web: Command Reference*.

Manage ACL entries in ACL policies

You can manage ACL entries in ACL policies with the Web Portal Manager or **pdadmin** utility.

In the following sections, instructions are provided for using either Web Portal Manager or **pdadmin**, or both. For online help while usingWeb Portal Manager, click the question mark to open a separate help window for the current page.

Create an ACL entry

You can create an ACL entry for an ACL policy with Web Portal Manager or the **pdadmin** utility.

Use this procedure to create the ACL entry for any user, group, or special ACL entry type (**any-other** and **unauthenticated**).

Creating an ACL entry with Web Portal Manager

You can create an ACL entry with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** → **List ACL**.
3. From the Manage ACLs page, click the link for the ACL policy you want to change.
4. From the ACL Properties page, click **Create**.
5. Select the appropriate entry type: **user**, **group**, **any-other**, or **unauthenticated**.
6. For **user** or **group**, specify the name.
7. Select the check box for each permission to enable.
8. Click **Apply**.

Creating an ACL entry with pdadmin

You can create an ACL entry for an ACL policy in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **acl modify** command with the **set** option.

Results

For example, to create the permissions for user `maryj` for the `Test-ACL` ACL policy to have `r` (read) action bit, enter the following command:

```
pdadmin sec_master> acl modify Test-ACL set user maryj r
```

To show the modifications to the ACL, use the **`acl show`** command. For example, to show the ACL named `Test-ACL`, enter the following command:

```
pdadmin sec_master> acl show Test-ACL

          ACL Name:  Test-ACL
          Description:
          Entries:   User   maryj r
```

See the *IBM Security Access Manager for Web: Command Reference*.

Modify permissions for an ACL entry

You can modify permissions for an ACL policy with the Web Portal Manager or **`pdadmin`** utility.

Use this procedure to modify the permissions for any user, group, or special ACL entry type (**any-other** and **unauthenticated**).

Modifying permissions for an ACL entry with Web Portal Manager

You can modify permissions for an ACL entry using Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** > **List ACL**.
3. From the Manage ACLs page, click the link for the ACL policy you want to change.
4. From the ACL Properties page, click the permission link.
5. From the ACL Entry Properties page, select the check box for each permission to enable the permission. Clear the check box for each permission to disable the permission.
6. Click **Apply**.

Modifying permissions for an ACL entry with pdadmin

You can modify permissions for an ACL entry in the domain with the **`pdadmin`** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **`acl modify`** command with the **set** option.

Example

For example, to modify the permissions for user `maryj` for the `Test-ACL` ACL policy to have `r` (read) and `w` (write) action bits, enter the following command:

```
pdadmin sec_master> acl modify Test-ACL set user maryj rw
```

To show the modifications to the ACL, use the **`acl show`** command. For example, to show the ACL named `Test-ACL`, enter the following command:

```
pdadmin sec_master> acl show Test-ACL

          ACL Name:  Test-ACL
          Description:
          Entries:   User   maryj rw
```

See the *IBM Security Access Manager for Web: Command Reference*.

Remove ACL entries from an ACL policy

You can remove ACL entries from an ACL policy with the Web Portal Manager or **pdadmin** utility.

Use this procedure to remove the ACL entry for any user, group, or special ACL entry type (**any-other** and **unauthenticated**).

Removing ACL entries from an ACL policy with Web Portal Manager

You can remove ACL entries from an ACL policy with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** > **List ACL**.
3. From the Manage ACLs page, click the link for the ACL policy that you want to remove.
4. From the ACL Properties page, select the user, group, or special ACL entry type to remove.
5. Click **Delete**.

Removing ACL entries from an ACL policy with pdadmin

You can remove ACL entries from an ACL policy in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **acl modify** command with the **remove** option.

Example

For example, to remove the ACL entry for user maryj from the Test-ACL ACL policy, enter the following command:

```
pdadmin sec_master> acl modify Test-ACL remove user maryj
```

To show the modifications to the ACL, use the **acl show** command. For example, to show the ACL named Test-ACL, enter the following command:

```
pdadmin sec_master> acl show Test-ACL

          ACL Name:  Test-ACL
          Description:
          Entries:
```

See the *IBM Security Access Manager for Web: Command Reference*.

Manage extended attributes in ACL policies

The following sections describe using **pdadmin** and Web Portal Manager.

For online help while usingWeb Portal Manager, click the question mark to open a separate help window for the current page.

Create extended attributes for an ACL policy

You can create an extended attribute for an ACL policy with the Web Portal Manager or **pdadmin** utility.

Creating extended attributes for an ACL policy with Web Portal Manager

You can create extended attributes for an ACL policy with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** > **List ACL**.
3. From the Manage ACLs page, click the link of the ACL policy for which you want to create an extended attribute.
4. From the ACL Properties page, click the **Extended Attribute** tab.
5. Click **Create**.
6. From the Create Extended Attribute page, define the extended attribute:
   a. In the **Attribute Name** field, type the name of the attribute. This field is displayed only when the attribute type is "Generic Attribute".
   b. In the **Attribute Type** field, select the type of attribute.
   c. In the **Attribute Value** field, select the value for the attribute, unless the selected attribute type is "Generic Attribute". When you select the "Generic Attribute" attribute type, type the value for the attribute.
7. Click **Apply**.

Creating extended attributes for an ACL policy with pdadmin

You can create extended attributes for an ACL policy in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **acl modify** command with the **set attribute** option.

Example

For example, to create a generic attribute named Dept_No with a value of 445 and associate it with the ACL named Test-ACL, enter the following command:

```
pdadmin sec_master> acl modify Test-ACL set attribute Dept_No 445
```

See the *IBM Security Access Manager for Web: Command Reference*.

Modifying extended attributes from an ACL policy with pdadmin

You can modify extended attributes from an ACL policy in the domain only with the **pdadmin** utility.

About this task

Web Portal Manager does not support modifying attributes. To use Web Portal Manager, an administrator needs to delete the attribute and then create the attribute again.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **acl modify** command with the **set attribute** option.

Example

For example, to modify a generic attribute named Dept_No and add a value of 445 and associate it with the ACL named Test-ACL, enter the following command:

```
pdadmin sec_master> acl modify Test-ACL set attribute Dept_No 445
```

See the *IBM Security Access Manager for Web: Command Reference*.

List extended attributes of an ACL policy

You can list the extended attributes of an ACL policy with the **pdadmin** utility.

Listing extended attributes of an ACL policy with Web Portal Manager

You can list all of the extended attributes of an ACL policy with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** → **List ACL**.
3. From the Manage ACLs page, click the link for name of the ACL policy that you want to view.
4. From the ACL Properties page, click the **Extended Attribute** tab.

Results

The ACL Properties page displays all the extended attributes for the selected ACL policy.

Listing extended attributes of an ACL policy with pdadmin

You can list extended attributes of an ACL policy with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **acl list** command with the **attribute** option.

Example

For example, to list the extended attributes of the ACL policy named pub_acl_3, enter the following command:

```
pdadmin sec_master> acl list pub_acl_3 attribute
```

See the *IBM Security Access Manager for Web: Command Reference*.

View extended attributes of an ACL policy

You can view the extended attributes of an ACL policy with the Web Portal Manager or **pdadmin** utility.

Viewing extended attributes of an ACL policy with Web Portal Manager

You can view extended attributes of an ACL policy with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL → List ACL**.
3. From the Manage ACLs page, click the link for the ACL policy that you want to view.
4. Click the **Extended Attribute** tab.

Viewing extended attributes of an ACL policy with pdadmin

You can view extended attributes of an ACL policy in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **acl show** command with the **attribute** option.

Example

For example, to show the `myAttribute` attribute of the `test-acl` ACL policy, enter the following command:

```
pdadmin sec_master> acl show test-acl attribute myAttribute
```

See the *IBM Security Access Manager for Web: Command Reference*.

Delete extended attributes from an ACL policy

You can delete an extended attribute for an ACL policy with the Web Portal Manager or **pdadmin** utility.

Deleting extended attributes from an ACL policy with Web Portal Manager

You can delete extended attributes from an ACL policy with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL → List ACL**.
3. From the Manage ACLs page, click the link for the ACL policy from which you want to delete extended attributes.
4. From the ACL Properties page, click the **Extended Attributes** tab.
5. Select the extended attributes.
6. Click **Delete**.

Deleting extended attributes from an ACL policy with pdadmin

You can delete extended attributes from an ACL policy in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **acl modify** command with the **delete attribute** option.

Example

For example, to delete the extended attributed named Dept_No from the ACL named Test-ACL, enter the following command:

```
pdadmin sec_master> acl modify Test-ACL delete attribute Dept_No
```

See the *IBM Security Access Manager for Web: Command Reference*.

Deleting extended attribute values from an ACL policy with pdadmin

You can delete extended attribute values from an ACL policy only with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **acl modify** command with the **delete attribute** *attribute_name attribute_value* options.

Example

For example, to delete the value 445 from the extended attributed named Dept_No from the ACL named Test-ACL, enter the following command:

```
pdadmin sec_master> acl modify Test-ACL delete attribute Dept_No 445
```

Only the attribute value is deleted.

See the *IBM Security Access Manager for Web: Command Reference*.

Manage action groups

Permissions grant access to do a specific operation on resources that are protected by Security Access Manager.

Security Access Manager provides 17 predefined permissions for immediate use. These permissions are stored in the predefined action group named primary.

Each permission is associated with an action bit. These predefined permissions are described in “Default permissions in the primary action group” on page 76.

Security Access Manager can create resource manager-specific permissions. For example, you can define the Enqueue permission to grant access to put messages in a message queue.

Security Access Manager supports a total of 32 action groups, including the primary action group.

When you define an action group, the following guidelines and limitations apply:

- Each action group can hold up to 32 action bits (including the action bits for the 17 predefined permissions).

- An action bit is made up of a letter: a-z, A-Z.
- Each action bit character can be used only one time in an action group.
- You can reuse the same action bit in other action groups.

Create action groups

You can create an action group with the Web Portal Manager or **pdadmin** utility.

Creating action groups with Web Portal Manager

You can create action groups with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** → **Create Action Group**.
3. Type the new **Action Group Name**. For example, type `test-group`.
4. Click **Create**.

Results

If successful, a message is displayed when the action group is created.

Creating action groups with pdadmin

You can create action groups in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **action group create** command.

Example

For example, to create an action group named `test-group`, enter the following command:

```
pdadmin sec_master> action group create test-group
```

The `primary` action group always appears in a group listing and cannot be deleted.

You must have an entry in an ACL on the `/Management/ACL` object with the modify (**m**) action to create action groups and the delete (**d**) permission to delete action groups.

See the *IBM Security Access Manager for Web: Command Reference*.

List action groups

You can list all action groups with the Web Portal Manager or **pdadmin** utility.

Listing action groups with Web Portal Manager

You can list all action groups with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** → **List Action Groups**.

Results

The Manage Action Groups page displays a list of all action groups in the domain.

Listing action groups with pdadmin

You can list all action groups in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **action group list** command.

Example

For example, to list all action groups, enter the following command:

```
pdadmin sec_master> action group list
```

See the *IBM Security Access Manager for Web: Command Reference*.

Delete an action group

You can delete an action group with the Web Portal Manager or **pdadmin** utility.

Deleting an action group with Web Portal Manager

You can delete an action group with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL → List Action Groups**.
3. From the Manage Action Groups page, select one or more check boxes for the action groups that you want to delete.
4. Click **Delete**.
5. Confirm the deletion by clicking **Delete** again on the Delete Action Groups page.

Deleting an action group with pdadmin

You can delete an action group with the **pdadmin** utility.

Procedure

1. Log on to the domain as a domain administrator.
2. Use the **action group delete** command.

Example

For example, to delete the action group named `test-group`, enter the following command:

```
pdadmin sec_master> action group delete test-group
```

See the *IBM Security Access Manager for Web: Command Reference*.

Manage actions

You can manage actions with the Web Portal Manager or **pdadmin** utility.

Create actions in an action group

You can create an action in an action group with the Web Portal Manager or **pdadmin** utility.

Creating actions in an action group with Web Portal Manager

You can create actions in an action group with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** > **List Action Groups**.
3. From the Manage Action Group page, click the link for the action group name in which to create the permission. For example, select the `Test-Group` link.
4. From the Action Group Properties page, click **Create** to display the Create Action page. The **Action Group Name** is automatically completed.
5. Type a single character **Action Name**. For example, type `x`.
6. In the **Action Label** field, type a short description of the permission. For example, type `Execute`.
7. In the **Action Type** field, type a description of the permission, such as the application to which the permission is specific. For example, type `WebSEAL`.
8. Click **Create**.

Results

If successful, a message is displayed when the permission is created.

Creating actions in an action group with pdadmin

You can create actions in an action group with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **action create** command.

Example

For example, to create an `x` action bit in the `Test-Group` action group, enter the following command:

```
pdadmin sec_master> action create x Execute WebSEAL Test-Group
```

See the *IBM Security Access Manager for Web: Command Reference*.

List actions in an action group

You can list all actions in an action group with the Web Portal Manager or **pdadmin** utility.

Listing actions in an action group with Web Portal Manager

You can list actions in an action group by using Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL → List Action Groups**.
3. From the Manage Action Group page, click the link for the action group name.

Listing actions in an action group with pdadmin

You can list the actions in an action group with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **action list** command.

Example

For example, to list the actions in the Test-Group action group, enter the following command:

```
pdadmin sec_master> action list Test-Group
```

See the *IBM Security Access Manager for Web: Command Reference*.

Delete actions from an action group

You can delete an action from an action group with the Web Portal Manager or **pdadmin** utility.

Deleting actions from an action group with Web Portal Manager

You can delete actions from an action group with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL → List Action Groups**.
3. From the Manage Action Group page, click the link for the action group name that contains the permission to be deleted.
4. From the Action Group Properties page, select the permission to delete.
5. Click **Delete**.
6. Confirm the deletion by clicking **Delete** on the Delete Action page.

Deleting actions from an action group with pdadmin

You can delete actions from an action group with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **action delete** command.

Example

For example, to delete the x action bit from the Test-Group action group, enter the following command:

```
pdadmin sec_master> action delete x Test-Group
```

See the *IBM Security Access Manager for Web: Command Reference*.

Chapter 9. Protected object policy management

The access control list (ACL) policies provide the authorization service with information to make a `yes` or `no` answer on a request to access a protected object and do some operation on that object. A *protected object policy (POP)* contains additional conditions on the request. The conditions are passed back to the resource manager along with the `yes` ACL policy decision from the authorization service.

It is the responsibility of Security Access Manager and the resource manager to enforce the POP conditions.

Table 17 lists the available attributes for a POP that are provided by Security Access Manager.

Table 17. POP attributes that Security Access Manager provides

POP attribute	Description
Name	Specifies the name of the policy. This attribute relates to the *pop-name* variable in the **pop** command documentation.
Description	Specifies the descriptive text for the policy. This attribute occurs in the **pop show** command.
Warning mode	Provides administrators a means to test ACLs, POPs, and authorization rules. Warning mode provides a way to test the security policy before it is made active.
Audit level	Specifies the type of auditing: all, none, successful access, denied access, or errors. Audit level informs the authorizations service that extra services are required when permitting access to the object.
Time-of-day Access	Day and time restrictions for successful access to the protected object. Time-of-day places restrictions on the access to the object.
IP endpoint authorization method policy	Specifies authorization requirements for access from members of external networks. The IP endpoint authentication method policy places restrictions on the access to the object.
EAS trigger attributes	Specifies an External Authorization Service (EAS) plug-in that is started to make an authorization decision with the externalized policy logic of the customer.
Quality of Protection	Specifies the degree of data protection: none, integrity, or privacy. Quality of Protection informs the authorizations service that extra services are required when permitting access to the object.

Although Security Access Manager provides these POP attributes, it enforces only the following attributes:
- Name
- Description
- Warning mode
- Audit level
- Time-of-day Access

Each resource manager or plug-in can optionally enforce one or more of the following attributes:
- IP endpoint authorization method policy
- EAS trigger attributes
- Quality of Protection

For Security Access Manager IP address support:
- You can grant access to a protected resource based on the IP address that is used by the identity. For example, only users from IP address `9.18.n.n` are allowed to access the protected resource.
- You can define that an additional authentication level is required to access this protected resource based on the IP address that is used by the identity. The step-up level authentication is described in "Configure levels for step-up authentication" on page 114 and the *IBM Security Access Manager for Web: WebSEAL Administration Guide*.

Manage protected object policies

You create and configure a protected object policy (POP) and then attach the POP to objects in the protected object space.

In the following sections, instructions are provided for using either Web Portal Manager or **pdadmin**, or both. For online help while usingWeb Portal Manager, click the question mark to open a separate help window for the current page.

Note: There are no equivalent **pdadmin** commands for importing, exporting, or cloning POPs.

Create a POP

You can create a POP with the Web Portal Manager or **pdadmin** utility.

After creating a POP, you can attach it to an object. For information about attaching a POP, see "Attach a POP to an object" on page 106.

Creating a POP with Web Portal Manager

You can create a POP with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **POP** > **Create POP** to display the Create POP page.
3. In the **POP Name** field, type the name for the POP. For example, type `poptest1`.
4. In the **Description** field, type a description of the POP.
5. Select one or more check boxes for the appropriate audit levels. The audit level is the level of auditing that applies when a resource is accessed to which this POP is attached. You can select more than one audit level. The following choices are available:

   Permit

   Audits all of the requests on a protected object that result in successful access.

   Deny Audits all of the requests on a protected object that result in a denial of access.

Error Audits all of the internally generated error messages that result from a denial of access to the protected object.

Admin
Audits not used by Security Access Manager. However, this option can be used by custom applications.

See "Set an audit level" on page 110.

6. Select the **Warn Only On Policy Violation** check box to enable warning mode attributes. A warning mode attribute indicates whether a policy violation that is related to a resource results in denial of access or in an auditable failure. An auditable failure is an access attempt to a resource to which a POP applies, that results in auditing the access, not denying the access.

   See "Set a warning mode" on page 110.
7. Select a type of **Quality of Protection**. The level of protection applies when a resource is accessed to which this POP is attached. The following choices are available:

   None Requires no Quality of Protection.

   Integrity
   Uses some mechanism to ensure that the data is not changed.

   Privacy
   Requires data encryption for Secure Sockets Layer (SSL).

   See "Set a Quality of Protection level" on page 113.
8. Optional: For **Time of Day Access**, specify the days and times of the day that the resource can be accessed.
   - Select the check boxes for the days of the week that the resource can be accessed.
   - Select either **All Day** or **Between hours of** for the access times that the resource can be accessed on the selected days.
   - If you select **Between hours of**, you must also specify the **Start time** and **End time**.
   - If you select **Between hours of**, you must also specify the **Local Time** or **UTC Time** (Coordinated Universal Time).

   See "Set a time-of-day restriction" on page 111.
9. Click **Create** or click **Create Another** if you want to create another POP.

   If successful, a message confirms that the POP was created.
10. If you clicked **Create**, click **Done**. Otherwise, repeat this procedure. Start at step 3 on page 100 to create another POP.

Creating a POP with pdadmin

You can create a POP in the domain with the `pdadmin` utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the `pop create` command.

Results

After creating a POP, you can attach it to an object. See "Attach a POP to an object" on page 106.

Example

For example, to create a POP named poptest1, enter the following command:

```
pdadmin sec_master> pop create poptest1
```

The new POP contains the following default settings:

```
pdadmin sec_master> pop show poptest1
       Protected object policy:  poptest1
       Description:
       Warning:  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
           Any Other Network 0
```

See the *IBM Security Access Manager for Web: Command Reference*.

Modify a POP

You can modify a POP with the Web Portal Manager or **pdadmin** utility.

Modifying a POP with Web Portal Manager

You can modify a POP with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **POP** > **List POP** to display the Manage POPs page.
3. Click the link for the POP. For example, select poptest1 to display the POP Properties page.
4. Click the **General** tab to change the information for the POP, as needed. For example, change the description from Test POP to Test 1 for POP and then click **Apply**.
5. Click the **Attach** tab to change the protected object attachments.
6. Click the **IP Auth** tab to change the IP authentication.
7. Click the **Extended Attributes** tab to change an extended attribute.

Modifying a POP with pdadmin

You can modify a POP with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **pop modify** commands.

Example

For example to enable the warning mode and set the audit level to permit and deny for the poptest1 POP, enter the following commands:

```
pdadmin sec_master> pop modify poptest1 set warning yes
pdadmin sec_master> pop modify poptest1 set audit-level permit,deny
```

To show these modifications, use the **pop show** commands. For example, to show the modifications to the poptest1 POP, enter the following command:

```
pdadmin sec_master> pop show poptest1

       Protected object policy:  poptest1
       Description:  Test 1 for POP
       Warning:  yes
       Audit level:  permit, deny
       Quality of protection:  none
       Time of day access: sun, mon, tue, wed, thu, fri, sat:
           anytime:local
       IP Endpoint Authentication Method Policy
           Any Other Network 0
```

See the *IBM Security Access Manager for Web: Command Reference*.

List POPs

You can list all POPs with the Web Portal Manager or **pdadmin** utility.

Listing POPs with Web Portal Manager

You can list all POPs with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **POP** → **List POP** to display the Manage POPs page.

Results

All the POPs for the domain are listed as links.

Listing POPs with pdadmin

You can list all POPs in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **pop list** command.

Example

For example, to list all POPs, enter the following command:

```
pdadmin sec_master> pop list
```

See the *IBM Security Access Manager for Web: Command Reference*.

View a POP

You can view a POP with the Web Portal Manager or **pdadmin** utility.

Viewing a POP with Web Portal Manager

You can view a POP with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **POP** > **List POP** to display the Manage POPs page.
3. Click the link for the POP. For example, select poptest1 to display the POP Properties page.

4. On the **General** tab, change the information for the POP, as needed. For example, change the description from `Test POP` to `Test 1 for POP`, and then click **Apply**.
5. Click the **Attach** tab to view the protected object attachments.
6. Click the **IP Auth** tab to view the IP authentication.
7. Click the **Extended Attributes** tab to view all extended attributes.

Viewing a POP with pdadmin

You can view a POP with the **`pdadmin`** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **`pop show`** commands.

Example

For example, to show the modifications to the POP name `poptest1`, enter the following command:

```
pdadmin sec_master> pop show poptest1

       Protected object policy:  poptest1
       Description:  Test 1 for POP
       Warning:  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
           Any Other Network 0
```

See the *IBM Security Access Manager for Web: Command Reference*.

Cloning a POP

You can clone a POP only with the Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **POP** > **List POP**.
3. From the Manage POPs page, select the POP you want to clone.
4. From the POP Properties page, click **Clone**.
5. From the Clone POP page, in the **POP Name** text field, type the name of the POP. For example, type `Test-POP`. The default value is the name of the original POP with the prefix `Clone`. This field is required.
6. Optional: In the **Description** text field, type the description of the POP. For example, type `Clone of new POP`. The default value is the description of the original POP.
7. Click **Clone**.

Results

If successful, a link for this cloned POP is created and a success message is displayed.

Importing POPs

You can import a POP in the domain only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **POP** > **Import POP**.
3. From the Import POP page, complete one of the following steps:
   - In the **POP File Name** field, type the name of the POP to import. For example, type `popImport.xml`.
   - Click **Browse** to select a file name.
4. The file that contains the POP might be encrypted when it was exported. In the **Encryption String** text field, type the string that was used to encrypt the XML file.
5. Click **Import**.

Results

If successful, the imported POP is available when you list all the POPs.

Exporting all POPs

You can export all POPs in the domain only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **POP** > **Export All POPs** to display the Export POP to File page.
3. Optional: In the **Encryption String** text field, type the string to use to encrypt the XML file. If not specified, the exported file is in plain text.
4. When an encryption string is provided, in the **Confirm Encryption String** text field, type the string again.
5. Click **Export** to display the File Download window.
6. Click **Save** to display the Save As window.
7. Click **Save** to create the file that contains the exported POP description. The default file name is `popExport.xml`.

Results

If successful, the exported POP description is available in the specified location.

Exporting a single POP

You can export a single POP in the domain only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **POP** > **List POP**.
3. From the Manage POPs page, select the POP that you want to export.
4. From the POP Properties page, click **Export** to display the Export POP to File page.
5. Optional: In the **Encryption String** text field, type the string to use to encrypt the XML file. If not specified, the exported file is in plain text.

6. When an encryption string is provided, in the **Confirm Encryption String** text field, type the string again.
7. Click **Export** to display the File Download window.
8. Click **Save** to display the Save As window.
9. Click **Save** to create the file that contains the exported POP description. The default file name is `popExport.xml`.

Results

If successful, the new XML file is available in the specified location.

Exporting multiple POPs

You can export POPs in the domain from a list only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **POP** > **List POP**.
3. From the Manage POPs page, select the POPs that you want to export.
4. Click **Export** to display the Export POP to File page.
5. Optional: In the **Encryption String** text field, type the string to use to encrypt the XML file. If not specified, the exported file is in plain text.
6. When an encryption string is provided, in the **Confirm Encryption String** text field, type the string again.
7. Click **Export** to display the File Download window.
8. Click **Save** to display the Save As window.
9. Click **Save** to create the file that contains the exported POP descriptions. The default file name is `popExport.xml`.

Results

If successful, the new XML file is available in the specified location.

Attach a POP to an object

You can attach a POP to an object with the Web Portal Manager or **`pdadmin`** utility.

To do this task, the administrator requires the attach (**a**) permission.

Note: When you attach a POP to an object that protects a case-insensitive junction, you must use all lower-case syntax.

Attaching a POP to an object with Web Portal Manager

You can attach a POP to an object with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **POP → List POP** to display the Manage POPs page.
3. Click the link for the POP.
4. From the POP Properties page, click the **Attach** tab.
5. Click **Attach** to display the Attach POP page.

6. Type the **Protected Object Path** for the protected object to which to attach the POP. Express® the path as the full path name. For example, type `/WebSEAL/serverA/index.html`.

   Note: When you attach a POP to an object that protects a case-insensitive junction, you must use all lower-case syntax.
7. Click **Attach**.

Results

If successful, the protected object is added to the list at the POP Properties–Attach page.

Attaching a POP to an object with pdadmin

You can attach a POP to protected object in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **pop attach** command.

   Note: When you attach a POP to an object that protects a case-insensitive junction, you must use all lower-case syntax.

Example

For example, to attach a POP named `poptest1` to a protected object named `/WebSEAL/serverA/index.html` enter the following command:

```
pdadmin sec_master> pop attach /WebSEAL/serverA/index.html poptest1
```

See the *IBM Security Access Manager for Web: Command Reference*.

Detach a POP from an object

You can detach a POP from a protected object with the Web Portal Manager or **pdadmin** utility.

To do this task, the administrator requires the attach (**a**) permission.

Detaching a POP from an object with Web Portal Manager

You can detach a POP from an object with the Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **Object Space** → **Browse** to display the Browse Object Space page.
3. Click the link for the POP.
4. From the POP Properties page, click the **Attach** tab.
5. Select one or more check boxes for the protected objects from which you want to detach the POP.
6. Click **Detach** to display the Detach POP from Object page, where you are prompted to confirm or cancel the detachment.

Detaching a POP from an object with pdadmin

You can detach a POP from an object with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **pop detach** commands.

Example

For example, to detach the POP from the protected object named /WebSEAL/serverA/index.html, enter the following command:

```
pdadmin sec_master> pop detach /WebSEAL/serverA/index.html
```

See the *IBM Security Access Manager for Web: Command Reference*.

Locate where a POP is attached

You can locate where a POP is attached with the Web Portal Manager or**pdadmin** utility.

Locating where a POP is attached with Web Portal Manager

You can locate where a POP is attached with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **ACL** > **List POP**. A list of POP names is displayed. Each POP name is a link that you can click to display the POP properties page.
3. Click the **Attach** tab.

Locating where a POP is attached with pdadmin

You can locate where a POP is attached in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **pop find** command.

Example

For example, to find where the POP named poptest1 is attached, enter the following command:

```
pdadmin sec_master> pop find poptest1
      /WebSEAL/serverA/index.html
```

See the *IBM Security Access Manager for Web: Command Reference*.

Delete a POP

You can delete a POP with the Web Portal Manager or **pdadmin** utility.

Deleting a POP with Web Portal Manager

You can delete a POP with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **POP → List POP** to display the Manage POPs page.
3. Select one or more check boxes for the POPs that you want to delete.
4. Click **Delete** to display the Delete Pop page.

5. Click **Delete** to confirm the deletion.

Deleting a POP with pdadmin

You can delete a POP with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **pop delete** command.

Results

For example, to delete the POP named poptest2, enter the following command:

```
pdadmin sec_master> pop delete poptest2
```

See the *IBM Security Access Manager for Web: Command Reference*.

Network-based authorization algorithm

The authorization server uses an algorithm to process the conditions in a POP.

1. Check ACL permissions.

   Note: The ACL policy bypass (**B**) permission overrides POP authorization conditions on an object. This permission must be used only by a high-level administrator who needs full access to the protected object space.
2. Verify whether a rule is attached to the object, then verify that all the access decision information (ADI) is present for the coming rule evaluation. If it is not, then find it by querying one of the available sources.
3. Check the IP endpoint authentication method policy on the POP.
4. Check the time-of-day policy on the POP.
5. Check the audit level policy on the POP.
6. Check the authorization rule policy if a rule is attached to the object.
7. If an external authorization service (EAS) operation or POP trigger applies to this access decision, then start the EAS that applies.

Network-based authorization policy

Use the network-based authorization policy to control access to objects based on the IP address of the user.

When an environment contains both IP version 4 (IPv4) and IP version 6 (IPv6) address formats, be aware of the following restrictions:

- For administration commands (for example, **pop modify set ipauth**), IPv4 clients must provide addresses in IPv4 format even with IPv6 servers.
- For C APIs, IPv4 clients *must* provide addresses in IPv4 format even with IPv6 servers.
- For C APIs, IPv6 clients *can* provide addresses in IPv4 or IPv6 format to IPv6 servers.
- For Java methods, both IPv4 and IPv6 clients *must* provide addresses in IPv4 format to IPv4 servers.
- For Java methods, IPv4 clients *can* provide addresses in IPv4 or IPv6 format to IPv6 servers.

For an IPv6 address to be accepted (commands, C APIs, and Java methods), the server *must* be IPv6. You cannot provide an IPv6 address to an IPv4 server.

The network-based authorization policy is set in the IP endpoint authentication method attribute of a POP. You can use this functionality to prevent specific IP addresses or IP address ranges from accessing any resources in your domain. When setting an authorization policy, you can apply requisite step-up configuration.

When you define a network-based authentication policy, specify these parts of the attribute:

- Step-up authentication
- Allowed networks

You can also apply step-up authentication configuration to this policy and require a specific authentication method for each specified IP address range. See "Step-up authentication" on page 114.

Note: The IP address used by the resource manager for enforcing the network-based authorization policy must be the IP address of the originator of the connection.

In this case, the resource manager cannot definitively identify the true IP address of the client. When setting a network-based authorization policy that depends on specific client IP addresses, ensure that those network clients are connecting directly to the resource manager.

Configure POP attributes

POP attributes impose access conditions on an object based on the time of the access. They also indicate whether the access request must be audited.

Set a warning mode

The **`pop modify set warning`** command defines the warning mode attribute. A security administrator uses this command to debug or troubleshoot the accuracy of the authorization policy set on the protected object space.

When you set the warning mode 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 security policy that is attached to the object is set to deny this access.

Audit records are generated that capture the results of all security policies with warning mode set throughout the object space. The audit log shows the outcome of an authorization decision as if the warning attribute was set to `no`. Therefore, the administrator can determine if the policy is set and enforced correctly.

For example:

```
pdadmin sec_master> pop modify poptest1 set warning yes
```

See *IBM Security Access Manager for Web: Command Reference*.

Set an audit level

The **`pop modify set audit-level`** command specifies the granularity level of auditing for a POP.

For example, auditing might be set to record unsuccessful events. You can use the results to detect an unusual number of failed access attempts on a particular resource.

Auditing records are written in a standard Extensible Markup Language (XML) format that allows easy parsing to extract whatever information is required. For example:

```
pdadmin sec_master> pop modify pop_name set audit-level permit,deny
```

Table 18. Audit levels

Value	Description
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 that result from a denial of access to the protected object.

You can apply any combination of these values or specify either `all` to audit all requests or `none` to audit no requests. When enabling granular auditing, specify one or more of the following values:

- `permit`
- `deny`
- `error`

When you specify multiple granular values, use a comma as a separator character between these values.

See *IBM Security Access Manager for Web: Command Reference*.

Set a time-of-day restriction

Use the **pop modify set tod-access** command defines the time-of-day (TOD) attribute. Use this command to place specific day and time conditions on the access to a protected object.

This type of condition might be useful to limit access to information that regularly requires periods of inactivity for modification and updates.

```
pdadmin sec_master> pop modify pop_name set tod-access
time_of_day_string
```

The time-of-day-string argument includes a day-range and a time-range and uses the following format:

```
{anyday|weekday|day_list}:
{anytime|time_spec-time_spec}
[:{utc|local}]
```

The *day_list* variable can be any combination of the following values:

```
mon, tue, wed, thu, fri, sat, sun
```

The *time_spec* range variable must be expressed (in 24 hour time) in the following format:

```
hhmm-hhmm
```

For example, you can specify the time range with the following string:

```
0700-1945
```

The optional time zone `[:{utc|local}]` for the server (not the client) is `local` by default.

For example to change the time-of-day attribute to Monday, Tuesday, and Friday from 1:15 p.m. to 5:30 p.m. local time for the POP named `poptest1`, enter the following command:

```
pdadmin sec_master> pop modify poptest1 set tod-access mon,tue,fri:1315-1730
```

Note: When modifying a protected object 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 greater than the specified end time, then the access is allowed until the specified end time of the next day.

See *IBM Security Access Manager for Web: Command Reference*.

Specify IP addresses and ranges

The **`pop modify set ipauth`** command specifies a network or network range and the required authentication level in the POP.

The network (or network range) can be an IP version 4 (IPv4) or an IP version 6 (IPv6) address.

Note: When adding addresses to a POP, IPv4 addresses must be specified in IPv4 format, due to limitations in the operating system functions provided to Security Access Manager.

All POPs have an **anyothernw** (any other network) IP entry whose default authentication level is `0`. The **anyothernw** entry applies to all networks not specified in the POP. Authentication level `0` adds no additional requirement for authentication. The **anyothernw** authentication level can be modified to a non-zero number or to `forbidden`.

The **anyothernw** entry appears in a POP as `Any Other Network` in the output of the **`pop show`** command:

```
pdadmin sec_master> pop show poptest1

       Protected object policy: poptest1
       Description: Test POP
       Warning: 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
           Any Other Network 0
```

You might need more information about setting the IP authentication mechanism with the **`pop modify`** command. See the *IBM Security Access Manager for Web: Command Reference*.

Adding IP entries

The **`pdadmin pop modify set ipauth add`** command specifies the network (or network range). The command also specifies the required authentication level in the IP endpoint authentication method attribute. You might need to add IP entries to a POP.

Procedure

Specify network (or network range) with an authentication level as a number or as `forbidden`.
Specify an authentication level of `0` to allow authentication. A `forbidden` authentication level indicates that authentication is denied. Specify an authentication greater than `0` to step-up a user to an authentication level. The enforcement of step-up authentication is the responsibility of resource managers. See "Step-up authentication" on page 114.

Note: When adding addresses to a POP, IPv4 addresses must be specified in IPv4 format, due to limitations in the operating system functions provided to Security Access Manager.

Example

The following example adds an IP entry for identities from IPv4 addresses that begin with `9.`

```
pdadmin sec_master> pop modify poptest1 set ipauth add 9.0.0.0 255.0.0.0 5
```

The following example adds an entry for an IPv6 network range:

```
pdadmin sec_master> pop modify poptest1 set ipauth add \
  fedc:ba98:7654:3210:fedc:ba98:7654:3210 ffff:ffff:ffff:ffff:ffff:ffff:ffff:0 6
```

The following example prevents all users (except users specified in the examples) from accessing the object:

```
pdadmin sec_master> pop modify poptest1 set ipauth anyothernw forbidden
```

See the *IBM Security Access Manager for Web: Command Reference*.

Deleting IP entries

The **`pdadmin pop modify set ipauth remove`** command disables authorization requirements for IP addresses that were previously added to a POP.

Procedure

1. Use the **`pdadmin pop modify set ipauth remove`** command for an existing POP. The following example deletes an IPv4 entry from the `poptest1` POP:

   ```
   pdadmin sec_master> pop modify poptest1 set ipauth remove 9.0.0.0 255.0.0.0
   ```

   Only network entries that were previously added can be removed.
2. Repeat the step for additional entries that you want to remove.

What to do next

See the *IBM Security Access Manager for Web: Command Reference*.

Set a Quality of Protection level

The Quality of Protection POP attribute specifies what level of data protection is required when doing an operation on an object.

The Quality of Protection POP attribute permits a single transaction where the `yes` response to the ACL decision also includes the required Quality of Protection level. If the resource manager cannot guarantee the required level of protection, the request is denied.

Use the following **pop modify** command syntax to modify the QoP level for an object:

```
pdadmin sec_master> pop modify pop-name set qop {none|integrity|privacy}
```

Table 19. Quality of Protection levels

QoP level	Description
none	Requires no Quality of Protection.
privacy	Data encryption is required for Secure Sockets Layer (SSL).
integrity	Use some mechanism to ensure that the data is not changed.

For example, to modify the POP named poptest1 to set the Quality of Protection level to use SSL data encryption, enter the following command:

```
pdadmin sec_master> pop modify poptest1 set qop privacy
```

Step-up authentication

You can use protected object policies (POPs) to enforce certain access conditions on specific resources. The authentication strength policy makes it possible to control access to objects based on authentication method.

You can use this functionality, sometimes known as step-up authentication, to ensure that users who access more sensitive resources use a stronger authentication mechanism. You might want this condition because of the greater threat of improper access to certain resources.

For example, you can provide greater security to a junctioned region of the protected object space. Apply a step-up POP policy that requires a stronger level of authentication than the client used when initially entering the domain.

The authentication strength policy is set in the IP endpoint authentication method attribute of a POP policy.

Configure levels for step-up authentication

The first step in configuring authentication-specific access is to configure the supported authentication methods and determine the order in which these authentication methods must be considered stronger.

Any client that accesses a resource manager has an authentication level, such as “unauthenticated” or “password”. The level indicates the method with which the client was last authenticated by the resource manager.

In some situations, it might be necessary to enforce minimum safe levels of authentication required to access certain resources. For example, in one environment, authentication by token pass code might be considered more secure than authentication by user name and password. Another environment might require different standards.

The step-up authentication mechanism does not force clients to restart their sessions with the resource manager when they do not meet the required level of authentication. Instead, the step-up authentication mechanism provides clients a second chance to authenticate with the required method of authentication (level).

Step-up authentication allows resource managers to control how users access protected resources. If step-up authentication is required because the user has not authenticated with the sufficient method, the authorization engine still permits the access decision. However, the resource manager is presented with a required authentication level as an output of the authorization decision. The resource manager can then decide how to further authenticate the user to gain the required level of authentication to access the protected object.

How a particular authentication method is mapped to an authentication level is determined by the resource manager application. For all cases, the absolute minimum acceptable method of authentication must be set as level 0. More secure methods are mapped to integer numbers in ascending order (1..x) from that point forward.

Apply step-up authentication policy

Step-up authentication is implemented through a POP policy placed on the objects requiring authentication-sensitive authorization. You can use the IP endpoint authentication method attribute of a POP policy.

The **pop modify set ipauth** command specifies both the allowed networks and the required authentication level in the IP endpoint authentication method attribute.

Note: When specifying an IPv4 address, it must be in IPv4 format.

The configured authentication levels can be linked to IP address ranges. This method is intended to provide management flexibility. If filtering users by IP address is not important, you can set a single entry for **anyothernw** (any other network). This setting affects all accessing users, regardless of IP address, and requires the users to authenticate at the specified level. This method is the most common method for implementing step-up authentication.

The **anyothernw** entry is used as a network range that matches any network not otherwise specified in the POP. Use this method to create a default entry that can either deny all unmatched IP addresses or allow anyone access who meets the authentication level requirement.

By default, **anyothernw** occurs in a POP with an authentication level index of 0. The entry occurs as `Any Other Network` in the output of the **pop show** command. The following output shows a sample for the `poptest1` POP:

```
pdadmin sec_master> pop show poptest1

      Protected object policy: poptest1
      Description: Test POP
      Warning: 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
          Any Other Network 0
```

See the *IBM Security Access Manager for Web: Command Reference*.

Distinguish step-up from multi-factor authentication

Security Access Manager step-up authentication and multi-factor authentication are two different mechanisms for controlling access to resources.

Multi-factor authentication forces a user to authenticate with two or more levels of authentication. For example, the access control on a protected resource can require the user to authenticate with both user name and password (level 1). The access control can also require the user to authenticate with user name and token passcode (level 2).

Security Access Manager step-up authentication relies on a pre-configured hierarchy of authentication levels and enforces a specific level of authentication according to the policy set on a resource. Step-up authentication does not force the user to authenticate with multiple levels of authentication to access any specified resource. Instead, step-up authentication requires the user to authenticate at a level at least as high as the level required by the policy that protects the resource.

The following example shows the series of commands that are needed to define step-up authentication:

```
pdadmin > pop create test1
pdadmin > pop modify test1 set ipauth anyothernw 1
pdadmin > pop attach /WebSEAL/hostA/junction test1

pdadmin > pop create test2
pdadmin > pop modify test2 set ipauth anyothernw 2
pdadmin > pop attach /WebSEAL/hostA/junction/applicationA test2
```

In the previous example, the `/WebSEAL/hostA/junction` object is protected by a POP requiring authentication level 1. The `/WebSEAL/hostA/junction/applicationA` object is protected by a POP requiring authentication level 2.

Under step-up authentication, user name/password (level 1) authentication is required to access `/WebSEAL/hostA/junction`.

However, user name/token passcode (level 2) authentication is required to access `/WebSEAL/hostA/junction/applicationA`. If the user is currently logged in with a user name and password, a prompt appears requesting user name and token passcode information (the step-up). However, if the user initially logged on to WebSEAL with a user name and a token passcode, access to `applicationA` is immediate, assuming a successful ACL check.

Multi-factor authentication requires both level 1 and level 2 authentication for access to `applicationA`.

Chapter 10. Authorization rules management

These topics provide information about Security Access Manager authorization rules. Authorization rules are conditions contained in an authorization policy that are used to make access decisions based on attributes such as user, application, and environment context.

Authorization rules overview

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 that are based on data supplied to the authorization engine within the user credential. Data might be supplied from the resource manager application or from the encompassing business environment. The language of an authorization rule allows customers to work with complex, structured data by examining the values in that data and making informed access decisions. This information can be defined statically within the system or can be defined during a business process. Rules can also be used to implement extensible, attribute-based, authorization policy by using attributes within the business environment or attributes from trusted external sources.

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

Access decision information

The data and attributes in rule conditions collectively are called *access decision information* (ADI). Authorization API attributes, which are name and value pairs, form the basis of all ADI that can be referenced in a rule or presented to the authorization engine.

Sources for retrieving ADI

The authorization engine can gather ADI from several sources.

User credential entitlements

You can insert additional entitlements data as attribute name-value pairs into the client credential by a Security Access Manager authorization client. Insertion can occur during the user authentication process or at any time during the process of the transaction.

For example, Security Access Manager can be configured to gather entitlements at the time that a user is authenticated. You can configure entitlement services to run during credential acquisition, collect entitlements data, and then append the data to the credential.

Security Access Manager provides a credential attributes entitlement service that retrieves entitlements data from the user registry. Or, you can define your own entitlement services. See *Authorization C API Developer Reference*.

Any attribute added to the user credential can be used as ADI in a rule definition. There are also attributes that are built into the Security Access Manager user credential when it is created by the authorization engine. Just like attributes that can be added to the credential by the resource manager, the built-in credential attributes can be used in authorization rules. The built-in credential attributes include items of information, such as the user name (or the principal UUID). The attributes also include the groups (or the group UUID) of which the user is a member.

See the *Authorization C API Developer Reference* for a table of valid credential attribute names. All credential attribute names begin with `azn_cred_` (for example, `azn_cred_principal_uuid`). This table lists attribute names available in the Security Access Manager authenticated user credential, their value, and a description.

Many attributes in this table are also available in an **unauthenticated** user credential, except attributes related to the identity of a user. For example, attributes such as the user name, principal UUID, group name, and group UUID, and the LDAP DN for LDAP configurations are not available in an **unauthenticated** credential.

When developing rules that use these particular attributes, the authorization engine requires all ADI to be present before a rule can be evaluated. If the ADI is not available, the authorization decision is returned with an error status. Requiring the user to authenticate before accessing the protected object with such a rule attached ensures that the authenticated credential information is available. This requirement can be achieved with an ACL entry on the object that requires authenticated access.

Application context information

Authorization rules might require application context information to complete an evaluation. Context information includes information that is not an entitlement but is specific to the current transaction or operation.

An example is a transaction amount, such as purchase price or transfer amount. This information is passed to the decision through the `app_context` attribute list of the `azn_decision_access_allowed_ext()` call. Security Access Manager WebSEAL also uses this mechanism to pass the values of certain HTML tags and HTML request data (from a get or post request) into the access decision for use in a rule evaluation.

Authorization engine context information

If required, the authorization engine automatically provides context information before the authorization rule is evaluated. The ADI provided by the authorization engine includes the name of the protected object that is the target of the access decision. The ADI also includes the string of operations that the requesting user wants to do on the protected object.

The following attribute names are reserved for these data items:
- `azn_engine_target_resource`
- `azn_engine_requested_actions`

Dynamic ADI retrieval entitlement services

The final source for retrieving ADI is the dynamic ADI retrieval entitlements service. This class of authorization entitlement services is designed to retrieve ADI from an external source.

These services can be developed to retrieve ADI from an enterprise database that contains employee, customer, partner, or inventory information. The dynamic ADI retrieval service is called to retrieve ADI when the access decision is being made. Calling both at the same time has the benefit of being able to retrieve volatile data, such as quotas, at a time when its value is most current.

The Security Access Manager Attribute Retrieval Service (AMWebARS, **now deprecated**) is an example of a service that can retrieve ADI from external sources.

A replacement example uses the Web Service Description Language (WSDL) file in the Security Access Manager Application Development Kit to create and deploy a custom attribute retrieval service. See the *IBM Security Access Manager for Web: Installation guide* for more information about setting up the IBM Security Access Manager for Web custom attribute retrieval service. Also see the *IBM Security Access Manager for Web: WebSEAL Administration Guide* for more information about using the WSDL file.

Now deprecated, AMWebARS was the official package name for a Security Access Manager J2EE web service that implements a dynamic ADI retrieval service. To facilitate communication between the resource manager, which is starting the rules engine, and AMWebARS, which is done with SOAP over HTTP, the Security Access Manager runtime environment (pdrte package) provides an authorization entitlement service called `azn_ent_amwebars`.

See the *Authorization C API Developer Reference* for more information about developing with dynamic ADI retrieval entitlement services to fetch ADI when the rule is evaluated. See the *Administration C API Developer Reference* for an in-depth discussion of attribute lists, their formats, and the authorization APIs that are used to manipulate them. See "Format and constraints of rules" on page 127.

Volatile versus nonvolatile data

In general, the source for any particular piece of ADI depends largely on what the data is. The most important question is whether the data is volatile. For example, is it possible for the data to change during the lifetime of the session of the user? Is it important to use the most up-to-date information when it does change? Volatile data must be retrieved with a dynamic ADI retrieval service unless the resource manager application can provide this data.

Application-specific data that is nonvolatile and not user-specific is provided by the resource manager application. Data that is nonvolatile and user-specific is loaded into the user credential when the user is authenticated. The data is kept with the credential for the lifetime of the user session.

The set of data provided by the authorization engine, including the target protected object and permissions, is fixed and cannot be changed.

Authorization rule language

Extensible Style Language (XSL) is the language that specifies rules. Extensible Markup Language (XML) is used for the data that forms an input to the rules. The combination of XML and XSL provides a platform independent way to express both the inputs to the rules evaluator and the rules themselves.

XML also supports expressing complex data types in a structured and standard manner in text format. This text format allows rules for processing the XML data to be written without having to cater to platform and programming language specifics.

XSL is a functional style sheet language that can be used to do simple tasks or complex tasks that depend on your needs. XSL possesses an inherent ability to analyze and evaluate XML data, which is becoming the standard for data representation in e-business models. XSL is built on other XML-based standards such as XPath, which is the expression language at the core of an authorization rule.

To implement rules-based authorization policy, it is necessary to impose a number of constraints on the XSL rules. Constraints include the requirements that the output of the rule evaluation is simple text and that the output conforms to one of a known set of result strings. See "Format and constraints of rules" on page 127.

It is also necessary to impose constraints on the XML input document that is built as input to the rule evaluation. The ADI XML document model enables the authorization engine to detect when ADI is missing. The ADI might need to be requested from the resource manager or an external entity through the dynamic ADI retrieval service interface.

ADI XML document model

The ADI XML document model (or ADI XML model) is a set of restrictions placed on the XSL/XML model by the authorization rules implementation. The ADI XML model enables the interface to be simple and yet functional for authorization purposes.

The model constrains the authorization rules to function within a predetermined XML document format with the same top-level XML document element for all rules. The XML ADI is imported by the rules evaluator from credential attributes, from application context, or from other data sources. The XML ADI must be inserted into this XML document before authorization rules can use the data. Similarly to simplify the process of defining rules, the authorization rules must operate within the confines of the ADI XML model. The ADI XML model requires the XML document to contain the following top-level XML element. All target ADI for a particular rule evaluation is inserted in the top-level XML element. The `XMLADI` element is created automatically as part of the rule evaluation process by the authorization engine.

```
<XMLADI>
<!--   XML formatted ADI are inserted here.  -->
</XMLADI>
```

As a result of this restriction, the XPath to the data used in an authorization rule must include the prefix `/XMLADI` to access a particular data element within the model. For example, you might add an ADI item of `JohnSmith` to the document to access the fields of `JohnSmith` within the ADI XML document. In this case, specify the XPath `/XMLADI/JohnSmith` to access the data contained in the XML object `JohnSmith`.

An XPath is the path to a particular child element within the hierarchy of a structured XML data object. Much like a directory path on a hard disk drive is used to access a specific file, an XPath designation starts from the root of the document (in this case `/XMLADI`). The designation traces a path from this root down through its child elements to the specific element that is being referenced. For example, with the example entitlement JohnSmith in the "XML entitlement example" on page 123 as a reference, the JohnSmith XML object has a child element called CreditCard. The child elements of the CreditCard element are attributes which are common to most credit cards. To access Balance under the CreditCard element of JohnSmith, you would use the following XPath:

```
"/XMLADI/JohnSmith/CreditCard/Balance"
```

XPaths like this example are the means by which authorization rules access the ADI data values that are needed to make attribute-based authorization decisions.

All data elements are restricted to work within the ADI XML model. The authorization rules must also be restricted to operate on or match XPaths within the model. Therefore, XSL template match statements are also restricted to matching XPaths starting from /XMLADI within the ADI XML document. See "Format and constraints of rules" on page 127.

Containers and XML ADI container names

When data is requested from a resource manager, the granularity of the XML data returned is at the level of a single container of information.

The container is normally also the smallest data element (for example, elements that might be considered for billing purposes). This convention was adopted for the ADI XML model as well. The ADI that is used in authorization rules is also defined and manipulated as containers of XML data. For example, the JohnSmith XML object defined in "XML entitlement example" on page 123 is an example of an ADI container.

The top-most element in the definition of an item of ADI is termed the *container name* of that item of ADI. To define an authorization rule, always reference the XPath to the XML definition of data in any ADI container. Specify the name of the container as the first element /XMLADI in the XPath specification for the data element.

Returning to the example ADI item JohnSmith, you can assume that there is a container received from the data provider named JohnSmith. To access any element within the JohnSmith container, the XPath specification must be prefixed with JohnSmith. For example, JohnSmith/CreditCard/AcctNumber refers to the AcctNumber value. To access this information from within an authorization rule, this XPath must also be prefixed by the top-level element of the XML target ADI input document. The element is XMLADI (for example, /XMLADI/JohnSmith/CreditCard/AcctNumber). However, both of these XPaths are valid when used in an authorization rule. The validity is due to the default template match statement that is added to all authorization rules that do not explicitly include one. The default template match statement matches the ADI XML document from /XMLADI. JohnSmith can be referred to either with a relative reference or with an absolute reference that is prefixed with /XMLADI. See "Format and constraints of rules" on page 127.

Limitations of container names

One restriction imposed by the ADI XML document model is that each item of ADI consumed by the rules evaluator must have a unique container name. The container name must not be confused with containers provided by other entitlements data providers.

For example, two different data providers might provide a data item called TxInfo. The rules evaluator does know to which provider it must make a request to get this item of data. To help differentiate items of ADI with the same name, XML provides defining namespaces for data. The namespace ID of the namespace can then be used to differentiate one ADI element from another. For TxInfo, you might define a namespace companyA and reference this instance of ADI with companyA:TxInfo. See "XML namespace definitions" on page 124.

This restriction on naming containers among data providers is not enforced by the authorization engine. If the engine encounters multiple instances of the same item of ADI (for example, TxInfo), it adds them all to the ADI XML document for use in the evaluation. In the ADI XML document, there can be two items of ADI data with the same container name within the ADI XML input document. The assumption is then made that they are structured in the exact same way. For example, a particular application request might involve a number of individual transactions, each with its own transaction amount. An authorization rule can be formulated to add all these items together. The rule compares the sum of the items to a predefined total transactions limit or to a per-transaction limit with an XSL node select statement. "Example: ADI from dynamic ADI retrieval services" on page 131 in the "Examples of authorization rules" on page 129 section shows an example rule. The rule sums multiple transaction elements in this way and even counts the number of instances of a particular ADI element.

XML access decision information

By default, the rule evaluator automatically transforms into XML format any name-value pair attributes passed to it by the calling application. The attributes were identified as target access decision information (ADI) for the current evaluation.

When transforming the attribute to XML, the attribute name is used as the container name of the XML data item. The attribute value is converted into an XML value. The container name of an item of ADI equates to the XML element name in the XML definition. For example, the following XML data is generated for attribute name VPS_CREDIT_CARD with a string attribute value of 5517 3394 8324 0965:

```
<VPS_CREDIT_CARD>5517 3394 8324 0965</VPS_CREDIT_CARD>
```

The container name and XML element name in this case is VPS_CREDIT_CARD. The graphical user interface, the command-line interface, and the Security Access Manager authorization API attribute list interfaces do not permit the administrator to define rules that contain invalid XML container names.

The application might pass entitlements or application context that are already formatted as XML for an access decision. In this case, the authorization rules evaluator expects the data to be of type azn_string_t and expects the format of the string to be XML. The attribute name must match the container name of the XML data item. If the names do not match, the evaluator does not evaluate the rule correctly.

The evaluator identifies XML format data by locating the less than (<) character at the beginning of the attribute value. If the attribute value does not begin with a less than character, the data is not considered to be an XML data item. The evaluator attempts to convert the data item to XML format automatically. This means of identification is used only on attributes or application context identified as target ADI for the access decision. Non-XML attribute values that start with a less than character cannot be used by the application. An error status is returned from the authorization decision. If the data is not correct XML, the XSL processor fails and returns an error to denote the failure.

Data items that must be defined in XML must be entirely defined in XML. Definition must not rely on the translation mechanism for non-XML items to generate the appropriate XML element name automatically. For example, to define an attribute to contain the XML definition of MY_CREDIT_CARD_NUM, you must add an attribute with the attribute name MY_CREDIT_CARD_NUM. The attribute value for MY_CREDIT_CARD_NUM is:

```
<MY_CREDIT_CARD_NUM>5517 3394 8324 0965</MY_CREDIT_CARD_NUM>
```

By defining the XML element as opposed to defining only its value, XML attributes can be added to the element definition. The addition does not affect the name by which the ADI is referred to when talking with data providers.

In the following definition of the XML item MY_CREDIT_CARD_NUM, the CardType XML attribute has the value of "visa". XML attributes are defined in the element start tag of the element to which they apply. XML attributes are equivalent to any other first-level child element of the XML object. To reference the attribute CardType, the required XPath is:

```
/XMLADI/MY_CREDIT_CARD_NUM/CardType
```

XML attributes must not be confused with the authorization API attributes and attribute lists that are used to carry data into and out of the authorization process.

```
<MY_CREDIT_CARD_NUM CardType="visa">
   5517 3394 8324 0965
</MY_CREDIT_CARD_NUM>
```

The ability to add XML attributes to an element definition is useful when it comes to defining a namespace for the data item. See "XML namespace definitions" on page 124.

If the ADI attribute contains multiple attribute values (string, XML, or any combination), the evaluator converts each attribute value as a separate instance of ADI. For example, the TxData attribute has values of 100 and 500. The evaluator inserts the following XML item declarations into the ADI XML document:

```
<TxData>100</TxData>
<TxData>500</TxData>
```

The policy administrator can then design an authorization rule that uses XSL language node selection statements to work with these two values independently. Alternatively, the authorization rule can add the values and compare the total with some predefined limit. If TxData is compared to a value, it is treated as a node set comparison where each TxData value is compared to the data. Success is indicated when any of the TxData values equal the target data. Node set comparisons have slightly different behavior than expected when using the != operator. In most cases, use the not() function instead. For information about when to use != and not() when comparing a node set, see "Example: ADI from dynamic ADI retrieval services" on page 131.

XML entitlement example

The following example is an ADI XML document that might be passed to the XSL processor from the rules evaluator during the evaluation of an authorization rule.

The document contains two containers: JohnSmith and AmountReqd. The attribute value of the container JohnSmith is defined in XML. The AmountReqd container is translated to XML from an incoming string application context attribute. The container JohnSmith is an entitlement and the container AmountReqd is an item of transaction context.

The authorization rules evaluator automatically encompasses all the data under the XML top-level node declaration XMLADI when the ADI XML document is created. This top-level element was added for clarity.

The XML document that is passed to the evaluation routines by the authorization rules evaluator is as follows:

```
<XMLADI>
  <JohnSmith>
    <CreditCard>
      <AcctNumber>0123456776543210</AcctNumber>
      <Limit>10000.00</Limit>
      <Balance>2000.00</Balance>
    </CreditCard>
    <MileagePlus>
      <MemberStatus>100k</MemberStatus>
      <CardNumber>12345678</CardNumber>
    <MileagePlus>
  <JohnSmith>
  <AmountReqd>500.00</AmountReqd>
</XMLADI>
```

When referencing a particular ADI item in the XMLADI document available to a rule, the XPath path specifier can begin from the container name of the XML element. For example, the name might be JohnSmith. The default template rule matches the /XMLADI element automatically. If the callers want to specify their own template match statement explicitly, they can do so.

In this example, the ADI container names are JohnSmith and AmountReqd. See "Format and constraints of rules" on page 127.

XML namespace definitions

XML namespaces differentiate between XML items with the same name. XML namespaces also group XML data of the same type or function. The same principles can be used with ADI that is defined for use with authorization rules.

For example, a customer database and a product inventory database might both define ADI called name that might be used in authorization rules. By defining an XML namespace with the namespace ID item, you can differentiate between the two instances of name by calling the ADI from the product database item:name. This example provides a namespace definition for the item namespace:

```
xmlns:item="http://mycompany/namespaces/items
```

where xmlns is a standard XML attribute name and item is the namespace ID chosen for the namespace. The URI following the = is used to distinguish one namespace ID from another.

This namespace declaration associates the namespace ID item with the URI string:

```
http://mycompany/namespaces/items
```

The value of the URI string is of no consequence to the XML and XSL processors but it must be unique. Unlike the XML and XSL processors, the Security Access Manager authorization engine does not permit two namespace IDs to be assigned the same URI value. The Security Access Manager authorization engine uses the

URI to uniquely identify the namespaces. Defining two namespaces with the same URI results in an initialization error. The authorization application cannot start, and an error is logged to the error log of the application.

The source from which the item name is to be obtained must be aware of this relationship. The source must be able to make the connection between the item:name requested by the authorization engine and the name data stored in the product database. The source must also be able to provide this data to the authorization engine in an attribute called item:name when it is needed. For example, a dynamic ADI retrieval service must understand that, when it is asked for item:name, it must fetch the required value by looking for name in the product database. The service needs to return the data to the authorization engine in an attribute called item:name. When an application uses namespaces to differentiate or aggregate ADI items, it is required to define the namespace for both the XML and XSL processors.

To define a namespace for the XSL processor, add the namespace definition to the xsl-stylesheet-prolog configuration file entry described in "input-adi-xml-prolog and xsl-stylesheet-prolog" on page 135. This example shows how to add a namespace definition for the item namespace to the xsl-stylesheet-prolog entry:

```
xsl-stylesheet-prolog = <?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
     xmlns:item="http://mycompany/namespaces/items" version='1.0'>
<xsl:output method = 'text' omit-xml-declaration='yes'
     encoding='UTF-8' indent='no'/>
<xsl:template match='text()'>
</xsl:template>
```

There are two ways to define a namespace prefix to the XML processor:

- Define the namespace globally for the entire XMLADI document.
- Define it individually in those ADI items that use the prefix.

In both cases, the namespace declaration must be included in the start tag for the XML element.

The first and simplest method of defining a namespace for the XML processor is to add the namespace definition to the XMLADI document element start tag. Adding the definition to the XMLADI document element start tag is easiest to do because it automatically defines the namespace for the entire document. Any ADI items in the document whose names are prefixed with this namespace ID do not have to have the namespace definition added to their own element start tag.

This method does not suffer any of the drawbacks of defining the namespace by using the second method. The [xmladi-attribute-definitions] stanza was added to the configuration file to allow customers to define namespaces globally for use in the XMLADI document. For information about how to add a namespace definition to the [xmladi-attribute-definitions] stanza, see "[xmladi-attribute-definitions]" on page 136.

The second method of specifying an XML namespace definition to the XML processor is to add the definition to the XML value of the ADI element. For example, to add the XML namespace definition to the item:name XML item with a string value of Widget A, you would define item:name in XML as follows:

```
<item:name xmlns:item="http://mycompany/namespaces/items">
  Widget A
</item:name>
```

The ADI `item:name` must be added to an attribute list with the `item:name` attribute name. Its value is the entire XML element definition in the example entered as a single contiguous text string.

There are some drawbacks to defining the XML namespace in the XML definition of each ADI item rather than defining it globally for the entire XML ADI document. For instance, the value of any ADI items that use a namespace ID prefix must be in XML. The namespace definition can be added only to the XML definition of the value of the item, as demonstrated for `item:name` in the example. As a result, items of ADI with namespace prefixes cannot have the value `100`. The value of the item must be an XML fragment, such as the string `<prefix:adi_name>100</prefix:adi_name>`.

Any ADI source that can provide values for namespace prefixed ADI items must ensure that the appropriate namespace definitions for the item are added to each XML formatted value that it returns. When the service does not normally return XML formatted data and is not aware of namespace prefixes, you must changed it. The change causes an increased processing load for dynamic ADI retrieval services.

By defining the namespace globally, you can avoid all of these complications. If a namespace is not defined for either the XML or XSL processors, an error is logged to the application error logs. The error indicates that the namespace ID does not have an associated URI mapping. This problem might occur during the creation of the rule if the XSL processor is not notified of the new namespace. The problem might occur during rule evaluation if the XML processor is not notified.

Authorization rules evaluator

The authorization rules evaluator evaluates authorization rules within the constraints that are required by the authorization engine.

The authorization rules evaluator takes the rule policy that is attached to the target protected object and evaluates the rule by calling the XSL processor. The input XML document for the transformation contains a definition for how the authorization engine can retrieve one of the following sources for the ADI:

- User credential entitlements that request the authorization
- Application context information that is passed in by the access decision call (passed in by the resource manager)
- Security Access Manager authorization engine context
- Dynamic ADI retrieval entitlement services

The authorization engine expects the rules evaluation to result in the return of one of the string identifiers as shown in Table 20. These identifiers ensure uniqueness when an XSL rule is written incorrectly and the evaluation returns incorrect information. Delimiting the identifiers with an exclamation point (`!`) enables the evaluator to identify errant cases.

Table 20. String identifiers returned by rules evaluation

Delimiter	Meaning
`!TRUE!`	Access is permitted.
`!FALSE!`	Access is denied.
`!INDIFFERENT!`	The rules engine has no opinion.

The identifiers must be the only text in the output document, although they can be surrounded by white space. A value other than the defined valid values or an empty document might be returned. In this case, the access decision fails and an error code is returned to the resource manager to indicate that the rule is not compliant. The format of an authorization rule is outlined in "Format and constraints of rules."

In addition, the maximum length of any result text that is returned by a rule evaluation is limited to 1023 characters. Rules returning more text than this limit cause the access decision to fail at run time with a minor error code of `ivacl_s_rule_result_string_too_large`.

Format and constraints of rules

An authorization rule must be defined as an XSL template in an XSL style sheet with the style sheet prolog that is specified in the configuration file.

The rule must be written in a valid XSL template rule format and must return a text document that contains one of the following string identifiers:

- `!TRUE!`
- `!FALSE!`
- `!INDIFFERENT!`

The identifiers must be the only text in the output document but they can be surrounded by white space. The identifiers are not case-sensitive. A value other than one of the identifiers listed or an empty document might be returned. The access decision fails and an error code is returned to the resource manager to indicate that the rule is not compliant. See "Authorization rules evaluator" on page 126.

For authorization decisions, the rule must return the expected decision data to the rules evaluator. The data that is returned from the rules-driven entitlements interface must be able to be expressed as a text name-value attribute pair in the entitlements output parameter of the `azn_entitlement_get_entitlements()` method. Many data providers return entitlements data in XML format. No additional transformation is required to pass these entitlements into the rules evaluator as ADI.

All ADI that is passed to the rules evaluator must be specified in XML. Non-XML ADI that is passed to the access decision or retrieved from the credential is formatted into XML by the evaluator before an authorization rule can be evaluated.

The result of the XSL transformation done by an XSL authorization rule must be a text output document that contains only one of the supported string identifiers.

The following example references the XML data item that is defined in `JohnSmith`. The condition that the following example rule evaluates is expressed, as follows:

```
if ((AmountReqd + Credit Card Balance) < Credit Card Limit
     && MileagePlus Status is "100k")
```

The corresponding authorization rule is:

```
<xsl:if test="(AmountReqd + JohnSmith/CreditCard/Balance)
  &lt; JohnSmith/CreditCard/Limit
    and JohnSmith/MileagePlus/MemberStatus = '100k'">
  !TRUE!
</xsl:if>
```

This example rule is the simplest form for specifying an authorization rule. It does not include its own template match statement and it accepts the default template match statement, which is set to /XMLADI. Template match statements are an XSL language construct that is used to select the point in the hierarchy of an XML document at which the XSL rules, which are contained within the template match statement, are applied. The default template match statement of the ADI XML model matches from the top of the XMLADI document by specifying the XPath /XMLADI.

To add your own template match statement to a rule definition, only two additional lines are needed. For example, you might rewrite the example to include your own explicit template match statement that matches from the root of the XMLADI document. Modify the rule as follows:

```
<xsl:template match="/XMLADI">
<xsl:if test="(AmountReqd + JohnSmith/CreditCard/Balance)
  &lt; JohnSmith/CreditCard/Limit
    and JohnSmith/MileagePlus/MemberStatus = '100k')
  !TRUE!
</xsl:if>
</xsl:template>
```

To reference any data item in the document, the XPath to each node must include the XMLADI node. For example, to access the credit card balance, the full path would be /XMLADI/JohnSmith/CreditCard/Balance. When a rule is built, the rule writer must understand what the correct XPath is from the current point in the tree. The XPath accesses the XML data nodes and subnodes. The current point in the tree is selected with the template match statement. The template match statement allows an XSL programmer to shorten the XPath to each data element by specifying that the XPath processing occur further down the XML document tree.

The <xsl:template match="/XMLADI"> statement tells the XSL processor that all relative XPaths within the bounds of the template statement must be assumed to be relative to the node XMLADI. To shorten the XPaths even further, the template match statement can be set at /XMLADI/JohnSmith. In this case, the credit card balance might be termed CreditCard/Balance.

Policy administrators must also make the following assumptions about the XSL style sheet document that is created by the rules evaluator to contain the rule that they devise:

- If a style sheet prolog is specified in the azn client configuration file, that prolog is imported into the empty style sheet. If no prolog is specified, the following default prolog is used instead:

  ```
  <!-- Required for XSLT language -->
  <?xml version="1.0" encoding='UTF-8'?>
  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
            version="1.0">

  <!-- Required to constrain output of rule evaluation -->
  <xsl:output method="text" omit-xml-declaration="yes"
            encoding='UTF=8' indent="no"/>

  <!-- Need this to ensure default text node printing is off -->
  <xsl:template match="text()"></xsl:template>
  ```
- Among other things, this prolog sets the XSL style sheet syntax to version 1.0, which is supported by the embedded XSL processor. The prolog sets the namespace for XSL documents to xsl, which requires that all XSL language-specific identities be prefixed by xsl:. This prefix is the standard mode of operation for XSL style sheets. Most attributes in this prolog must be in the style sheet. If not, the results that are returned from the rules evaluator do not conform to the expected results.

- All authorization rules must be enclosed in an `xsl:template` match statement. If the rule is defined with its own `xsl:template` match statement, the rule is accepted as is. This acceptance allows the rule creator to specify the level within the ADI XML document at which the rule matches data items. In this case, the match statement must be the first statement encountered by the evaluator when validating the rule. Otherwise, it is assumed that there is no template match statement. If there is a match statement but the match statement does not begin with the `/XMLADI` absolute path, the rule is returned as not valid. Relative match statements are not accepted at this level.
- If no match statement is specified in the rule, the rule is automatically enclosed in the following match statement:

  ```
  <xsl:template match="/XMLADI">
     ...
  <xsl:template>
  ```
- Therefore, all rules devised without an explicit template match statement must use XPath expressions that assume the XML context node is `/XMLADI`. The XPath expression for any ADI item must begin with the container name of the item and must be fully qualified.

Note: Authorization rules are processed internally with a recursive algorithm. Avoid creating very long logical expressions that can exhaust the system stack. Long expressions are problematic on systems that have a small stack by default.

For example, this rule might exhaust the system stack:

```
<xsl:choose><xsl:when test="(((azn_cred_principal_name='9410431')) \
 or ((azn_cred_principal_name='user1')) \
 or ((azn_cred_principal_name='user2')) \
 ...
 or ((azn_cred_principal_name='user500'))\
 )">!TRUE!</xsl:when><xsl:otherwise>!FALSE!</xsl:otherwise></xsl:choose>
```

A better option is to add the users to a group and write a rule to check group membership.

As a preventive measure two new options are added:

- `xsl-max-logical-expressions`
- `xsl-eval-expressions-check`

The `xsl-max-logical-expressions` option limits the number of logical 'or' and 'and' operators in a logical expression when a rule is created. The `xsl-eval-expressions-check` option uses the value from the `xsl-max-logical-expressions` option to limit the same logical operators when a rule is evaluated.

See the `[aznapi-configuration]` stanza in *IBM Security Access Manager for Web: Plug-in for Web Servers Administration Guide*.

Examples of authorization rules

These examples demonstrate how rules can be implemented.

Example: ADI from resource manager

This example relies mostly on ADI that is passed in to the access decision call. The example also requires an ADI container called `printQuota` to be stored in the requesting user credential or passed in as application context.

The access decision logic defined by this rule is to permit access only when one of the following conditions is true:

- The user is in the printUsers group.
- The user requests a print operation (p).
- The user requests to queue a print job for printing later (q) and the print quota is less than 20.

```
<xsl:if test='azn_cred_groups = "cn=printUsers,o=ibm,c=us"
    and (contains(azn_engine_requested_actions,"p")
      or contains(azn_engine_requested_actions,"q"))
    and printQuota &lt;20'>
  !TRUE!
</xsl:if>
```

The test condition for the group name returns an appropriate result regardless of the number of groups that the requesting user is in. The condition is an XSL node test that compares each value within the XML element azn_cred_groups with the DN string. To determine the opposite case (for example, that the requesting user is not in the printUsers group), the syntax requires a slightly different expression. See "Example: ADI from entitlement data" for an example of how to test for whether a set of values like the azn_cred_group_names attribute does not contain a certain member.

Example: ADI from entitlement data

This example shows how a rule works on data that is in the authorization credential.

It evaluates the following attributes:
- azn_cred_principal_name
- azn_cred_groups
- azn_cred_registry_id

Each of the xsl:when statements are evaluated. The first statement with conditions that are all true returns a result. Each condition tested has a comment that explains its action.

```
<!-- Example choose rule -->

<xsl:choose>
  <!-- Explicitly allow if the requesting user is myuser0 -->
  <xsl:when test="azn_cred_principal_name = 'myuser0'">
    !TRUE!
  </xsl:when>

  <!-- Explicitly deny if the requesting user is myuser1 -->
  <xsl:when test="azn_cred_principal_name = 'myuser1'">
    !FALSE!
  </xsl:when>

  <!-- Explicitly allow if the requesting user's LDAP DN  -->
  <!-- is the same as that specified   -->

  <xsl:when test="azn_cred_registry_id = 'cn=myuser3,secAuthority=Default'">
    !TRUE!
  </xsl:when>

  <!-- This rule permits access to any user who is a member of mygroup1 -->
  <!-- but is not a member of mygroup2     -->

  <xsl:when test="azn_cred_groups = 'mygroup1'
    and not (azn_cred_groups = 'mygroup2')">
    !TRUE!
  </xsl:when>
  <xsl:otherwise>
    !FALSE!
  <xsl:otherwise>
</xsl:choose>
```

The fourth xsl:when statement uses the not() function to negate the Boolean result of the following test:

```
azn_cred_groups = 'mygroup2'
```

The not() function is used instead of the valid authorization rule operator != operator because, in this case, the azn_cred_groups attribute is a multi-valued attribute. Multi-valued attributes like azn_cred_groups return a set of values, referred to as a *node-set* in XSL, to be tested by the condition. Each node value in the set is tested against the condition individually and !TRUE! is returned if any of the conditions evaluate to true. In any case, where the user is in more than one group, other than mygroup2, the result of the node test is always !TRUE!. To test the nonexistence of something in a node-set, use the not() function instead of the != operator. For example, you can test that the condition group is mygroup2 is not true.

Example: ADI from dynamic ADI retrieval services

This example evaluates an application-defined XML input document that is provided by a dynamic entitlement service that was written with the dynamic ADI retrieval service.

The code that must be written might create a batch object that contains a list of operations that are to be done together. The batch object consists of a number of transaction elements. Each transaction consists of an item and the amount of those items to order.

With these assumptions, the following XML object might be used as input for making the authorization decision:

```
<!-- batched transaction -->
<batch>
  <max_tx_count>5</max_tx_count>
  <max_tx_amount>150</max_tx_amount>
  <account>customerA</account>
  <transaction>
    <item>widgetA</item>
    <amount>10</amount>
  </transaction>
  <transaction>
    <item>widgetB</item>
    <amount>20</amount>
  </transaction>
  <transaction>
    <item>widgetC</item>
    <amount>30</amount>
  </transaction>
  <transaction>
    <item>widgetD</item>
    <amount>40</amount>
  </transaction>
  <transaction>
    <item>widgetE</item>
    <amount>50</amount>
  </transaction>
</batch>
```

With this expected XML object, you might create the following authorization rule:

```
<!--Compare group to batch customer and num tranactions
    and total tx amounts to limits.-->
<xsl:if test="azn_cred_groups = batch/account
      and count (batch/transaction) &lt;= batch/max_tx_count
      and sum (batch/transaction/amount) &lt;= batch/max_tx_amount">
    !TRUE!
</xsl:if>
```

The authorization rule checks that the requesting user is a member of a group whose name matches the name of the account in the transaction. In this example, it is `customerA`. If the requesting user is not a member of this group, the user is not authorized to submit batch requests on behalf of `customerA`. Then, the rule checks that the total number of transactions within the batch is less than or equal to the `max_tx_count` element of the batch object. The rule also checks that the total number of items ordered in the entire request is less than the `max_tx_amount` element of the batch object. The rule calls the `count()` and `sum()` functions. The `count()` function counts the number of instances of a transaction element within the batch. The `sum()` function totals the value of all the amount elements within all transaction elements in the batch.

For additional information of creating authorization rules, see the *Authorization C API Developer Reference*.

Methods of providing ADI to the rules evaluator

A resource manager application can provide ADI from the resource manager to the rules evaluator in one of two ways. One method is to add the attributes to the application context parameter. The other method is to configure the rules evaluator to supply the missing ADI to the authorization engine only when it is explicitly requested.

The first method is to provide the ADI by adding the attributes to the application context parameter passed to the `azn_decision_access_allowed_ext()` method. The problem with this method is that the resource manager must know which ADI is going to be needed by a particular access decision. Alternatively, you can provide all the ADI for all known rules to the authorization engine for every access decision call regardless of whether a rule is involved in the decision.

The first method might be acceptable and even desirable for a smaller set of ADI. However, for a larger and more varied set of possible ADI, a second method is needed. You can configure the resource manager to supply the missing ADI to the authorization engine only when it is explicitly requested. With this method, the authorization engine can be configured with a set of ADI prefixes that can be provided by the resource manager upon request. The authorization engine fails the access decision and notifies the resource manager of the ADI it needs in a permission information attribute returned by the `azn_decision_access_allowed_ext()` method. The attribute contains a list of the ADI that is needed to successfully evaluate the rule. The ADI was not found in the application context that was passed in. The ADI also did not have a prefix that matches those prefixes that the resource manager identified as its own.

The permission information attribute is named `azn_perminfo_rules_adi_request` and contains a text attribute value for each item of ADI required. The resource manager looks for this attribute when the access decision fails. When it is present, the resource manager scans the list of ADI names in the attribute and gathers the requested data to try the access decision with this additional data again. If the requested data cannot be provided, the resource manager must deny access and log the problem as a failure due to insufficient rules data. The requested list contains only the ADI items that are identified as being provided by the resource manager. The unique prefix added to the attribute name is used to identify the ADI. All resource managers that provide data to the evaluation process in this manner must define a unique prefix by which their ADI data set can be identified.

Permission information is returned to a resource manager application only when the authorization client was configured that way. To activate the return of the azn_perminfo_rules_adi_request permission information attribute, the name of this attribute must either be added to the azn_init_set_perminfo_attrs initialization attribute or the equivalent permission-info-returned entry in the [aznapi-configuration] stanza.

The ADI prefixes that are recognized by the resource manager can be configured using the resource-manager-provided-adi entry or the azn_init_resource_mgr_provided_adi initialization attribute. See "resource-manager-provided-adi" on page 134 and for the initialization attribute, see *Authorization C API Developer Reference*.

The authorization engine attempts to anticipate the need to request information from the resource manager by obtaining the rule policy object on the protected object early in the access decision process. The authorization engine then compares the required ADI in the rule with the ADI names in the application context parameter that is passed by the resource manager. The ADI names, which are missing from the application context and which are specific to the resource manager, are added to the returned permission information object.

ADI prefixes must be unique to identify them as resource manager ADI and to avoid conflict with ADI provided in the credential from the authorization engine or from an external data provider.

Reason codes for rule failures

This feature allows the target application to fail or permit the access request based on the rule failure reason code it is given by the resource manager.

The authorization engine processes all policies for the access decision as normal. If the rule evaluation fails, the engine returns access denied with a reason code in the azn_perminfo_reason_rule_failed permission information attribute list.

When access is denied, the application must check the permission_info attribute list returned from the access decision call. The application determines whether a rule failure reason code was returned from the access decision. The resource manager does not need to check for the attribute on a successful access decision call. The Security Access Manager application is an example of an aznAPI resource manager that can use the rule failure reason code. When configured, Security Access Manager forwards the reason code to the protected web application. The protected web application must be mounted through a secure junction to have access to the reason code defined for the authorization rule. The use of rule failure reason codes in Security Access Manager is limited to the protected object space of junctioned web applications.

The attribute value (the reason code) of the azn_perminfo_reason_rule_failed attribute is a single string. The value is determined and defined by the policy administrator and is set in the rule policy object when it is first created. The only constraint on the value of the reason code is that the value must be a string.

The following conditions must be met before a rule failure reason code is returned to the caller:

- The reason code is returned only when the access request is denied and the rule policy evaluation denies access. However, the reason code is not returned for every case in which access is denied. The reason code is not returned when the rule evaluation succeeds. The rule failure reason code is not returned if the rule failed due to a rule syntax error. The code is not returned if there was insufficient ADI to do the rule evaluation. In the latter cases, the authorization decision is failed with an error status.
- There must be a reason code set in the attached rule policy object. This value is set in the rule policy with the admin API or the **pdadmin** utility.
- The aznAPI application must be enabled to return the rule failure reason as permission information. To do this action, either the `azn_init_set_perminfo_attrs` initialization parameter or the equivalent configuration file entry in the `[aznapi-configuration]` stanza (stanza entry `permission-info-returned`) must include the attribute name `azn_perminfo_reason_rule_failed`. This feature enables the attribute to be returned by the authorization engine in the permission information output parameter (`perminfo`) of `azn_decision_access_allowed_ext()`. See the *Authorization C API Developer Reference*.

Configuration file and initialization attributes

A number of configuration file entries and initialization attributes are available to control aspects of the initialization of the rules evaluator within the authorization engine. The configuration entries are in the configuration file of the resource manager.

An example of this `aznAPI.conf` configuration file is provided in the `example/authzn_demo/cpp` directory of the Security Access Manager Application Developer Kit (ADK) package. Configuration files are also used by Security Access Manager resource management applications. These configuration entries can be added to the configuration file of these applications. See the application configuration file documentation for the specific Security Access Manager application.

Initialization attributes are the programmatic equivalent of configuration attributes and are intended to be used to develop a custom resource manager application. The authorization-rule-specific initialization attributes and the process of developing a custom resource manager aznAPI application are described in the *Authorization C API Developer Reference*.

resource-manager-provided-adi

The `resource-manager-provided-adi` configuration stanza entry defines the prefixes that the authorization engine uses to determine the set of missing ADI from the resource manager.

This entry uses a string prefix as its value. To specify more than one prefix, you must add multiple stanza entries as in the following examples:

```
resource-manager-provided-adi = sales_customer_
resource-manager-provided-adi = sales_item_
```

These examples notify the authorization engine that any ADI it requires that begins with `sales_customer_` or `sales_item_` is provided by the resource manager application. ADI items named sales_customer_name, sales_customer_address, sales_item_count, and sales_item_price are examples of ADI that the authorization engine would request from the resource manager.

dynamic-adi-entitlement-services

The dynamic-adi-entitlement-services configuration entry lists the service IDs of the dynamic ADI retrieval entitlement services. These services must be called by the authorization engine if ADI is missing from the requesting user credential or from the application context and cannot be gathered from the resource manager.

Any entitlement service configured under this entry is called by the authorization engine with the azn_entitlement_get_entitlements() interface and is passed the azn_perminfo_rules_adi_request attribute. The string values of this attribute are the container names of the ADI that are still required. If the dynamic ADI retrieval service can fulfill the request, it returns the requested data to the authorization engine in the entitlements parameter. Security Access Manager provides demonstrations of how an entitlement service can perform the functions of a dynamic ADI retrieval service and a credential attribute retrieval service. See the *Authorization C API Developer Reference*.

To specify that the authorization engine must call multiple dynamic ADI retrieval services, you must specify multiple entries. The following examples demonstrate how to specify the service IDs of two different entitlement services for use as dynamic ADI entitlement services. The service IDs must correspond to valid entitlement service definitions in the [aznapi-entitlement-service] stanza.

```
dynamic-adi-entitlement-services = ent_cred_attrs_id
dynamic-adi-entitlement-services = ent_svc_demo_id
```

input-adi-xml-prolog and xsl-stylesheet-prolog

You can use the input-adi-xml-prolog and xsl-stylesheet-prolog configuration entries to change the XML and XSL prolog statements. These statements are appended to the ADI XML document and authorization rule style sheet before they are passed to the rules evaluator for processing.

The format and defaults for each of these entries are:

```
input-adi-xml-prolog=<?xml version="1.0" encoding="UTF-8"?>
```

and

```
xsl-stylesheet-prolog=<?xml version="1.0" encoding='UTF-8'?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version='1.0'>
<xsl:output method = 'text'omit-xml-declaration='yes' encoding='UTF-8'indent='no'/>
<xsl:template match='text()'>
</xsl:template>
```

Due to the constraints imposed by the authorization rule model, there are a number of prolog attributes that are required by the authorization engine. All of the attributes are specified in the default prolog entries. If any of these attributes are changed or omitted from the entry, the authorization client fails to start and returns an error.

Note: Ensure that you are familiar with the Xalan XSL processor and the Xerces XML processor. Be familiar with the use of prolog statements before you change these entries from the default values.

[xmladi-attribute-definitions]

The [xmladi-attribute-definitions] stanza enables customers to add XML attribute definitions, such as XML namespace definitions, to the XMLADI document start tag.

For example, an application might want to use namespaces to differentiate or aggregate ADI items, as described in “XML namespace definitions” on page 124. The XML processor must be notified of the namespace with an XML namespace definition. The namespace definition can be added to this stanza, and it is automatically added to the XMLADI document element start tag. The benefit of adding definitions to the XMLADI document start tag is that the attribute definitions are available for all ADI items that are defined in the XMLADI document. The attribute definitions are available whether their values were retrieved from the credential, generated by the authorization engine, or retrieved by a dynamic ADI entitlement service. For example:

```
[xmladi-attribute-definitions]
  xmlns:myNS = "http://myURI.mycompany.com"
  appID = '"Jupiter" - Account Management Web Portal Server #1.'
```

The XMLADI element start tag that results from these definitions is:

```
<XMLADI xmlns:myNS="http://myURI.mycompany.com"
  appID='"Jupiter" - Account Management Web Portal Server #1.'>
```

Both the namespace ID myNS and the attribute appID are defined globally in the XMLADI document.

Manage authorization rules

These topics describe how to use the Web Portal Manager or **pdadmin** utility.

For online help while usingWeb Portal Manager, click the question mark to open a separate help window for the current page.

Note: There are no equivalent **pdadmin** commands for importing, exporting, or cloning authorization rules.

Note:

1. When providing rule text with the **pdadmin** utility, enclose the rule text in double quotation marks ("). Double quotation marks embedded within the rule text must be escaped with a backward slash (\) so that they are ignored by the **pdadmin** utility. The XSL processor treats single and double quotation marks equally for defining text strings. They can be used interchangeably, but they must always be paired appropriately. For example:

   ```
   pdadmin sec_master> authzrule create testrule1
     "<xsl:if test='some_piece_of_ADI =\"any string\"'>!TRUE!</xsl:if>"
   ```

Create an authorization rule

You can create an authorization rule with the Web Portal Manager or **pdadmin** utility.

Creating an authorization rule with Web Portal Manager

You can create an authorization rule with Web Portal Manager.

About this task

To create an authorization rule, complete the following steps.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule** > **Create AuthzRule** to display the Create AuthzRule page.
3. Type the **AuthzRule Name** for the authorization rule that you want to create (for example, r2).

   Note: Do not use the following characters in the name of a rule:

   ```
   ! " # & ( ) * + , ; : < > = @ \ |
   ```
4. In the **Description** field, type a description of the authorization rule. For example, type the following text:

   ```
   time-of-day rule for engineering object space
   ```
5. In the **AuthzRule Text** field, type the text of the rule policy. For example, type the following information:

   ```
   <xsl:template match="/XMLADI">
     <xsl:if test="(AmountReqd +JohnSmith/CreditCard/Balance)
       <JohnSmith/CreditCard/Limit
         and JohnSmith?mileagePlus/MemberStatus='100k'>
       !TRUE!
     </xsl:if>
   </xsl:template>
   ```
6. In the **Fail Reason** field, type the text that you want to be returned to the resource manager if the rule denies access to a protected object. For example, type `error`.
7. Click **Create**. If successful, the new rule is displayed as a link on the Manage AuthzRules page. If you select the authorization rule link, the properties of that rule are displayed.

Creating an authorization rule with pdadmin

You can create an authorization rule with the **pdadmin** utility.

About this task

To create an authorization rule with the **pdadmin** utility, complete the following steps.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **authzrule create** command.

Example

When providing rule text with the **pdadmin** utility, enclose the rule text in double quotation marks ("). Double quotation marks embedded within the rule text must be escaped with a backward slash (\) so that they are ignored by the **pdadmin** utility. The XSL processor treats single and double quotation marks equally for defining text strings. They can be used interchangeably, but they must always be paired appropriately. For example:

```
pdadmin sec_master> authzrule create testrule1
"<xsl:if test='some_piece_of_ADI =\"any string\"'>!TRUE!</xsl:if>"
```

See the *IBM Security Access Manager for Web: Command Reference*.

Modify an authorization rule

You can modify an authorization rule with the Web Portal Manager or **pdadmin** utility.

Modifying an authorization rule with Web Portal Manager

You can modify an authorization rule with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule** > **List AuthzRule** to display the Manage AuthzRules page.

   A list of authorization rules that were created in Security Access Manager are displayed. Each rule is a link that displays properties for that rule when selected.
3. Click the authorization rule link for the rule that you want to change. The AuthzRule Properties page is displayed.
4. As needed, change the following information:
   - The description
   - The authorization rule text
   - The fail reason

   For example, if no description currently exists, add a description. If a description currently exists, change the authorization rule description by typing the new description in the **Description** field (for example, adding the words `updated June 23 2003`):

   ```
   updated June 23 2003 time-of-day rule for engineering object space
   ```
5. Click **Apply** for the changes to take effect.

Modifying an authorization rule with pdadmin

You can modify an authorization rule in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **authzrule modify** command.

Example

For example, to change the rule named `r2` to return a fail reason code of `warning`, enter the following command:

```
pdadmin sec_master> authzrule modify r2 failreason warning
```

See the *IBM Security Access Manager for Web: Command Reference*.

List authorization rules

You can list the authorization rules that the Web Portal Manager or **pdadmin** utility created.

Listing authorization rules with Web Portal Manager

You can list all authorization rules with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule → List AuthzRule** to display the Manage AuthzRules page.

Results

A list of names for authorization rules that were created in Security Access Manager are displayed as links. If you select an authorization rule link, the properties of that rule are displayed.

Listing authorization rules with pdadmin

You can list authorization rules in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **authzrule list** command.

   ```
   pdadmin sec_master> authzrule list
   ```

   See the *IBM Security Access Manager for Web: Command Reference*.

Cloning an authorization rule with Web Portal Manager

You can clone an authorization rule in the domain only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule → List AuthzRule**.
3. From the Manage AuthzRules page, select the authorization rule you want to clone.
4. From the AuthzRule Properties page, click **Clone**.
5. From the Clone AuthzRule page, type an **AuthzRule Name** For example, type `Test-AuthzRule`. The default value is the name of the original authorization rule with the prefix `Clone`. This field is required.
6. Optional: Type a **Description** of the authorization rule. For example, type `Clone of new authorization rule`. The default value is the description of the original authorization rule.
7. Click **Clone**. If successful, a link for this cloned authorization rule is created and a success message is displayed.

Importing authorization rules with Web Portal Manager

You can import authorization rules in the domain only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule → Import AuthzRule**.
3. From the Import AuthzRule page, complete one of the following steps:
   - In the **AuthzRule File Name** field, type the name of the authorization rule to import. For example, type `ruleImport.xml`.
   - Click **Browse** to select a file name.
4. The file that contains the authorization rule might be encrypted when it was exported. In the **Encryption String** text field, type the string that was used to encrypt the XML file.

5. Click **Import**.

Results

If successful, the imported rule is available when you list all the rules.

Exporting all authorization rules with Web Portal Manager

You can export all authorization rules in the domain only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule → Export All AuthzRules** to display the Export AuthzRule to File page.
3. Optional: In the **Encryption String** text field, type the string to use to encrypt the XML file. If not specified, the exported file is in plain text.
4. When an encryption string is provided, in the **Confirm Encryption String** text field, type the string again.
5. Click **Export** to display the File Download window.
6. Click **Save** to display the Save As window.
7. Click **Save** to create the file that contains the exported rule descriptions. The default file name is `ruleExport.xml`.

Results

If successful, the exported rule descriptions are available in the specified location.

Exporting a single authorization rule with Web Portal Manager

You can export a single authorization rule in the domain only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule → List AuthzRule**.
3. From the Manage AuthzRules page, select the authorization rule that you want to export.
4. From the AuthzRule Properties page, click **Export** to display the Export AuthzRule to File page.
5. Optional: In the **Encryption String** text field, type the string to use to encrypt the XML file. If not specified, the exported file is in plain text.
6. When an encryption string is provided, in the **Confirm Encryption String** text field, type the string again.
7. Click **Export** to display the File Download window.
8. Click **Save** to display the Save As window.
9. Click **Save** to create the file that contains the exported authorization rule description. The default file name is `ruleExport.xml`.

Results

If successful, the exported rule description is available in the specified location.

Exporting multiple authorization rules with Web Portal Manager

You can export multiple authorization rules in the domain only with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule → List AuthzRule**.
3. From the Manage AuthzRules page, select the authorization rule that you want to export.
4. Click **Export** to display the Export AuthzRule to File page.
5. Optional: In the **Encryption String** text field, type the string to use to encrypt the XML file. If not specified, the exported file is in plain text.
6. When an encryption string is provided, in the **Confirm Encryption String** text field, type the string again.
7. Click **Export** to display the File Download window.
8. Click **Save** to display the Save As window.
9. Click **Save** to create the file that contains the exported authorization rule descriptions. The default file name is `ruleExport.xml`.

Results

If successful, the new XML file is available in the specified location.

Attach an authorization rule to a protected object

You can attach an authorization rule to a protected object with the Web Portal Manager or **pdadmin** utility.

Attaching an authorization rule to a protected object with Web Portal Manager

You can attach an authorization rule to a protected object with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule → List AuthzRule** to display the Manage AuthzRules page.

   A list of authorization rules that were created in Security Access Manager are displayed. Each rule is a link that displays properties for that rule when selected.
3. Click the link for the authorization rule that you want to attach to an object. For example, the `r2` authorization rule. The AuthzRule Properties page is displayed.
4. Click the **Attach** tab to view a list of protected objects to which the authorization rule is already attached, if any.
5. Click **Attach** to display the Attach AuthzRule page.
6. Type the **Protected Object Path** of the protected object to which you want to attach the authorization rule. This field is required. Be sure to type the full path name. For example, type the following path:

   ```
   /WebSEAL/tivoli.com/w3junction/index.html
   ```

7. Click **Attach**. If successful, the new protected object is added as a link to the list of objects to which the authorization rule is attached on the AuthzRule Properties–Attach page.

Attaching an authorization rule to a protected object with pdadmin

You can attach an authorization rule to a protected object with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **authzrule attach** command.

Example

For example, to attach a rule named r2 to a protected object named /WebSEAL/tivoli.com/w3junction/index.html, enter the following command:

```
pdadmin sec_master> authzrule attach /WebSEAL/tivoli.com/w3junction/index.html r2
```

See the *IBM Security Access Manager for Web: Command Reference*.

Detach an authorization rule

You can detach an authorization rule from a protected object with the Web Portal Manager or **pdadmin** utility.

Detaching an authorization rule with Web Portal Manager

You can detach an authorization rule from a protected object with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule → List AuthzRule** to display the Manage AuthzRules page.

   A list of authorization rules that were created in Security Access Manager are displayed. Each rule is a link that displays properties for that rule when selected.
3. Click the link for the authorization rule that you want to detach from an object. The AuthzRule Properties page is displayed.
4. Click the **Attach** tab to view a list of protected objects to which the authorization rule is already attached, if any.
5. Select one or more check boxes for the protected objects from which you want to detach the authorization rule.
6. Click **Detach** to display the Detach AuthzRule from Object page where you are prompted to confirm or cancel the request.

Detaching an authorization rule with pdadmin

You can detach an authorization rule from a protected object in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **authzrule detach** command.

Example

For example, to detach a rule from a protected object named /WebSEAL/tivoli.com/w3junction/index.html, enter the following command:

```
pdadmin sec_master> authzrule detach /WebSEAL/tivoli.com/w3junction/index.html
```

For more information, see the *IBM Security Access Manager for Web: Command Reference*.

Locate where an authorization rule is attached

You can find the protected objects that have an authorization rule attached with the Web Portal Manager or **pdadmin** utility.

Locating where an authorization rule is attached with Web Portal Manager

You can locate all the protected objects that are attached to an authorization rule with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule** > **List AuthzRule**. A list of authorization names is displayed. Each authorization rule name is a link that you can click to display the AuthzRule Properties page.
3. Click the **Attach** tab.

Locating where an authorization rule is attached with pdadmin

You can locate all the protected objects to which an authorization rule is attached in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **authzrule find** command.

Example

For example, to find the protected objects attached to a rule named r2, enter the following command:

```
pdadmin sec_master> authzrule find r2
```

See the *IBM Security Access Manager for Web: Command Reference*.

Delete an authorization rule

You can delete an authorization rule with the Web Portal Manager or **pdadmin** utility.

Deleting an authorization rule with Web Portal Manager

You can delete an authorization rule with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **AuthzRule** > **List AuthzRule** to display the Manage AuthzRules page.

A list of authorization rules that were created in Security Access Manager are displayed. Each rule is a link that displays properties for that rule when selected.

3. Select one or more check boxes for the links that you want to delete. For example, you might select the check box for the authorization rule named r2.
4. Click **Delete** to display the Delete AuthzRules page where you are prompted to confirm or cancel the deletion.

Deleting an authorization rule with pdadmin

You can delete an authorization rule in the domain with the **pdadmin** utility.

Procedure

1. Log on to the domain as the domain administrator.
2. Use the **authzrule delete** command.

Example

For example, to delete a rule named r2, enter the following command:

```
pdadmin sec_master> authzrule delete r2
```

See the *IBM Security Access Manager for Web: Command Reference*.

Chapter 11. Manage users and groups

An initial domain administrator is created when a new domain is created.

The domain administrator has the necessary privileges to manage the domain. The domain administrator can create and configure users, groups, resources, and applications, and can delegate administration tasks within the domain as required.

A *user* represents any authenticated Security Access Manager identity. Typically, these authenticated identities represent network users or resource managers.

A *group* is a collection of one or more users. An administrator can use group ACL entries to assign the same permissions to multiple users. New users to the domain gain access to objects by becoming members of groups. Group membership eliminates the need to create new ACL entries for each new user. Groups can represent organizational divisions or departments within a domain. Groups are also useful in defining roles or functional associations.

An *account* refers to users and groups collectively.

A registry unique identifier (UID) specifies the location in the user registry where the new user is created. Similarly, a registry group unique identifier (GID) specifies the location in the user registry where the new group is created. For registry UIDs and GIDs, you must type the full path name for the new user or group. The path format depends on the type of registry that the product is using. The following list shows sample formats for different user registries:

LDAP `cn=IBM-Support,o=ibm,c=us`

Active Directory
`cn=IBM-Support,dc=Austin,dc=US`

The registry UID or registry GID provides extra security in the case where a user or group is deleted from the domain and then recreated with the same name. For example, even though a new user has the same name as the deleted user, Security Access Manager allocates a new registry UID to this user. Because the registry UID is new, any existing ACL entries that refer to the old user name do not grant any rights to the new user. Stale UIDs from deleted users and groups are silently removed by the policy server.

Manage users

You can do the following user tasks:

In the following sections, instructions are provided for using either Web Portal Manager or **pdadmin**, or both. For online help while usingWeb Portal Manager, click the question mark to open a separate help window for the current page.

Create a user

You can create a user with the Web Portal Manager or **pdadmin** utility.

When a user is created, the domain administrator assigns a *user name*, which is sometimes called a *principal name*. The user name must be unique within the domain, because it is used by Security Access Manager to identify this user. A registry user identifier, known as a *distinguished name* (DN), is also assigned to uniquely identify the user definition in the user registry. The format of the DN depends on the registry type that is being used. Also assigned are the *common name* (CN) and *surname* (SN) of the user that is being defined.

Note: When Active Directory Lightweight Directory Service (AD LDS) is used as the user registry, users must be created within the same AD LDS partition in which Security Access Manager was configured.

Creating a user with Web Portal Manager

You can create a user in a domain with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the domain as a domain administrator.
2. Click **User → Create User**.
3. In the **User Id** text field, type the user name (for example `maryj`).
4. Click **Group Membership** to search for groups in which the user can be a member.
5. In the **First Name** text field, type the name of the user (for example `Mary`).
6. In the **Last Name** text field, type the family or surname of the user (for example `Jones`).
7. In the **Password** text field, type the password. Passwords must adhere to the password policies that are set by the domain administrator.
8. In the **Confirm Password** text field, type the password again.
9. In the **Description** text field, type the description for the user (for example, `Member of Marketing Group`.
10. In the **Registry UID** text field, type the registry UID. The registry UID specifies the location in the user registry where the new user is created. For example: `cn=maryj,o=ibm,c=us,dc=mkt`. Lotus® Notes® users require the full path name for the user that is being created. For example: `Mary Jones/IBM/US`
11. Select the **Account Valid** check box to indicate that the new user can participate in the domain. If this option is not selected, the new user account is not valid and the user cannot log on.
12. Select the **GSO User** check box to indicate the use of global sign-on (single sign-on) for Security Access Manager.
13. Select the **Password Valid** check box to force a password change the next time the user logs in to the domain. If this option is not selected, the user is informed that the password expired.
14. Click **No Password Policy** to indicate that you do not want the initial password to conform to the password policies that are set by the domain administrator.
15. Click **Create**.

Results

A message is shown if the user ID is created.

Creating a user with pdadmin

You can create a user with the **`pdadmin`** utility.

Procedure

1. Log on to the appropriate domain as a domain administrator.
2. Use the **user create** command to create the user.

Example

For example, to create the user named maryj with global sign-on capability, enter the following command:

```
pdadmin sec_master> user create –gsouser maryj "cn=Mary Jones,o=IBM,c=us,dc=mkt" \
"Mary Jones" Jones pwd2pwd2
```

The format of the distinguished name depends on the type of user registry. For more information, see the *IBM Security Access Manager for Web: Command Reference*.

List users

You can search for users with the Web Portal Manager or **pdadmin** utility.

However, when the user registry contains many user definitions, use wildcard characters with discretion. When a pattern includes one or more wildcard characters, the command attempts to find all user definitions that match the specified pattern. However, the command displays only the specified number of matching definitions in the user registry. If the user registry contains 10,000 definitions, specifying a single wildcard ("*"), with a limit of 100, finds all 10,000 definitions. However, the command displays only the first 100 matching definitions.

Note: If many users are defined in the user registry (for example, more than 100,000), avoid specifying the global wildcard (*). Instead, use a search filter that is as specific as possible, or qualify the search pattern to limit the search results.

For example, you might use the pdadmin tool and list users whose names start with John. Limit the search results by specifying the number of records to return, as in the following command: `pdadmin user list john* 100`

For the specific syntax of the user list command, see the *IBM Security Access Manager for Web: Command Reference*.

For more specific information about tuning the directory server to achieve best results, see the *IBM Security Access Manager for Web: Performance Tuning Guide*.

Listing users with Web Portal Manager

You can search for and list up to a maximum of 100 users with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log on to the appropriate domain as a domain administrator.
2. Click **User** > **Search Users** to display the User Search page.
3. At the User Search page, specify the pattern to filter user ID names. Use wildcard characters with discretion.
4. Use the default value of `100` or type another value in the **Maximum Results** field. This number limits the number of user IDs that are displayed.
5. Click **Search** to display a table of user IDs. Each user ID is displayed as a link.

From the User Search page, you can do these tasks: create a user, delete one or more existing users, and click the link to view user properties.

6. Use the default value of 15 user IDs per page, or click **Options** to type the number of user IDs to view per page. Toggle back by clicking **Hide Options**.
7. Use the default value of `None`, meaning that no text is used for filtering. Alternatively, click **Filters** to find user IDs that contain, start with, or end with the text that you specify. Toggle back by clicking **Hide Filters**.

Listing users with pdadmin

You can search for a list of users with the **pdadmin** utility.

Procedure

1. Log on to the domain as a domain administrator.
2. Use the **user list** command to list users.

Example

For example, to search for and list up to a maximum of 100 users, enter the following command:

```
pdadmin sec_master> user list * 100
```

For more information, see the *IBM Security Access Manager for Web: Command Reference*.

Change a password

You can change a user password with the Web Portal Manager or **pdadmin** utility.

The new password must comply with the password policies that are currently in effect.

Note: You might use Active Directory as your user registry with the Active Directory server on Windows 2008 SP1 or later. In this case, old passwords might still be able to be used after a password change.

See the following web page:

> http://support.microsoft.com/?id=906305

When setting or changing a password, the password must comply with the following policies:

- The defined Security Access Manager password policy
- The password policy for the underlying operating system
- The password policy for the underlying user registry

When enforcing the password policy, Security Access Manager validates compliance in the following sequence:

1. Against the Security Access Manager password policy currently in effect
2. Against the underlying user registry

Although a password complies to the defined Security Access Manager password policy, validation might fail against the password policy of the underlying operating system or user registry.

For additional information about setting the password policy for Security Access Manager users, see “Setting global user policy” on page 151.

Changing a password with Web Portal Manager

You can change the password for the specified user ID with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log in to the domain as a domain administrator.
2. Click **User → Change My Password**.
3. Verify that the **User ID** identifies the login identifier for the user whose password you want to change.
4. In the **Current Password** text field, type the existing password for the specified user ID.
5. In the **New Password** text field, type the new password for the specified user ID.
6. In the **Confirm New Password** text field, type the password again.
7. Click **Apply**.

Changing a password with pdadmin

You can change the password for the user with the **pdadmin** utility.

Procedure

1. Log in to the domain as a domain administrator.
2. Use the **user modify** command with the **password** option.

Example

For example, to change the password for the user dlucas to newpasswd, enter the following command:

```
pdadmin sec_master> user modify dlucas password newpasswd
```

For more information, see the *IBM Security Access Manager for Web: Command Reference*.

Setting user policy

You can change the user policy settings for specific users, such as password policies, login-failure policies, access policies, and account expiration policies with the Web Portal Manager or **pdadmin** utility.

Note:

- The valid range for numbers can be any number. However, use a reasonable number for the task that you want to do. For example, a minimum password length must be long enough to protect your system. The minimum length must not be so short as to make it easy for someone to determine your password by trying different combinations.
- When defining the password policy, ensure that this definition complies with the password policy of the underlying operating systems and user registries.
- When using Security Directory Server as your user registry, you can take advantage of its password history policy. For additional information about setting the password history policy when using Security Directory Server as your user registry, see “Setting the password history policy” on page 324.

- When modifying 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 greater than the specified end time, then the access is allowed until the specified end time of the next day.

Setting user policy with Web Portal Manager

You can change policy settings for a specific user with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log in to the domain as a domain administrator.
2. Click **User** > **Search Users** to display the User Search page.
3. From the list of matching users, select the user whose policy is to be changed. The User Properties page for that user is displayed.
4. Click the **Policy** tab.
5. Modify the following policies as needed:
   - For **Max Login Failures**, select **Unset** or **Set** to set or unset the maximum number of login failures before the account is no longer allowed to participate in the secure domain. If you select **Set**, either accept the default value of 10 or change the value to a number equal to or greater than zero.
   - For **Disable Time Interval**, select **Unset**, **Disable**, or **Set** to set the time, in seconds, or to disable each user account when the maximum number of login failures is exceeded. If you select **Set**, either accept the default value of 180 seconds or change the value to a number equal to or greater than zero.
   - For **Minimum Password Length**, select **Unset** or **Set** to set the minimum number of characters required for the password. If you select **Set**, either accept the default value of eight alphanumeric characters or change the value to a number greater than zero.
   - For **Maximum Password Age**, select **Unset** or **Set** to set the maximum time that a password can be used before it expires. The maximum password age is relative to the last time the password was changed. If you select **Set**, either accept the default value of 91 days (`91-00:00:00`) or change the value to a number equal to or greater than zero. A value of 0 (`000-00:00:00`) indicates that the password never expires.
   - For **Minimum Password Alphas**, select **Unset** or **Set** to set the minimum number of alphabetical characters required in a password. If you select **Set**, either accept the default value of four alphabetical characters or change the value to a number greater than zero.
   - For **Minimum Password Non-Alphas**, select **Unset** or **Set** to set the minimum number of non-alphabetic characters required in a password. If you select **Set**, either accept the default value of one non-alphabetic character or change the value to a number greater than one.
   - For **Max Password Repeated Characters**, select **Unset** or **Set** to set the maximum number of repeated characters allowed in a password. If you select **Set**, either accept the default value of two repeated characters or change the value to a number greater than two.
   - For **Password Spaces Allowed**, select **Unset**, **Yes**, or **No** to determine whether spaces are allowed in passwords. You can accept the default setting of **Unset**. You can change the value to **Yes** to allow spaces in passwords or to **No** to not allow spaces in passwords.
   - For **Max Concurrent Web Sessions**, select **Displace**, **Unset**, **Unlimited**, or **Set** to set the maximum number of concurrent web sessions allowed. If you select **Set**, type a number equal to or greater than one.

Note: This policy applies only to certain components. A *web session* is a user session that is maintained by the web security solutions, such as WebSEAL and plug-ins for web servers. Refer to the component administration guides to see whether this setting is applicable and whether specific configuration options are required to enforce this policy.

- For **Account Expiration Date**, select **Unset**, **Unlimited**, or **Set** to set the account expiration date. You can accept the default setting of **Unset**. You can change it to **Unlimited** or **Set**.

  If you select **Set**, type the four-digit year in the **Year** field.

  Either accept the default value of `Jan 01-00:00:00` or change the value to the date and time, specified as `Month DD:hh:mm:ss`. The hours must be entered using a 24-hour clock (for example, `09` for 9:00 a.m. or `14` for 2:00 p.m.).
- For **Time of Day Access**, select **Unset** or **Set** to set the time of day access policy. If you select **Set**, either accept the default settings or change them.

  You can change these values:
  - Select the days of the week from the choices provided.
  - Select **All Day** or **Between hours of**.

    If you select **Between hours of**, also select the **Start Time**. The start time format is specified as hours and minutes. The start time is expressed by using a 24-hour clock.

    If you select **Between hours of**, also select the **End Time**. The end time format is specified as hours and minutes. The end time is expressed by using a 24-hour clock.

    If you select **Between hours of**, also select **Local Time** or **UTC Time**. The time zone is local by default; UTC is Coordinated Universal Time.

6. Click **Apply**.

Setting user policy with pdadmin

You can set or change user policy settings with the **`pdadmin`** utility.

Procedure

1. Log in to the domain as a domain administrator.
2. Use the **`policy set`** command.

Example

For example, to set the maximum password age of 31 days 8 hours and 30 minutes for user `bsmith`, enter the following command:

```
pdadmin sec_master> policy set max-password-age 031-08:30:00 -user bsmith
```

For more information, see the *IBM Security Access Manager for Web: Command Reference*.

Setting global user policy

You can change global user settings, such as password policies, login-failure policies, access policies, and account expiration policies with the Web Portal Manager or **`pdadmin`** utility.

Notes:

- The valid range for numbers can be any number. However, use a reasonable number for the task that you want to do. For example, a minimum password length must be long enough to protect your system. The minimum length must not be so short as to make it easy for someone to determine passwords by trying different combinations.

- When you define the password policy, ensure that this definition complies with the password policy of the underlying operating systems and user registries.
- When you use Security Directory Server as your user registry, you can take advantage of its password history policy. For more information about setting the password history policy when you use Security Directory Server as your user registry, see “Setting the password history policy” on page 324.
- When you modify a password policy, 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 greater than the specified end time, then the access is allowed until the specified end time of the next day.
- Certain global user policies can be applied to basic users. See the following table for which policies are available to basic users.

Table 21. Global user policies for basic users

Policy	Available to basic users
`account-expiry-date`	Yes
`disable-time-interval`	No
`max-concurrent-web-sessions`	Yes
`max-login-failures`	No
`max-password-age`	No
`max-password-repeated-chars`	Yes
`min-password-alphas`	Yes
`min-password-length`	Yes
`min-password-non-alphas`	Yes
`password-spaces`	Yes
`tod-access`	Yes

Setting global user policy with Web Portal Manager

You can change global user settings with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log in to the domain as a domain administrator.
2. Click **User → Show Global User Policy**.
3. For **Max Login Failures**, select **Unset** or **Set** to set or clear the maximum number of login failures before the account is no longer allowed to participate in the secure domain. If you select **Set**, either accept the default value of 10 or change the value to a number equal to or greater than zero.
4. For **Disable Time Interval**, select **Unset**, **Disable**, or **Set** to set the time, in seconds, or to disable each user account when the maximum number of login failures is exceeded. If you select **Set**, either accept the default value of 180 seconds or change the value to a number equal to or greater than zero.
5. For **Minimum Password Length**, select **Unset** or **Set** to set the minimum number of characters required for the password. If you select **Set**, either accept the default value of eight alphanumeric characters or change the value to a number greater than zero.

6. For **Maximum Password Age**, select **Unset** or **Set** to set the maximum time that a password can be used before it expires. The maximum password age is relative to the last time the password was changed. If you select **Set**, either accept the default value of 91 days (`91-00:00:00`) or change the value to a number greater than zero.
7. For **Minimum Password Alphas**, select **Unset** or **Set** to set the minimum number of alphabetical characters required in a password. If you select **Set**, either accept the default value of four alphabetical characters or change the value to a number greater than zero.
8. For **Minimum Password Non-Alphas**, select **Unset** or **Set** to set the minimum number of non-alphabetic characters required in a password. If you select **Set**, either accept the default value of one non-alphabetic character or change the value to a number greater than one.
9. For **Max Password Repeated Characters**, select **Unset** or **Set** to set the maximum number of repeated characters allowed in a password. If you select **Set**, either accept the default value of two repeated characters or change the value to a number greater than two.
10. For **Password Spaces Allowed**, select **Unset**, **Yes**, or **No** to determine whether spaces are allowed in passwords. You can accept the default setting of **Unset**. You can change the value to **Yes** to allow spaces in passwords or to **No** to not allow spaces in passwords.
11. For **Max Concurrent Web Sessions**, select **Displace**, **Unset**, **Unlimited**, or **Set** to set the maximum number of concurrent web sessions to allow. If you select **Set**, type a number equal to or greater than one.

    Note: This policy applies only to certain components. A *web session* is a user session that is maintained by the web security solutions, such as WebSEAL and plug-ins for web servers. Refer to the component administration guides to see whether this setting is applicable and whether specific configuration options are required to enforce this policy.
12. For **Account Expiration Date**, select **Unset**, **Unlimited**, or **Set** to set the account expiration date. You can accept the default setting of **Unset**. You can change it to **Unlimited** or **Set**.

    If you select **Set**, type the four-digit year in the **Year** field.

    Either accept the default value of `Jan 01-00:00:00` or change the value to the date and time, specified as `Month DD:hh:mm:ss`. The hours must be entered using a 24-hour clock (for example, `09` for 9:00 a.m. or `14` for 2:00 p.m.).
13. For **Time of Day Access**, select **Unset** or **Set** to set the time of day access policy. If you select **Set**, either accept the default settings or change them.

    You can change these values:
    - Select the days of the week from the choices provided.
    - Select **All Day** or **Between hours of**.

      If you select **Between hours of**, also select the **Start Time**. The start time format is specified as hours and minutes. The start time is expressed by using a 24-hour clock.

      If you select **Between hours of**, also select the **End Time**. The end time format is specified as hours and minutes. The end time is expressed by using a 24-hour clock.

      If you select **Between hours of**, also select **Local Time** or **UTC Time**. The time zone is local by default; UTC is Coordinated Universal Time.
14. Click **Apply**.

Setting global user policy with pdadmin

You can set or change global user settings with the **pdadmin** utility.

Procedure

1. Log in to the domain as a domain administrator.
2. Use the **policy set** command.

Example

For example, to set a global user policy to a maximum password age of 31 days 8 hours and 30 minutes, enter the following command:

```
pdadmin sec_master> policy set max-password-age 031-08:30:00
```

For more information, see the *IBM Security Access Manager for Web: Command Reference*.

Import users

You can import a user that exists in a user registry and make that user a Security Access Manager user with the Web Portal Manager or **pdadmin** utility.

When a user is imported, the domain administrator assigns a user name, which is sometimes called a *principal name*. The user name must be unique within the domain because it is used by Security Access Manager to identify this user.

Note: When AD LDS is used as the user registry, you can import only existing users defined within the same AD LDS partition in which Security Access Manager was configured.

Importing users with Web Portal Manager

You can import a user that exists in a user registry and make that user a Security Access Manager user.

Procedure

1. Use Web Portal Manager to log in to the domain as a domain administrator.
2. Click **User → Import User**.
3. Type a **User Id** (for example `maryj`).
4. Click **Group Membership** to search for groups in which the user can be a member.
5. Type a **Registry UID**. The registry UID specifies the location in the user registry to be imported. For example: `cn=maryj,o=ibm,c=us,dc=mkt`. Lotus Notes users require the full path name for the user that is being imported. For example: `Mary Jones/IBM/US`
6. Select the **Account Valid** check box to indicate that the new user can participate in the domain. If this option is not selected, the new user account is not valid and the user cannot log in.
7. Select the **GSO User** check box to indicate that the user can use the global sign-on (single sign-on) for Security Access Manager.
8. Select the **Password Valid** check box to force a password change the next time the user logs in to the domain. If this option is not selected, the user is informed that the password expired.
9. Click **Create**.

Results

A message occurs if the user ID is created.

Importing groups with pdadmin

You can import a user that exists in a user registry and make that user a Security Access Manager user with the **pdadmin** utility.

Procedure

1. Log in to the domain as a domain administrator.
2. Use the **user import** command to import an existing user.

Example

For example, to import the user information for the user named maryj from the existing user registry definition, enter the following command:

```
pdadmin sec_master> user import –gsouser maryj "cn=Mary Jones,o=IBM,c=us,dc=mkt"
```

Note: When using an LDAP user registry and if necessary, the user information that is imported to the domain can be imported again to another domain.

For more information, see the *IBM Security Access Manager for Web: Command Reference*.

Delete a user

You can delete a user with the Web Portal Manager or **pdadmin** utility.

When you delete a user, this user is removed from all objects with which it is associated. If this user is the only ACL entry that is associated with an ACL policy, no other user or group can manage this ACL policy. Before deleting a user, you must validate that there are other users or groups that can manage this ACL policy.

Deleting a user with Web Portal Manager

You can delete a user from a domain with the Web Portal Manager.

Procedure

1. Use Web Portal Manager to log in to the domain as a domain administrator.
2. Click **User → Search Users**.
3. Search for one or more user names to delete and click **Search**.
4. Select the check boxes next to the user names to delete and then click **Delete**.
5. From the Delete Selected Users page, click **Delete Users** to confirm the deletion or click **Delete Users and Registry Entries** to also remove the registry entries associated with the selected users.

Deleting a user with pdadmin

You can delete a user from the domain with the **pdadmin** utility.

Procedure

1. Log in to the domain as a domain administrator.
2. Use the **user delete** command to delete a user. Any resource credentials associated with a user account are automatically removed when the user account is deleted. If the user does not exist in the user registry, an error is displayed.

Example

For example, to delete the user named jdoe and the associated information from the user registry, enter the following command:

```
pdadmin sec_master> user delete –registry jdoe
```

For more information, see the *IBM Security Access Manager for Web: Command Reference*.

Manage groups

You can do the following group tasks:

In the following sections, instructions are provided for using either Web Portal Manager or **pdadmin**, or both. For online help while usingWeb Portal Manager, click the question mark to open a separate help window for the current page.

Create a group

You can create a group with the Web Portal Manager or **pdadmin** utility.

When a group is created, the domain administrator assigns a group name. The group name must be unique within the domain because it is used by Security Access Manager to identify this group.

Note: When Active Directory Lightweight Directory Service (AD LDS) is used as the user registry, groups must be created within the same AD LDS partition in which Security Access Manager was configured.

Creating a group with Web Portal Manager

You can create a group in the domain with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log in to the domain as a domain administrator.
2. Click **Group → Create Group**.
3. Type a **Group Name** for the group (for example, sales).
4. Optional: Type a **Description** for the group (for example, Sales).
5. Type a **Registry GID**. The registry GID specifies the location in the user registry where the new group is created. For example: cn=Sales,o=ibm,c=us,dc=mkt. Lotus Notes users require the full path name for the group that is being created. For example: Sales/IBM/US.
6. Optional: Type the path in the **Object Container** field to the Security Access Manager object space where the group is to be created.
7. Click **Create**.

Results

The new group is displayed as a link. Select the link and the properties for the new group are displayed.

Creating a group with pdadmin

You can create a group in the domain with the **pdadmin** utility.

Procedure

1. Log in to the domain as a domain administrator.
2. Use the **group create** command to create a group and optionally place this group in a group container object. If the container object does not currently exist, it is automatically created.

Results

For example, to create the group named sales, enter the following command:

```
pdadmin sec_master> group create sales "cn=sales,o=IBM,c=us,dc=mkt" Sales
```

For more information, see the *IBM Security Access Manager for Web: Command Reference*.

List groups

You can search for group names with the Web Portal Manager or **pdadmin** utility.

Listing groups with Web Portal Manager

You can search for and list up to a maximum of 100 groups with Web Portal Manager.

Procedure

1. Use Web Portal Manager to log in to the domain as a domain administrator.
2. Click **Search Groups**.
3. At the Group Search page, use the special character (*) to filter group names.
4. Use the default value of 100 or type a **Maximum Results** number to limit the number of group names that you want to view.
5. Click **Search** to display a table of group names. Each group name is displayed as a link.

   From the Group Search page, you can do these tasks: create a group, delete one or more existing groups, and click the link to view group properties.
6. Use the default value of 15 group names per page or click **Options** to enter the number of group names you want to view per page. Toggle back by clicking **Hide Options**.
7. Use the default value of None, meaning that no text is used for filtering. Alternatively, click **Filters** to find group names that contain, start with, or end with the text that you specify. Toggle back by clicking **Hide Filters**.

Listing groups with pdadmin

You can search for a list of groups with the **pdadmin** utility.

Procedure

1. Log in to the domain as a domain administrator.
2. Use the **group list** command to list groups.

Example

For example, to search for and list up to a maximum of 100 groups, enter the following command:

```
pdadmin sec_master> group list * 100
```

For more information, see the *IBM Security Access Manager for Web: Command Reference*.

Import groups

You can import an existing group from a user registry into the domain and make that group a Security Access Manager group with the Web Portal Manager or **pdadmin** utility.

When a group is imported, the domain administrator assigns a group name. The group name must be unique within the domain because it is used by Security Access Manager to identify this group.

Note: When AD LDS is used as the user registry, you can import only existing groups defined within the same AD LDS partition in which Security Access Manager was configured.
Attention: If you import a dynamic group, ensure that the policy server is enabled for dynamic group support. For blade systems to benefit from dynamic group support, also enable this stanza entry on each blade system. For details about enabling the policy server for dynamic groups, see "Enabling dynamic group support" on page 159.

Importing groups with Web Portal Manager

To import an existing group from a user registry into the domain and make that group a Security Access Manager group, complete the following steps:

Procedure

1. Use Web Portal Manager to log in to the domain as a domain administrator.
2. Click **Group** > **Import Group**.
3. Type a **Group Name** for the group. For example, type `sales`.
4. Type a **Registry GID**. The registry GID specifies the location in the user registry of the group to be imported. For example, type `cn=sales,o=ibm,c=us,dc=mkt`. Lotus Notes users require the full path name for the group that is being imported. For example: `sales/IBM/US`.
5. Optional: Type the path in the **Object Container** field to the Security Access Manager object space where the group is to be imported.
6. Click **Import**.

Results

The new group is displayed as a link. Select the link to display the properties for the new group.

Importing groups with pdadmin

You can import an existing group from a user registry into the domain and make that group a Security Access Manager group with the **pdadmin** utility.

Procedure

1. Log in to the domain as a domain administrator.
2. Use the **group import** command to import an existing group.
3. Optional: Place this group in a group container object. If the container object does not currently exist, it is automatically created.

Example

For example, to import the existing group named "cn=sales,o=IBM,c=us,dc=mkt" from the user registry, enter the following command:

```
pdadmin sec_master> group import sales "cn=sales,o=IBM,c=us,dc=mkt"
```

Note: The group information that is imported to the domain can be imported again to another domain, if necessary.

See the *IBM Security Access Manager for Web: Command Reference*.

Delete a group

You can delete a group with the Web Portal Manager or **pdadmin** utility.

When you delete a group, this group is removed from all objects with which it is associated. If this group is the only ACL entry that is associated with an ACL policy, no other user or group can manage this ACL policy. Before deleting a group in this case, you must validate that there are other users or groups that can manage this ACL policy.

Deleting a group with Web Portal Manager

You can delete a group from the domain with the Web Portal Manager.

Procedure

1. Use Web Portal Manager to log in to the domain as a domain administrator.
2. Click **Group** > **Search Groups**.
3. Search for one or more group names to delete and click **Search**.
4. Select check boxes next to the group names to delete, and click **Delete**.
5. From the Delete Selected Groups page, click **Delete Groups** to confirm the deletion or click **Delete Groups and Registry Entries** to also remove the registry entries associated with the selected groups.

Deleting a group with pdadmin

You can delete a group from the domain with the **pdadmin** utility.

Procedure

1. Log in to the domain as a domain administrator.
2. Use the **group delete** command to delete a group.

Example

For example, to delete the group named sales and the associated information from the user registry, enter the following command:

```
pdadmin sec_master> group delete –registry sales
```

For more information, see the *IBM Security Access Manager for Web: Command Reference*.

Enabling dynamic group support

You can enable dynamic group support on the policy server and all servers where dynamic groups are supported. Use the **pdadmin** utility. The command that you run depends on the type of user registry.

LDAP registry

For LDAP registry users, modify the `dynamic-groups-enabled` entry in the `[ldap]` stanza of the `ldap.conf` file.

To do so, edit the configuration files with the local management interface as follows:

```
dynamic-groups-enabled = yes
```

For configuration changes to take effect, you must restart the updated server.

Note: Dynamic groups are not supported for the AD LDS registry.

Related tasks:

Managing runtime configuration files
To manage configuration files with the local management interface, use the Runtime Component management page.

Chapter 12. Certificate and password management

To securely transfer information between servers and clients, you can configure Security Access Manager to use various server-side and client-side certificates, key files, and stash files for authentication. During the initial configuration, you can configure the settings for the default lifetime of the certificates and the key file passwords.

This information describes certificate and password management from the perspective of the administration C API run time. However, Security Access Manager also provides a Java run time to complete the same tasks. For more information about the administration Java runtime and classes, see the *Administration Java Classes Developer Reference* and the *Authorization Java Classes Developer Reference*.

Security Access Manager can use Secure Sockets Layer (SSL) for encryption, system authentication, and application-level authentication. When installed and configured, SSL uses certificates for operation that help to ensure a secure environment. Security Access Manager can also use Transport Layer Security (TLS) version 1 and SSL. TSL can be enabled to work when compliance modes are enabled and when compliance modes are disabled.

In the secure environment, the policy server acts as the certificate authority (CA) and is responsible for creating and renewing certificates. The IBM Security Access Manager runtime package (`pdrte`) relies on only SSL server-side authentication and does not require a client-side certificate. All of the Security Access Manager servers, including the policy server, authorization server, policy proxy server, and resource manager server, rely on client-side certificates to operate.

The Security Access Manager servers use certificates to authenticate themselves. For example, when the authorization server wants to communicate with the policy server, it presents its client-side certificate. In this example, the policy server is considered the server, and the authorization server is the client. The policy server verifies that the certificate is valid and is signed by a trusted signer. In this case, the trusted signer is the policy server itself, using the Security Access Manager certificate authority (PDCA) certificate. The authorization server does the same for the certificate presented by the policy server. During the Security Access Manager application-level authentication, the policy server:

- Determines whether the authorization server certificate is good.
- Tries to map the certificate to a Security Access Manager user.

If the authentication succeeds, then the servers can communicate.

The certificates used by Security Access Manager are in *key files*. Key files have a `.kdb` extension (or `.ks` extension for Java *keystores*). Key files must be secured and protected by the strictest operating system controls available, because they contain the private keys for the certificates. For example, the key file for the policy server is `ivmgrd.kdb` and, by default, it can be read and written to by only the **ivmgr** user.

The certificate files in a directory need to be accessible to the user **ivmgr** (or all users). Ensure that the `ivmgrd.kdb` file and the directory or folder that contains the `ivmgrd.kdb` file is accessible by the user **ivmgr** or all users. Ensure that these users have the appropriate permissions for this file.

Furthermore, to facilitate unattended server operation, there are files that contain an obfuscated (not encrypted) version of the password to the key files. These versions are *stash files*, which are denoted by a `.sth` file extension. Java key files that are generated by Security Access Manager do not have corresponding stash files. These stash files must be secured by using the local standard operating system measures. For the policy server, the stash file is `ivmgrd.sth` and its permissions are the same as the `ivmgrd.kdb` key file.

For security reasons, both the certificates and the key file passwords can be set to expire after a configurable amount of time. The default lifetime for a certificate is four years. The default lifetime for a key file password is 183 days. The fixed lifetime for the PDCA certificate is 20 years. By default, the Security Access Manager servers refresh the certificates and passwords automatically while they are running. The refresh process reissues a new certificate with a new lifetime and generates a new password with the configured lifetime.

The Security Access Manager calculates the life span of the certificate when you open the security context. When opening the security context, the Security Access Manager verifies the need to refresh the context. If there is a need to refresh the certificate, then Security Access Manager creates an SSL context with the new certificate and processes the request.

If the servers are not running in a specified time frame, then their certificates or passwords can expire. In this case, a manual refresh is necessary. Also, if a certificate or a password or the entire key file is damaged, then you must manually refresh it. Refreshing the expired or damaged certificate, password, or key file is necessary to maintain the security of the Security Access Manager domain. For information about manually refreshing the certificates, passwords, and key files, see "Key file and stash file renewal information" on page 163.

Initial configuration

You create certificates used by the Security Access Manager servers during the initial configuration of the servers. The Security Access Manager servers use these certificates to securely communicate with other servers.

In a new Security Access Manager installation, the policy server is the first server that is configured. During the configuration, two certificates are created: the PDCA certificate and a personal certificate that is used by the policy server and signed by the PDCA certificate. Both of these certificates are in the `ivmgrd.kdb` key file. During the policy server configuration, the IBM Security Access Manager runtime key file `pd.kdb` is created. The PDCA certificate is inserted into it as a trusted certificate.

When new systems are added to the Security Access Manager domain, the IBM Security Access Manager runtime package is configured first. As part of this configuration, the system `pd.kdb` and `pd.sth` files are created. The PDCA certificate is included in the key files as a trusted certificate.

When new resource managers, such as WebSEAL, are configured, the **`svrsslcfg`** utility or an equivalent application programming interface (API) is run. This utility creates a key file (such as `pdacld.kdb`) and places a personal certificate for the server in it. The utility also inserts the PDCA certificate as a trusted certificate in the key file. These two certificates are obtained from the policy server. The certificates are transported to the client system over SSL with the IBM Security Access Manager runtime key file.

For more information about the configuration files and certificate-related stanza entries, such as the configured key file and the configured stash files, see Appendix B, “Configuration file reference,” on page 189.

Key file and stash file renewal information

Servers have associated key files and stash files.

The following table describes the server key and stash files, including how they are created and refreshed.

Table 22. Server key and stash files

Server	Key and stash files	How created	How automatically updated	How manually updated
IBM Security Access Manager runtime package	`pd.kdb` and `pd.sth` (does not contain a client-side certificate)	During runtime configuration	Running the **`pdadmin`**[1] utility	Running the **`bassslcfg`** utility with the **–chgpwd** option
Policy server	`ivmgrd.kdb` and `ivmgrd.sth`	During server configuration	Running the **`pdmgrd`**[1,2] command	Running the **`mgrsslcfg`** utility with the **-chgpwd**[3] and **-chgcert**[3] options
Proxy server	`pdmgrproxyd.kdb` and `pdmgrproxyd.sth`	During server configuration	Running the **`pdmgrproxyd`**[1] command	Running the **`svrsslcfg`** utility with the **–chgpwd**[9] and the **–chgcert**[5] options

Authorization server	`[instance-]ivacld.kdb`
	`[instance-]ivacld.sth`
	Note: [instance-] is the instance of an authorization server on a computer. Having more than one authorization server on a computer generates multiple sets of .kdb and .sth filenames.
		During server configuration	Running the **`pdacld`**[1] command	Running the **`svrsslcfg`** utility with the **-chgpwd**[4] and **-chgcert**[5] options

Resource manager	The key files and stash file names are resource manager-dependent, and the file name is configurable.[6]	Running the **`svrsslcfg`** utility with the **–config** option	Running an instance of the resource manager[1]	Running the **`svrsslcfg`** utility with the **–chgpwd**[7] and **–chgcert**[8] options

Table notes:

[1] You can turn off automatic certificate and password refresh by setting the `ssl-auto-refresh` stanza entry to `no` in the `[ssl]` stanza in the respective configuration file.

[2] Because the policy server also acts as the certificate authority (CA) for the secure domain, it must be recycled after a refresh. It continues to operate normally until it is recycled, but it cannot issue or renew certificates for other servers until it is recycled. The policy server log file contains a message that states when to restart the server.

[3] Before running this command, stop the policy server.

[4] Before running this command, stop the authorization server.

[5] Before running this command, the policy server must be running. Stop the authorization server.

[6] Java resource managers have an equivalent to key files, known as Java keystores, where the application personal certificate and the PDCA certificate are stored. Java resource managers do not have a stash file equivalent. The names of keystores are created by running the Java `SvrSslCfg` class with the `–action config` option.

[7] Before running this command, the resource manager must be stopped.

[8] Before running this command, the policy server must be running, and the resource manager must be stopped.

[9] Before running this command, the proxy server must be stopped.

Regenerating certificates

If a private key in the PDCA certificate is compromised, then you must regenerate the key file. You might change Security Access Manager to a different compliance type that requires certificates with different bit strengths or signature algorithms. In this case, you must regenerate the key file.

About this task

Each key file contains a list of trusted certificate authorities (CAs). Each key file except `ivmgrd.kdb` has the Security Access Manager certificate authority (PDCA) certificate as a trusted certificate authority. This certificate authority signs all the other Security Access Manager certificates. This certificate authority is created during policy server configuration and is placed in the `ivmgrd.kdb` file.

It is important to protect the `ivmgrd.kdb` file to keep the private key in the PDCA certificate from being compromised. If the private key is compromised, the private key, each key file, and each certificate in the domain must be regenerated.

From the Java perspective, the IBM Security Access Manager Runtime for Java also stores the PDCA certificate. If the PDCA certificate is compromised and must be regenerated, you must reconfigure all servers that use the IBM Security Access Manager Runtime for Java.

You must also regenerate the key file for all resource managers that were previously configured with the `SvrSslCfg` class. Reconfigure these resource managers.

Procedure

1. Stop the policy server.
2. Regenerate the PDCA certificate and policy server certificate by generating a new `ivmgrd.kdb` file with the **`mgrsslcfg –config`** utility.
3. Regenerate the IBM Security Access Manager runtime certificates on the policy server by running the **`bassslcfg –config`** utility.
4. After obtaining the certificate authority certificate, you can choose to automatically download the certificate authority certificate or manually copy the file.
   - If auto-download is set to `on` (enabled) and the policy server is running, the certificate authority certificate is automatically obtained. By default, auto-download is enabled.

- If auto-download is set to `off` (disabled), the base-64 DER encoded version of the PDCA certificate must be copied to the system. This file is stored as `pdcacert.b64` on the policy server.

5. On each runtime system, run the **`bassslcfg –config`** utility.
6. On each authorization server in the domain, regenerate its key files by running the **`svrsslcfg –config`** utility. The policy server must be running. This command updates both the server certificate for the authorization server and its trusted certificate (the new PDCA certificate).
7. On each resource manager in the domain, regenerate its key files by running the **`svrsslcfg –config`** utility. The policy server must be running. This command updates both the server certificate for the authorization server and its trusted certificate, the new PDCA certificate.
8. On each Security Access Manager Java runtime system, run the **`pdjrtecfg -unconfig`** utility, the **`pdjrtecfg -config`** utility, and the **`java com.tivoli.pd.jcfg.SvrSslCfg -action replcert`** command.

Reconfiguring the PDCA on the policy server

If the certificate is compromised or expires, you must reconfigure the PDCA on the policy server.

Procedure

1. Stop all Security Access Manager services that are running on the system by entering the following command:
   - AIX®, Linux, and Solaris operating systems:
     ```
     pd_start stop
     ```
   - Windows operating systems:
     ```
     drive:\net stop servername
     ```
     Stop each Security Access Manager service. For example, to stop the policy server, type:
     ```
     C:\net stop IVMgr
     ```
2. Change to the directory where the key files are located. Assuming the default directory on a AIX, Linux, or Solaris operating system, enter the following command:
   ```
   cd /var/PolicyDirector/keytab
   ```
3. Rename the `ivmgrd.kdb` key file, `ivmgrd.sth` stash file, and `pdcacert.b64` PDCA file by entering the following commands:
   ```
   mv ivmgrd.kdb ivmgrd.kdb.old
   mv ivmgrd.sth ivmgrd.sth.old
   mv pdcacert.b64 pdcacert.b64.old
   ```
4. Configure the policy manager server to create a new key file and stash file. For example, enter the command but replace the value for the **compliance** option.
   ```
   /opt/PolicyDirector/sbin/mgrsslcfg -config -D yes -C compliance
   ```
5. Change the ownership of the newly created key file, stash file, and certificate to `ivmgr:ivmgr` by entering the following commands:
   ```
   chown ivmgr:ivmgr /var/PolicyDirector/keytab/ivmgrd.kdb
   chown ivmgr:ivmgr /var/PolicyDirector/keytab/ivmgrd.sth
   chown ivmgr:ivmgr /var/PolicyDirector/keytab/pdcacert.b64
   ```
6. Configure the IBM Security Access Manager runtime with the **`bassslcfg` –config** utility. For example, enter the command but replace the values for the **–c**, **–h**, and **–C** options.

```
bassslcfg -config -C {compliance} -h myhostname
-c /var/PolicyDirector/keytab/pdcacert.b64
```

7. Change the ownership of the new key file and stash file to `ivmgr:ivmgr` by entering the following commands:

   ```
   chown ivmgr:ivmgr /var/PolicyDirector/keytab/pd.kdb
   chown ivmgr:ivmgr /var/PolicyDirector/keytab/pd.sth
   ```

8. Start the Security Access Manager services on the computer by entering the following command:

   ```
   /opt/PolicyDirector/bin/pdmgrd
   ```

9. Update the certificates of the authorization, proxy, and resource servers and other C API applications that use `svrsslcfg -config` by entering the following command:

   ```
   svrsslcfg -chgcert
   ```

   This example shows the command (on one line) to update the certificate on the authorization server:

   ```
   svrsslcfg -chgcert -f /opt/PolicyDirector/etc/[instance-]ivacld.conf
   -P *** -A sec_master
   ```

10. Start the updated Security Access Manager servers by entering the following command:

    ```
    pd_start restart
    ```

11. Reconfigure the certificates of any other Security Access Manager Java applications on the policy server. See "Reconfiguring the certifications of Security Access Manager Java applications."

What to do next

After updating the PDCA on the policy server, you must update the certificates on all other systems that run Security Access Manager servers and applications.

The management environment must be running.

After regenerating the PDCA certificate on the policy server, you might need to copy the PDCA certificate to each runtime computer in the domain. If auto-download is enabled, you do not need to copy the file.

Reconfiguring the certifications of Security Access Manager Java applications

To use the new policy server certificate authority, you must reconfigure the PDCA in the configured Java run time. You must also reconfigure the certificates of any Security Access Manager Java application that uses the IBM Security Access Manager Runtime for Java. First, update the IBM Security Access Manager Runtime for Java configuration. Then, update the certificate of each Security Access Manager Java application that uses the run time.

Before you begin

Back up all the files in `[JRE]/PolicyDirector`. For WebSphere Application Server version 8.0 and later, the directory is `[WAS_HOME]/tivoli/tam/PolicyDirector`.

About this task

This procedure updates the IBM Security Access Manager Runtime for Java files. Then it updates the individual Security Access Manager Java components with the IBM Security Access Manager Runtime for Java.

The IBM Security Access Manager Runtime for Java files that must be updated are the `PDCA.ks` file and the `ssl-compliance` property in the `PD.properties` file.

There are several ways that you can reconfigure the certification of a Security Access Manager Java application:

- Unconfigure and then reconfigure the IBM Security Access Manager Runtime for Java.
- Obtain a `PDCA.ks` file from another IBM Security Access Manager Runtime for Java that was already updated. Then, copy the file into the target IBM Security Access Manager Runtime for Java.

  If you configured the Java application with the Security Access Manager, version 7.0, configuration program, you specified a location for the `PDCA.ks` file. Replace the `PDCA.ks` file at that location instead of the location in the JRE.

  1. To locate the `PDCA.ks` file, open the properties configuration file of your application for IBM Security Access Manager Runtime for Java. For example, the file might be named `pdwpm.properties`.
  2. In the file, find the `pdca-url` entry. The entry specifies the `PDCA.ks` file path.

     ```
     pdca-url=file\:/user_supplied_path/PDCA.ks
     ```
  3. Write the `PDCA.ks` file from an updated IBM Security Access Manager Runtime for Java into the location that the `pdca-url` entry specifies.
- Also update the `ssl-compliance` entry, if it exists. For example:

  ```
  ssl-compliance=none
  ```

  Change the value to the appropriate compliance level for Java application that you configured with Security Access Manager, version 7.0.

  For example:

  ```
  ssl-compliance=suite-b-192
  ```

Procedure

1. Update the `PDCA.ks` and `PD.properties` files by unconfiguring the Java runtime and then reconfiguring it.

   Note:

   - This step removes all files in the `[JRE]/PolicyDirector` directory and then re-creates the files. For WebSphere Application Server version 8.0 and later, the directory is `[WAS_HOME]/tivoli/tam/PolicyDirector`.
   - If any file under this directory was customized, then you must reapply the customization to the new file.
   - At this step, do not unconfigure the Security Access Manager Java applications that are configured to use the JRE.

   You might need more information about configuring or unconfiguring Security Access Manager run time for Java. See the **pdjrtecfg** command utility in the *IBM Security Access Manager for Web Command Reference*.
2. Update the WebSphere profile if:
   - The Security Access Manager compliance type changed and

- The Security Access Manager Java applications run in a WebSphere profile.

  The FIPS security mode must match the Security Access Manager compliance level.

3. Stop any processes that are using the JRE. For example, stop any WebSphere profiles that are using the JRE.
4. Update the `ssl.client.props` file of the WebSphere profile to allow WebSphere client applications to communicate with the profile if:
   - You are using a WebSphere Java run time and
   - You changed the FIPS security mode of the run time.

   o
5. Regenerate the certificates of each `SvrSslCfg` Security Access Manager Java application. This example illustrates how to reconfigure the Security Access Manager WebSphere Portal Manager certificates:

   ```
   java com.tivoli.pd.jcfg.SvrSslCfg -action replcert -admin_id sec_master
    -admin_pwd -cfg_file /opt/PolicyDirector/java/export/pdwpm/pdwpm.properties
   ```
6. Start the JRE and ensure that it operates properly in the updated Java run time. For WebSphere, start the WebSphere profile to start the JRE.

What to do next

Repeat this procedure for any other Security Access Manager Java run times that are on the system.

Reconfiguring the PDCA on the runtime systems

After you reconfigure the policy server and transfer the newly generated PDCA certificate to each runtime system, you must reconfigure the PDCA on the runtime systems.

Procedure

1. Stop all Security Access Manager services that are running on the system by entering the following command:
   - AIX, Linux, and Solaris operating systems:

     ```
     pd_start stop
     ```
   - Windows operating systems:

     drive:\net stop *servername*

     Stop each Security Access Manager service. For example, to stop the policy server, type:

     ```
     C:\net stop IVMgr
     ```
2. Configure the IBM Security Access Manager runtime with the **bassslcfg –config** utility. For example, enter the command but replace the values for the **–c** and **–h** options.

   ```
   /opt/PolicyDirector/sbin/bassslcfg -config -h
   policysvrhostname -c /var/PolicyDirector/keytab/pdcacert.b64
   ```
3. Run the **svrsslcfg -chgcert** command for the authorization, proxy, and resource servers and for any other C API applications that use **svrsslcfg -config**. This example is for the authorization server:

   svrsslcfg -chgcert -f /opt/PolicyDirector/etc/*[instance-]*ivacld.conf -P *** -A sec_master
4. Start the Security Access Manager services on the computer by entering the following command:

   ```
   pd_start start
   ```

5. Reconfigure the certificates of any other Security Access Manager Java applications on the policy server. See “Reconfiguring the certifications of Security Access Manager Java applications” on page 166.

What to do next

Reconfigure the certificates of any Security Access Manager Java applications. See “Server certificate revocation” on page 170.

Transferring the PDCA certificate to other systems

After regenerating the PDCA certificate, you can transfer the PDCA certificate to each system in the domain. In this case, your business security policy requires trusted transport of the PDCA signer certificate to the target machine. The network between the policy server and the target system contains untrusted segments.

About this task

If auto-download is disabled, then you must manually copy the file to each system. If the File Transfer Protocol (FTP) is supported in your environment, use one of the following FTP options:

- Use the **put** command from the policy server to transfer the certificate to the other system.
- Use the **get** command from the other system to retrieve the certificate from the policy server.

The following steps assume that the `pdcacert.b64` certificate is retrieved from the policy server:

Procedure

1. Change to the local directory on the policy server that contains the `pdcacert.b64` file:
   ```
   cd /var/PolicyDirector/keytab
   ```
2. Connect to the runtime system by opening an FTP session. To illustrate, `pdruntime1` is the name of the runtime system.
   ```
   ftp pdruntime1
   ```
3. Log on to the remote system with the appropriate user ID and password.
4. Change to the directory where you want to store the certificate. Assuming the default directory is on a AIX, Linux, or Solaris operating system, enter the following FTP command:
   ```
   cd /var/PolicyDirector/keytab
   ```
5. Indicate that the file to be transferred is a text (ASCII) file by entering the following command:
   ```
   ascii
   ```
6. To view the transfer process visually, enter the following command:
   ```
   hash
   ```
7. Start the transfer by entering the following command:
   ```
   put pdcacert.b64
   ```
8. After the transfer completes, end the FTP session by entering the following command:
   ```
   quit
   ```

Server certificate revocation

If a certificate on a resource manager is compromised, you can revoke the certificate and then replace it with a new certificate.

If the certificate on a C-based resource manager is compromised, you can run the **svrsslcfg –chgcert** utility to replace the existing server certificate and update the PDCA certificate.

For resource managers that are based on Java, use the `PDAppSvrConfig.replaceAppSvrCert()` method.

You also can reconfigure a C-based server by running the **svrsslcfg –unconfig** and **svrsslcfg –config** utilities. The policy server must be running when you reconfigure it. These commands update both the server certificate for the authorization server and its trusted certificate (the new PDCA certificate). Similarly, a resource manager based on Java can be unconfigured and reconfigured with the Java `SvrSslCfg` class.

Additional key file and stash file considerations

There are additional considerations for key file and stash file renewal.

- When a certificate and the password to the key file that contains that certificate are both expired, the password must be refreshed. For example, for the authorization server, run the **svrsslcfg –chgpwd** utility and then the **svrsslcfg –chgcert** utility. You must run these utilities because a valid password is needed to open the key file to obtain the certificate.
- The lifetime of a certificate is determined by the value of the `ssl-cert-life` entry in the `[ssl]` stanza of the `ivmgrd.conf` file when the policy server is started. Any certificates that are issued or renewed use this value. To increase or decrease this value, change the value and restart the policy server. The new value is in effect only for certificates that are issued or renewed from that point onward. The actual time is whichever value is less: the value specified in the `ivmgrd.conf` configuration file or the number of days before the policy server certificate authority certificate expires.
- For automatic password renewal, the lifetime of a password is controlled by the value of the `ssl-pwd-life` entry in the `[ssl]` stanza in effect when the server is started. For manual password renewal, the value is dictated by the value supplied to the **svrsslcfg –chgpwd** utility. This value is also written into the appropriate configuration file.
- The key file password refresh occurs after half the lifetime of the password expiration date. If the blade server is not running during the second half of the password life, the ACL update cannot refresh the password. The update uses the connection from the management server to the blade server, which uses the SSL connection protected by the certificate. The certificate, in turn, is protected by the password.
- Security Access Manager servers can also communicate with Lightweight Directory Access Protocol (LDAP) with SSL. In the standard configuration, this communication uses server-side authentication only. Therefore, the Security Access Manager server needs only the CA certificate that signed the LDAP server certificate or the LDAP server certificate itself. The expiration and management of these certificates are not handled by Security Access Manager. However, it is possible to include the LDAP certificate in the key file for a resource manager by running the **svrsslcfg –config** utility with the **-C** option.

Refresh certificates that are not managed by Security Access Manager with the same mechanism that created the initial certificate. The new certificate can be replaced in the key file by running the **svrsslcfg –modify –C** *new_cert_filename* utility.

- After running the **bassslcfg –config** utility, you might need to change the permissions on the pd.kdb and pd.sth files.
- The configuration files mentioned are found in the *install_dir*/etc directory. For example, on an AIX system, the policy server, authorization server, and runtime configuration files are /opt/PolicyDirector/etc/ivmgrd.conf, /opt/PolicyDirector/etc/*[instance-]*ivacld.conf, and /opt/PolicyDirector/etc/pd.conf. Similarly, the key files and stash files can be found in the *install_dir*/keytabs directory.
- Security Access Manager does not distinguish between export and domestic encryption. For encryption based on Java, the strength is regulated by the jurisdiction files that are present in the Java runtime environment. There is no set length for keys generated by the IBM Security Access Manager runtime.
- Both the public keys that are included in certificates and the private keys that might be stored in key files have key lengths. The maximum key length is 2048 bits. Public keys with 2048-bit key lengths can be generated by using the configuration utilities **bassslcfg**, **mgrsslcfg**, or **svrsslcfg**.

Chapter 13. Server management

This chapter provides detailed information about general administration and configuration tasks on the Security Access Manager servers.

Security Access Manager servers

Security Access Manager consists of server processes, or daemons.

The server processes (daemons) include:

pdmgrd
The server process for the policy server.

pdacld
The server process for the authorization server.

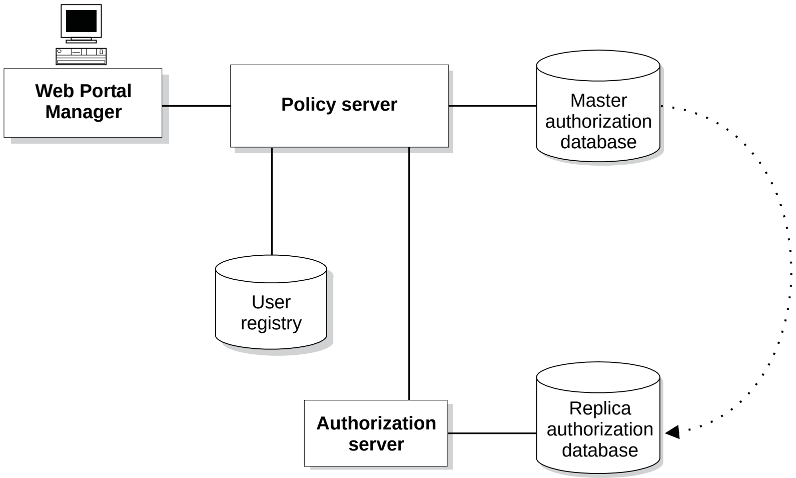


Figure 18. Security Access Manager server components

The authorization server allows other applications to make authorization calls to Security Access Manager with the authorization application programming interface (API). The authorization server also acts as a logging and auditing collection server to store records of server activity.

Server dependencies

To ensure optimal performance, you must consider several dependencies when you configure your server.

Dependencies include:

- There must be at least one instance of the policy server.
- There must be at least one policy server defined. You can have a single policy server and create as many domains as you want. When a domain is created, a separate policy database is also created for each domain. The single policy server can access any of the distinct policy databases.
- The policy server manages the policy database.

- There must be only one policy database (master authorization database) in a domain.
- The policy database must be on a highly available policy server with a robust file system.
- Each policy database is subject to a regular backup procedure. The administrator can specify the location for the backup files.
- The policy servers provide authorization database replication services to all other Security Access Manager servers in the domain that run in local cache mode.
- Each resource manager, such as Security Access Manager WebSEAL, applies security policy based on information from either the policy database or from a replicated authorization database.

Security Access Manager utilities

The Security Access Manager utilities are described in detail in the *IBM Security Access Manager for Web: Command Reference*.

The table at the beginning of the utilities section lists available utilities and their purposes.

The `pdadmin` utility, which is also described in *IBM Security Access Manager for Web: Command Reference*, provides commands that assist in troubleshooting problems. For example, the `pdadmin` utility includes the **server task stats** and **server task trace** commands that enable statistics gathering and capture information about error conditions. In addition, the *IBM Security Access Manager for Web Troubleshooting Guide* provides further diagnostic information for using the Security Access Manager `pdadmin` utility and other utilities.

Server configuration file tasks

You can use the server configuration files to customize the operation of Security Access Manager and its servers.

Various server configurations are discussed in Appendix B, "Configuration file reference," on page 189.

Changing configuration settings

You can change a configuration setting. For example, you might change Secure Sockets Layer (SSL) configuration settings for the Security Access Manager servers.

About this task

The configuration files, stanzas, and stanza entries are described in Appendix B, "Configuration file reference," on page 189.

Procedure

1. Make a backup copy of the configuration file that you plan to modify. If you encounter an error, use the backup copy to return the configuration file to a known working state.
2. Stop the Security Access Manager servers that are affected.
3. Use one of the following mechanisms to modify the configuration file:
   - Use the `pdadmin config` commands to modify the configuration file.

- Use the appropriate configuration tool for your server to change the configuration settings:
  - For the `ivmgrd.conf` file, use the **`mgrsslcfg`** utility.
  - For the `pd.conf` file, use the **`bassslcfg`** utility.
  - For all other configuration files, use the **`svrsslcfg`** utility.

  Note: Many stanzas or values are created or modified only by using Security Access Manager configuration utilities. Some values are completed automatically after the configuration is completed. Do not modify these values.
4. Start the Security Access Manager servers that are affected.

Example

For example, if you want to change the `ivmgrd.conf` file, you must stop the policy servers. Complete the change and restart all the policy servers to put the change into effect.

Automatic server startup at boot time

Stanza entries for automating server startup are in the `[pdrte]` stanza of the `pd.conf` configuration file.

By default, the `pd.conf` file is installed at the following location for AIX, Linux, and Solaris operating systems:

```
/opt/PolicyDirector/etc/pd.conf
```

By default, the `pd.conf` file is installed at the following location for Windows operating systems:

```
c:\Program files\tivoli\Policy Director\etc\pd.conf
```

Policy server

When the PDMgr package is installed, the policy server automatically starts after each system reboot.

```
[pdrte]
boot-start-ivmgrd = yes
```

To prevent the policy server from automatic startup, set:

```
boot-start-ivmgrd = no
```

Authorization server

When the PDAcld package is installed, the authorization server automatically starts after each system reboot.

```
[pdrte]
boot-start-[instance-]ivacld = yes
```

To prevent the authorization server from automatic startup, set:

```
boot-start-[instance-]ivacld = no
```

Policy server administration tasks

The policy server manages the policy database or databases and maintains location information about other Security Access Manager servers in each domain.

The policy server typically requires little administration or configuration. This section describes configuration tasks available to the administrator.

Replicate the authorization database

A Security Access Manager domain administrator can make security policy changes to a domain at any time.

A primary responsibility of the policy server is to make the necessary adjustments to the domain master authorization database to reflect these changes.

When the policy server modifies the master authorization database, it can send out notification of this change to all resource manager servers with replica databases. The authorization servers must then request a database update from the policy server.

Note: Additionally, resource manager servers can check for database updates by polling the policy server at regular intervals. Polling configuration for a WebSEAL client, for example, is explained in the *IBM Security Access Manager for Web: WebSEAL Administration Guide*.

Update notifications from the policy server can be configured as an automatic process or a manually controlled task. Notification is determined by the `auto-database-update-notify` stanza entry in the `[ivmgrd]` stanza of the `ivmgrd.conf` configuration file. By default, the stanza entry value is set to `yes` (update notification is automatically done by the policy server):

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

This automatic setting is appropriate for environments where database changes are few and infrequent. When you configure update notification to be automatic, you must also correctly configure the `max-notifier-threads` and `notifier-wait-time` stanza entries. For more information about these entries, see "Set the number of update-notifier threads" on page 177 and "Set the notification delay time" on page 177.

When you configure update notification to be manual, manual application of the **`server replicate`** command controls this event.

```
[ivmgrd]
auto-database-update-notify = no
```

This manual setting is appropriate for environments where database modifications occur frequently and involve substantial changes. In some cases, several database modifications can generate many update notifications that soon become obsolete because of the continuing changes to the master database. These obsolete notifications cause unnecessary network traffic and impair the performance of resource managers because of continued requesting and processing of policy updates.

Use the manual control of update notification to complete the process of modifying the master authorization database before update notifications are sent out to authorization servers with database replicas.

In manual mode, update notification uses the notifier thread pool as it does in automatic mode. Therefore, the manual mode setting is affected by the `max-notifier-threads` stanza entry value. For more information about this stanza entry, see "Set the number of update-notifier threads" on page 177.

Use the server replicate command

When you configure update notification to be manual, the manual application of the **server replicate** command controls this event.

```
pdadmin sec_master> server replicate -server test_server
```

When the **-server** option (test_server in the previous example) is specified, only that server is notified of changes to the master authorization database. A response indicates the success or failure of the notification and the replication.

When the **-server** option is not specified, all configured resource manager servers receive update notifications. A successful response indicates only that the policy server began sending out update notifications. The response does not indicate success or failure of the actual notification and replication processes.

The authorization required to run this command is the **s** action bit on the /Management/Server object.

For more information about the **server replicate** command, see the *IBM Security Access Manager for Web: Command Reference*.

Set the number of update-notifier threads

The policy server is responsible for synchronizing all database replicas in the domain.

When a change is made to the master database, notification threads announce this change to all replicas configured to receive update notifications. Each replica must then download the new information or the changes from the master.

The policy server configuration file, ivmgrd.conf, contains a stanza entry for setting the maximum number of update-notifier threads. This pool of threads allows simultaneous (parallel) notification.

For example, to concurrently notify 30 replicas of a database change, the thread pool must be set to at least 30. If there are more than 30 replicas, another round of notifications occurs (in this example, 30 at a time). All replicas are guaranteed to be notified, regardless of the value of this stanza entry.

The performance goal of the update-notifier threads value is to announce a database change as quickly as possible. Generally the value must be set to equal the number of existing replicas. This results in the performance advantage of a single pool of threads quickly accomplishing the notification task to all replicas at once.

The default event notifier thread pool is set as:

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

When the auto-database-update-notify stanza entry is set to yes, you must correctly configure this stanza entry and also the notifier-wait-time stanza entry. See also “Set the notification delay time.”

Set the notification delay time

When the policy server is instructed to change the master authorization database, it waits for a default period before sending out notifications to database replicas.

The default time delay is set at 15 seconds. This time delay is reset with each subsequent change to the database.

The purpose of the time delay is to prevent the policy server from sending individual replica notifications for each change in a series of database changes. The time delay helps to ensure optimal performance of the Security Access Manager system.

This performance feature is important for environments where batch changes are made to the authorization database. It is not efficient for policy changes to be sent to database replicas until all changes are made.

You can override this default notification time delay by changing the `notifier-wait-time` entry value in the `[ivmgrd]` stanza of the `ivmgrd.conf` configuration file. For example:

```
[ivmgrd]
notifier-wait-time = 20
```

By default, the value is set to 15 seconds.

When the `auto-database-update-notify` entry is set to `yes`, you must configure this entry and the `max-notifier-threads` entry. See also "Set the number of update-notifier threads" on page 177.

Chapter 14. High availability of the policy server

This chapter provides information about ensuring that Security Access Manager provides high availability for the policy server in case a server failure occurs.

This chapter describes how Security Access Manager supports the replication capability of the LDAP directory server to ensure that its data is always available.

For more concepts about high availability in an appliance environment, see High availability for the policy server.

Primary and replica LDAP servers

Security Access Manager allows primary and replica LDAP servers. The replica LDAP server, on a different node, can assume LDAP server operations if the primary LDAP server fails.

During failover, no write operations can occur. Only read-only LDAP server operations are permitted during failover.

See the LDAP server documentation for complete information about high availability of LDAP servers.

Chapter 15. Multiple-tenancy policy server

A *multiple-tenancy server* is a server that supports the hosting of multiple customers on a single server instead of on multiple client systems.

For example, your company might be sharing applications or data on your company server with your customer (for example, Smith-Davis Enterprises). Before adding data and information that belongs to another customer (for example, Systems, Inc.), you must ensure that these two customers cannot get access to the data or applications that belong to the other company.

Using a multiple-tenancy (multi-domain) server, you can run the applications or data for each company in an isolated server environment. Running in an isolated or partitioned server environment replaces the need to use multiple physical servers for each customer and their applications. Depending on the demands of your customers and their applications, you can host multiple clients on a single server. Replacing multiple servers with one server reduces the costs to your company for the services you provide to your customers. For example, fewer servers reduce hardware costs and reduce IT personnel burden. It is easier to manage a single server than it is to manage multiple servers.

A multiple-tenancy server is not necessarily less secure than the traditional one-server, one-client approach. Using technologies such as SSL and restricted access, you can protect two customers (users) on the same server from one another. Extra layers of security for multiple-user applications are designed into Security Access Manager. Security Access Manager compartmentalizes each domain to seal off users from one another rather than using the multiple-user security provisions that are provided by the operating system.

Multiple-tenancy is supported by Security Access Manager through the use of different Security Access Manager domains. These domains can be managed through the Security Access Manager administration interface. For more information about managing domains, see Chapter 5, “Domain management,” on page 55.

Chapter 16. Diagnostics and auditing

Security Access Manager provides ways to collect events that you can use for diagnostic and auditing purposes of the servers.

Events for diagnostics and auditing pertain to the operations of the Security Access Manager servers. These events do not pertain to the installation of these servers.

To enable diagnostics and auditing, you define which types of events to capture. When events are captured, they can be written to log files. Events can also be written to the standard output (STDOUT) device, to the standard error (STDERR) device, or to a combination of these destinations. Beyond these destinations, when events are captured, they can be redirected to a remote server or redirected for processing to an application that uses log agents.

During the installation of the Security Access Manager servers, the installation logs capture all messages for that specific installation. When using a native installation, the installation uses the operating system logs. For information about installation logs, see the *IBM Security Access Manager for Web: Troubleshooting Guide*.

Diagnostic events

For diagnostic purposes, define which message events and which trace events to capture. These events can help you troubleshoot problems.

To configure diagnostic events, define statements in the server-specific routing files. Each server has an associated routing file. The statements in these routing files are for both message events and trace events. You define the statements for message events by severity level. You define the statements for trace events by trace level and optionally by component.

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

Auditing events

For auditing purposes, define which audit, statistic, or other type of events to capture. Use these events to create snapshots of various server activities.

You can log audit events with Security Access Manager.

To configure auditing events, define stanza entries in the configuration files. Depending on your approach, define different stanza entries in different configuration files. For native Security Access Manager auditing, you define `logcfg` entries in the appropriate stanza of the server-specific configuration files.

For additional information about audit events, see the *IBM Security Access Manager for Web: Auditing Guide.*

Appendix A. Guidelines for changing configuration files

These guidelines are provided to help you change the Security Access Manager configuration files.

General guidelines

Use the following general guidelines when you change the configuration settings:

- Use the **config modify** command in the **pdadmin** command-line interface to update configuration files for Security Access Manager. See "config modify" in the *IBM Security Access Manager for Web: Command Reference* for more information and instructions for using these commands.
- There is no order dependency or location dependency for stanzas in any configuration file.
- Stanza entries are marked as required or optional. When an entry is required, the entry must contain a valid key and value.
- Do not change the names of the keys in the configuration files. Changing the name of the key might cause unpredictable results for the servers.
- Stanza entries and key names are case-sensitive. For example, `usessl` and `UseSSL` are treated as different entries.
- Spaces are not allowed for names of keys.
- For the key value pair format of *key* = *value*, the spaces that surround the equal sign (=) are not required. However, a good practice is to use spaces.
- Non-printable characters (such as tabs, carriage returns, and line feeds) that occur at the end of a stanza entry are ignored. Non-printable characters are ASCII characters with a decimal value less than 32.

Default values

Use the following guidelines when changing default configuration settings:

- Many values are created or modified only with configuration programs. Do not manually edit these stanzas or values.
- Some values are filled in automatically during configuration. These values are needed for the initialization of the server after the configuration.
- The default values for a stanza entry might be different, depending on the server configuration. Some key value pairs are not applicable to certain servers and are omitted from the default configuration file for this server.

Strings

Some values accept a string value.

When you manually edit the configuration file, use the following guidelines to change configuration settings that require a string:

- String values are expected to be characters that are part of the local code set.
- Additional or different restrictions on the set of allowable string characters might be imposed. For example, many strings are restricted to ASCII characters. Consult each stanza entry description for any restrictions.

- Double quotation marks are sometimes, but not always, required when you use spaces or more than one word for values. Refer to the descriptions or examples for each stanza entry.
- The minimum and maximum lengths of user registry-related string values, if there are limits, are imposed by the underlying registry. For example, for Active Directory the maximum length is 256 alphanumeric characters.

Defined strings

Some values accept a string value, but the value must be a string from a set of defined strings.

When you manually edit the configuration file, make sure that the string value that you type matches a valid value of one of the defined strings.

For example, the `[aznapi-configuration]` stanza section contains the following entry:

```
mode = {local|remote}
```

The value for `mode` is expected to be `local` or `remote`. Any other value is not valid and results in an error.

File names

Some values are file names.

For each stanza entry that expects a file name as a value, the description of the stanza entry specifies which of the following constructs are valid:

Filename
No directory path included.

Relative filename
A directory path is allowed but not mandatory.

These files typically are expected to be located relative to the location of a standard Security Access Manager directory. The stanza entry for each relative path name lists the root directory to which the file name is relative.

Fully qualified absolute path
An absolute directory path is required.

Some stanza entries allow more than one of these choices.

The set of valid characters in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks (").

Integers

Many stanza entries expect the value for the entry to be expressed as an integer.

When defining an entry with an integer, consider the following guidelines:

- Some stanza entries that take an integer value expect integer values within a valid range. The range is described in terms of a *minimum* value and a *maximum* value.

For example, in the [ivmgrd] stanza, the max-notifier-thread stanza entry has a minimum value of 1 thread and a maximum value of 128 threads.

- For some entries, the integer value must be positive and the minimum value is 1. For other entries, a minimum integer value of 0 is allowed.

  Use caution when setting an integer value to 0, which might disable the function that is controlled by that stanza entry. For example, in the [ivacld] stanza, the entry tcp-req-port = 0 disables the port number. Or, an integer value of 0 might indicate that the number is unlimited. For example, in the [ldap] stanza, the entry max-search-size = 0 means that there is no limit to the maximum search size.
- For some entries that require integer values, Security Access Manager does not impose an upper limit for the maximum number allowed. For example, there is typically no maximum for timeout-related values, such as timeout = *number* in the [ldap] stanza.

  For this type of entry, the maximum number is limited only by the size of memory allocated for an integer data type. This number can vary, based on the type of operating system. For systems that allocate 4 bytes for an integer, this value is 2147483647.

  However, as the administrator, use a number that represents the value that is most logical for the value you are trying to set.

Boolean values

Many stanza entries represent a Boolean value.

Security Access Manager recognizes the Boolean values yes and no.

Some of the entries in the configuration files are read by other servers and utilities. For example, many entries in the [ldap] stanza are read by the LDAP client. Some of these other programs recognize additional Boolean characters:

- yes or true
- no or false

Anything other than yes|true, including a blank value, is interpreted as no|false.

The recognized Boolean entries are listed for each stanza entry. Refer to the individual descriptions to determine when true or false are also recognized.

Appendix B. Configuration file reference

The way you use configuration files controls the operation of the Security Access Manager servers.

Each configuration file contains sections that are called *stanzas*.

Server configuration files are ASCII text-based and contain stanza entries. Configuration files are processed only when the servers start. The following configuration files are currently used by Security Access Manager:

pd.conf
: The configuration file that is used by the authentication server to configure the IBM Security Access Manager runtime. For details about the stanzas contained in this configuration file, see "IBM Security Access Manager runtime configuration file" on page 190.

*[instance-]***ivacld.conf**
: The configuration file that is used to configure an Security Access Manager authorization server instance. `[instance-]` represents the name of the specified authorization server instance. If an authorization server instance name contains an empty string, the configuration file is called `ivacld.conf`. For details about the stanzas contained in this configuration file, see "Authorization server configuration file" on page 190.

ivmgrd.conf
: The configuration file that is used to configure the Security Access Manager policy server. For details about the stanzas contained in this configuration file, see "Policy server configuration file" on page 191.

ldap.conf
: The configuration file that is used by the LDAP-based server to configure the LDAP-based user registry. For details about the stanzas that are contained in this configuration file, see "LDAP server configuration file" on page 191.

aznAPI.conf
: A template configuration file that is used to configure any Security Access Manager resource manager. For details about the stanzas that are contained in this template file, see "Resource manager configuration files" on page 192.

Location of configuration files

If you did not change the installation directories during Security Access Manager installation, the configuration files are in one of the following platform-specific directories:

AIX, Linux, and Solaris operating systems
: `/opt/PolicyDirector/etc`

Windows operating systems
: `c:\program files\tivoli\policy director\etc`

If you did not change the installation directories while installing the common audit service, the templates for the configuration files are located in one of the following platform-specific directories:

AIX, Linux, and Solaris operating systems
`/opt/PolicyDirector/etc/audit`

Windows operating systems
`c:\program files\tivoli\policy director\etc\audit`

IBM Security Access Manager runtime configuration file

For Security Access Manager servers, you must have the `pd.conf` configuration file.

Use this configuration file to automate server startup, to indicate whether the IBM Security Access Manager runtime is configured, and specify information about the user registry.

Stanza entries for automating server startup are in the `[pdrte]` stanza of the `pd.conf` configuration file.

This configuration file can include the following stanzas:
- `[meta-info]`
- `[pdrte]`
- `[ssl]`
- `[manager]`

The unconfiguration of the server with the `pd.conf` configuration file also queries information from this configuration file.

Authorization server configuration file

When you use the Security Access Manager authorization server, you must have the `[instance-]ivacld.conf` server configuration file.

Use this configuration file to customize the operation of each authorization server.

This configuration file can include the following stanzas:
- `[meta-info]`
- `[ivacld]`
- `[ldap]`
- `[ssl]`
- `[manager]`
- `[aznapi-configuration]`
- `[xmladi-attribute-definitions]`
- `[aznapi-entitlement-services]`
- `[aznapi-external-authzn-services]`
- `[aznapi-pac-services]`
- `[aznapi-cred-modification-services]`
- `[aznapi-admin-services]`
- `[configuration-database]`

The unconfiguration of the server with the *[instance-]*ivacld.conf configuration file also queries information from this configuration file.

Note: *[instance-]* represents the name of the specified authorization server instance. If an authorization server instance name contains an empty string, the configuration file is called ivacld.conf.

Policy server configuration file

When you use the Security Access Manager policy server, you must have the ivmgrd.conf server configuration file.

Use this configuration file to customize the operation of each policy server.

This configuration file can include the following stanzas:

- [meta-info]
- [ivmgrd]
- [ldap]
- [ssl]
- [aznapi-configuration]
- [xmladi-attribute-definitions]
- [aznapi-entitlement-services]
- [aznapi-pac-services]
- [aznapi-cred-modification-services]
- [aznapi-external-authzn-services]
- [delegated-admin]
- [configuration-database]
- [domains]
- [domain=*domain_name*]

The unconfiguration of the server with the ivmgrd.conf configuration file also queries information from this configuration file.

LDAP server configuration file

When you use LDAP as the user registry for Security Access Manager, use the ldap.conf configuration file to customize the LDAP-based stanza entries.

This configuration file includes the following stanzas:

- [ldap]
- [meta-info]

Note: The ldap.conf configuration file contains the following stanzas that contain entries that are for internal use only:

- [ldap-generic-general]
- [ldap-generic-pwd-change-error-map]
- [ldap-generic-acls]

Do not modify any of the values that are defined in these stanzas.

LDAP client with Active Directory server configuration file

When you use an LDAP client to retrieve data for the Active Directory user registry to which the Security Access Manager policy server is configured, you must have the `activedir_ldap.conf` configuration file.

Use this configuration file to customize the operation of each Active Directory user registry.

For example, you might have multiple platforms where the policy server is configured to use the Active Directory user registry. Other blades, such as WebSEAL on one platform, and the authorization server are configured to use the LDAP client to retrieve data from that Active Directory user registry on another platform.

This configuration file can include the following stanzas:
- `[meta-info]`

Resource manager configuration files

Security Access Manager provides a sample file that includes the more common configuration stanzas needed by resource managers.

Your documentation sources, when implementing your own plug-in or security-enhanced application, include the *Authorization Java Classes Developer Reference*.

When creating your own security resource manager or extending the functions provided by Security Access Manager, you can use the `aznAPI.conf` configuration file.

This configuration file can include the following stanzas:
- `[aznapi-configuration]`
- `[xmladi-attribute-definitions]`
- `[ssl]`
- `[ldap]`
- `[aznapi-entitlement-services]`
- `[aznapi-pac-services]`
- `[aznapi-cred-modification-services]`
- `[aznapi-external-authzn-services]`
- `[aznapi-admin-services]`
- `[manager]`

Appendix C. Configuration file stanza reference

Within configuration files, stanza labels occur within brackets, such as [*stanza-name*].

For example, the [ssl] stanza in the ivmgrd.conf configuration file defines the Secure Sockets Layer (SSL) configuration settings for the policy server. The [ldap] stanza defines the configuration settings that are required by the policy server to communicate with an LDAP-based user registry.

Each stanza in a Security Access Manager configuration file contains one or more key value pairs. The pairs contain information that is expressed as a paired set of parameters. Each stanza entry is a key-value pair in the following format:

```
key = value
```

Do not change the names of the keys in the configuration files. Changing the name of the key might cause unpredictable results in the servers. Spaces surrounding the equal sign (=) are typically used, but are not required.

The initial installation of Security Access Manager establishes many of the default values. Some values are static and never change; other values can be modified to customize server functionality and performance.

The following stanza descriptions provide a list of the valid stanza entries. Each stanza entry consists of key value pairs. Each stanza entry includes a description of its default behavior, when applicable.

[aznapi-admin-services] stanza

An administration service plug-in enables applications to do application-specific administration tasks.

The administration service plug-in is accessed by a calling application with one of the Security Access Manager administration interfaces.

The calling application can be an administrative utility such as the Web Portal Manager or **pdadmin** utility. The calling application can be a custom-built application that uses the Security Access Manager administration APIs.

The administration service maps the administration API calls to the corresponding administration service API calls and carries out the requested action. Each administration service plug-in is a stand-alone module that is dynamically loaded into the authorization service.

The parameters for configuring Security Access Manager administration service plug-ins are declared in the [aznapi-admin-services] stanza of these configuration files provided by Security Access Manager:

- The ivmgrd.conf configuration file for the policy server
- The *[instance-]*ivacld.conf configuration file for the authorization server
- The configuration files for the configured administration service plug-ins for your resource managers

The aznAPI.conf configuration file is provided with Security Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins typically provide the standard functions. Before you implement service plug-ins, read and thoroughly understand the concepts in the *Authorization C API Developer Reference*.

service-id

This stanza entry defines the authorization API service for functions that enable a plug-in to obtain the contents of a defined portion of the protected object hierarchy. Service functions can also enable a plug-in to define application-specific administration tasks that also return commands that do those tasks.

Syntax

```
service-id = {short_name|path_to_dll}
     [-pobj protected_object_hierarchy_name ] [& params]
```

Description

Defines the authorization API service for functions that enable a plug-in to obtain the contents of a defined portion of the protected object hierarchy. Service functions can also enable a plug-in to define application-specific administration tasks that also return commands that do those tasks.

Each stanza entry defines different types of aznAPI service.

Options

Each entry has the following format.

service-id

Developer-specified ID of the administration service. An authorization API application can register more than one administration service plug-in, but each must have a unique service ID.

short_name | *path_to_dll*

The path to the dynamic link library (DLL) that contains the executable code for the service.

If the DLL is in a directory that is normally searched by the system (for example, /usr/lib on AIX, Linux, and Solaris operating systems or the value of the PATH environment variable on Windows operating systems), do not specify the full path to the DLL, specify only the DLL name. If you want a platform-independent DLL name, so it can be loaded on any supported platform, provide a short name. The short name is added as a prefix and appended with known library prefixes and suffixes for each platform, and each possibility is searched in turn. For example, with a short name of azn_ent_user, the following names are automatically searched for on each platform:

AIX

```
libazn_ent_user.so
libazn_ent_user.a
```

Linux libazn_ent_user.so

Solaris

```
libazn_ent_user.so
```

Windows

azn_ent_user.dll

protected_object_hierarchy_name

Optional: The name of the protected object hierarchy. This option refers either to the name of a protected object space (hierarchy) or to a protected object. Protected object hierarchy names must be unique for each administration service plug-in within the scope of an authorization API application. To support failover, multiple authorization API application instances can be registered to service the same protected object hierarchy names. Failover support allows for the administration of an object space if a particular authorization API application server fails.

params

Optional: The additional initialization arguments that can be passed to the external authorization service. The arguments must be preceded by the ampersand (&); for example, `& -server fred`. The authorization service does not process the characters after the ampersand. It passes these characters directly to the administration service plug-in. The service definition is described in more detail in the *Authorization C API Developer Reference*.

Usage

Optional

Default value

There is no default value.

Example

```
AZN_ADMIN_SVC_TRACE = pdtraceadmin
```

[aznapi-configuration] stanza

Security Access Manager allows a highly flexible approach to authorization through the use of the authorization API.

The standards-based authorization API allows applications to make calls to the centralized authorization service. Security Access Manager provides built-in support of user name and password authentication as well through the authorization API.

The configuration key value pairs that are used for configuring audit files for Security Access Manager servers are in the `[aznapi-configuration]` stanza of each of the following configuration files:

- The `ivmgrd.conf` configuration file for the policy server
- The *[instance-]*`ivacld.conf` configuration file for the authorization server
- The `pdmgrproxyd.conf` configuration file for the policy proxy server
- The configuration files for your resource managers

  Other stanza entries that apply to the configuration files of your resource managers are described in the *Authorization C API Developer Reference*. Read and thoroughly understand these concepts so that they can provide the required standard functions. A sample `aznAPI.conf` configuration file is provided with Security Access Manager to use as a guide for creating your own resource manager configuration file.

audit-attribute

This stanza entry specifies the name of the access decision information (ADI) attribute to audit.

Syntax

```
audit-attribute = azn-attr
```

Description

Name of the access decision information (ADI) attribute to audit. An attribute can establish accountability by providing information to help identify potentially inappropriate access of assets. You can grant or deny access based on the rules that are applied to attributes.

For example, the WebSEAL switch-user authentication feature provides a mechanism to allow certain users to impersonate another user. When switch-user is used, an authorization request is evaluated against an assumed identity rather than the actual identity of the user. It is desirable to allow administrators to capture the user's actual identity.

You can audit the names or descriptions of the Security Access Manager policies (ACL, POP, and authorization rule) that are applied to the object that is accessed.

Options

azn_attr
: The authorization API attribute represents an alphanumeric string that is not case-sensitive. String values are expected to be characters that are part of the local code set.

Usage

Optional

Default value

There is no default value.

Example

The following example shows the configuration for WebSEAL:

```
audit-attribute = tagvalue_su-admin
```

azn-app-host

This stanza entry specifies the host name that the policy server uses when communicating with the resource manager.

Syntax

```
azn-app-host = other_hostname
```

Description

Attribute that is used to specify the host name that the policy server uses when communicating with the resource manager.

Options

For *other_hostname*, you can provide any valid internet host name. If this attribute is not specified, the default host name is used. Examples of valid host names:

- `mycomputer.city.company.com`
- `mycomputer`

By default, this attribute is disabled. When disabled, the stanza entry is commented out by using a pound sign (#) at the beginning of the stanza entry. The following example shows a commented out entry:

```
#azn-app-host = libra
```

To enable this value, uncomment the entry by removing the pound sign. Be sure to include a host name value.

Usage

Optional

Default value

There is no default value.

Example

```
azn-app-host = libra.dallas.ibm.com
```

azn-server-name

This stanza entry specifies the unique name of the Security Access Manager resource manager, authorization server, or policy server, that is configured into the domain.

Syntax

```
azn-server-name = server–hostname
```

Description

Unique name of the Security Access Manager resource manager, authorization server, or policy server, that is configured into the domain. The hyphen (-) character is required.

Note: The host name is generated and set during configuration. Do not edit this stanza entry.

cache-refresh-interval

This stanza entry specifies the poll interval (in seconds) between checks for updates to the master policy database.

Syntax

```
cache-refresh-interval = {disable|default|number_seconds}
```

Description

Poll interval (in seconds) between checks for updates to the master policy database.

Note: The local cache is rebuilt only if an update is detected.

This stanza entry is not used in the `ivmgrd.conf` file.

Options

`disable`
The interval value in seconds is not set.

`default`
The default value of 600 seconds is used.

number_seconds
The exact time interval in number of seconds. This value is between 0 and the size of an unsigned integer. The unsigned integer is approximately 136 years.

Usage

Optional

Default value

```
default
```

Example

```
cache-refresh-interval = 500
```

cred-attributes-entitlement-services

This stanza entry specifies the service that you can use to add external information to the user credential. The addition is in the form of credential attributes and allows applications to use that information in making access decisions.

Syntax

```
cred-attributes-entitlement-services =
    {short_name_entitlement_service|path_to_dll}
```

Description

Service that you can use to add external information to the user credential in the form of credential attributes. The addition allows applications to use that information in making access decisions. These extended attributes are stored in the user registry.

This service can also work with attributes with an API call. A list of authorization API entitlement service IDs are queried by the `azn_id_get_creds()` interface. The query compiles a list of attributes to be added to the user credential while the credential is being built.

A list of service identifiers, which can be found within the `[aznapi-entitlement-services]` stanza, is queried to compile a list of attributes. The attributes are added to the user credential while the credential is being built. Each service ID is queried in the order it is declared in the list. The attribute returned is inserted into the credential attribute list of each credential that is built. The following example shows two entries from the credential attribute list:

```
cred-attribute-entitlement-services = myEntSvcID
cred-attribute-entitlement-services = myOtherEntSvcID
```

Note: You cannot use this stanza entry to override read-only attributes in the credential attribute list that include the principal name, principal UUID, and others. The exception to this rule is for the azn_cred_groups attribute.

The *Authorization C API Developer Reference* lists the read-only attributes and contains more information about this service. The document explains why administrators who do not want this capability must ensure that the azn_mod_rad service is not loaded by the application.

Usage

Optional

Default value

There is no default value.

Example

```
cred-attribute-entitlement-services = myEntSvcID
```

db-file

This stanza entry specifies the name and location of the resource manager policy database cache file.

Syntax

db-file = *fully_qualified_path*

Description

Name and location of the resource manager policy database cache file. This value must be specified, and each server provides its own value.

This stanza entry is not used in the ivmgrd.conf file. The policy server has its own stanza entries for specifying the path to the master policy database.

Options

fully_qualified_path

Represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of valid characters in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). Windows operating systems path names, however, can have a backward slash (\) or a colon (:).

For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

Usage

Required for each specified server.

Default value

There is no default value.

Example

The following example sets the policy database with an absolute path on a Windows operating system:

```
db-file = C:\pd\db\ivacld.db
```

The following example sets the policy database with a relative path on a AIX, Linux, or Solaris operating system:

```
db-file = ./authzn_demo.db
```

dynamic-adi-entitlement-services

This stanza entry specifies the dynamic access decision information (ADI) retrieval entitlement service.

Syntax

```
dynamic-adi-entitlement-services = entitlement_service
```

Description

Dynamic access decision information (ADI) retrieval entitlement service.

Options

entitlement_service

A string value for the container names of the required ADI. A list of configured authorization API entitlements service identifiers (IDs) is queried by the authorization rules engine when missing ADI is detected during an authorization rule evaluation.

When ADI is found to be missing during a rule evaluation, each service in this list is queried in the order defined in this entry. These stanza entries must refer to existing entitlements services.

The service ID (for example, `bank_A_ADI`) is loaded by using service entries in the entitlement service configuration `[aznapi-entitlement-services]` stanza or in an initialization attribute.

See "dynamic-adi-entitlement-services" on page 135 and the *Authorization C API Developer Reference* for more information about rules processing and this service.

Usage

Optional

Default value

There is no default value.

Example

```
[aznapi-entitlement-services]
dynamic-adi-entitlement-services = bank_A_ADI
dynamic-adi-entitlement-services = bank_B_ADI
```

input-adi-xml-prolog

This stanza entry specifies the prolog to be added to the top of the XML document. This document is created with the Access Decision Information (ADI) needed to evaluate a Boolean authorization rule.

Syntax

```
input-adi-xml-prolog = prolog_attrs
```

Description

Prolog to be added to the top of the XML document. This document is created with the Access Decision Information (ADI) needed to evaluate a Boolean authorization rule.

If a style sheet prolog is specified, that prolog is imported into the empty style sheet. If no prolog is specified, a default prolog value is used instead. All of the required prolog attributes are specified in the default prolog entries.

Note: If any of these attributes are changed or omitted from the entry, the authorization client fails to start and returns an error.

Options

prolog_attrs

Prolog attributes that are required by the authorization engine and include the following attributes:

```
<?xml version="1.0" encoding="UTF-8"?>
```

See “input-adi-xml-prolog and xsl-stylesheet-prolog” on page 135.

Usage

Optional

Example

```
input-adi-xml-prolog = <?xml version="1.0" encoding="UTF-8"?>
```

listen-flags

This stanza entry specifies whether the reception of policy cache update notifications is on or off.

Syntax

```
listen-flags = {enable|disable}
```

Description

Specification of whether to turn on or off the reception of policy cache update notifications.

Options

enable Activates the notification listener.

disable
Deactivates the notification listener.

Usage

Optional

Default value

```
disable
```

Example

```
listen-flags = enable
```

logcfg

This stanza entry enables logging and auditing for the application.

Syntax

```
logcfg = audit.azn:[log-agent][[param[=value]] ...]
```

Description

Enables logging and auditing for the application. Category, destination, and other parameters are used to capture Security Access Manager auditing and logging events.

Each server provides its own setting for event logging in its corresponding configuration file.

Options

audit.azn:*log-agent*
Category of auditing event. Also specifies that the destination where *log-agent* is one of the following agents:

- stdout
- stderr
- file
- pipe
- remote

param=value
Allowable parameters. The parameters vary, depending on the category, the destination of events, and the type of auditing you want to do.

See *IBM Security Access Manager for Web: Troubleshooting Guide* for information about the log agents and the configuration parameters.

Usage

Optional

Default value

Remove the number signs (#) at the beginning of the configuration file lines to enable authentication or authorization auditing (or both) for the application.

Example

```
logcfg = audit.azn:file path=audit.log,flush_interval=20,log_id=audit_log
```

mode

This stanza entry specifies the operating mode for the resource manager.

Syntax

```
mode = {local|remote}
```

Description

Operating mode for the resource manager. This value cannot be changed after resource manager configuration.

Note: This stanza entry is set during configuration. Do not change it.

Options

local The resource manager uses a local policy cache.

remote
The resource manager uses a remote policy cache that is maintained by the authorization server.

Some configuration attributes apply only to resource managers that are configured to use local mode.

Usage

Required

Default value

```
local
```

Example

```
mode = remote
```

pd-user-name

This stanza entry specifies the Security Access Manager user account for the resource manager server.

Syntax

pd-user-name = *server_name/hostname*

Description

Security Access Manager user account for the resource manager server, either the policy proxy server, authorization server, or policy server, that is configured into the domain. The forward slash (/) character is required.

Note: The server name or host name is generated and set during configuration. Do not edit this stanza entry.

pd-user-pwd

This stanza entry specifies the Security Access Manager user account password for the resource manager.

Syntax

```
pd-user-pwd = server_password
```

Description

Security Access Manager user account password for the resource manager, which can be the authorization server or policy server that is configured into the domain.

Note: The server password is generated and set during configuration. Do not edit this stanza entry.

permission-info-returned

This stanza entry specifies the set of attributes that the caller wants to receive from the `azn_decision_access_allowed_ext()` function in the permission information attribute list.

Syntax

```
permission-info-returned = {attribute1 attribute2 ...}
```

Description

Set of attributes that the caller wants to receive from the `azn_decision_access_allowed_ext()` function in the permission information attribute list. Before you use this stanza entry and value, read and thoroughly understand the concept in the *Authorization C API Developer Reference*.

You can also define your own attributes. For example, you can set an attribute on an ACL with the **acl modify** command with the **set attribute** option.

When you add an attribute name to the list, the attribute can be returned only as permission information if it is applicable to the current decision call.

Options

For a list of the strings recognized by the authorization engine, see *Authorization C API Developer Reference*.

Usage

Optional

Default value

No information is returned.

Example

The following example returns permission information for all attributes in the list:

```
permission-info-returned = azn_perminfo_all_attrs
```

policy-cache-size

This stanza entry specifies the maximum size of the in-memory policy cache.

Syntax

```
policy-cache-size = size
```

Description

Maximum size of the in-memory policy cache. This size is configurable. The cache consists of policy and the relationships between policy and resources. The knowledge that a resource has no directly associated policy is also cached.

Specify the maximum cache size relative to the number of policy objects defined, the number of resources protected, and the available memory.

As a starting point, use the following algorithm:

> 3 * (number of policy objects + number of protected resources)

This value controls how much information is cached. A larger cache potentially improves the application performance, but uses additional memory as well.

Options

size Size is specified as the number of entries.

Usage

Optional

Default value

```
32768
```

Example

```
policy-cache-size = 32768
```

resource-manager-provided-adi

This stanza entry specifies the prefix that the authorization engine uses to determine the set of missing access decision information (ADI) provided by the resource manager.

Syntax

```
resource-manager-provided-adi = prefix
```

Description

Prefix that the authorization engine uses to determine the set of missing access decision information (ADI) provided by the resource manager. To specify more than one prefix, add multiple stanza entries.

These entries must refer to existing entitlements services that were loaded with service entries in the `[aznapi-entitlement-services]` configuration stanza or that were loaded with an initialization attribute. If an ADI is found to be missing during a rule evaluation, each service in this list is queried in the order defined.

Options

prefix A string prefix for its value. For example, you might want to notify the authorization engine that any ADI beginning with `sales_customer_` is provided by the resource manager application. The stanza entry is:

```
resource-manager-provided-adi = sales_customer_
```

See “resource-manager-provided-adi” on page 134.

Usage

Optional

Default value

There is no default value.

Example

The following example shows multiple stanza entries:

```
resource-manager-provided-adi = sales_item_
resource-manager-provided-adi = sales_customer_
```

xsl-stylesheet-prolog

This stanza entry specifies the prolog to be added to the top of the XSL style sheet with the XSL text that defines a Boolean authorization rule.

Syntax

```
xsl-stylesheet-prolog = prolog_attrs
```

Description

The prolog to be added to the top of the XSL style sheet that is created with the XSL text that defines a Boolean authorization rule.

If a style sheet prolog is specified, that prolog is imported into the empty style sheet. If no prolog is specified, a default prolog value is used instead. All of the required prolog attributes are specified in the default prolog entries.

When not specified, the default XSL style sheet prolog is:

```
!<-- Required for XSLT language -->
<?xml version="1.0" encoding='UTF-8'?>
<xsl:stylesheet xmlns:xsl=
   "http://www.w3.org/1999/XSL/Transform" version="1.0">

!<-- Required to constrain output of rule evaluation -->
<xsl:output method="text" omit-xml-declaration="yes"
    encoding='UTF-8' indent="no" />

!<-- Need this to ensure default text node printing is
      off -->
<xsl:template match="text()"></xsl:template>
```

Note: If any of the required prolog attributes are changed or omitted from the entry, then the authorization client fails to start and returns an error.

Use caution when changing this setting. See "input-adi-xml-prolog and xsl-stylesheet-prolog" on page 135.

Options

prolog_attrs
: Prolog attributes that are required by the authorization server.

Usage

Optional

Example

See "XML namespace definitions" on page 124 for a complete explanation of the name space example.

[aznapi-cred-modification-services] stanza

A credential modification service plug-in enables authorization API applications to do modifications on a Security Access Manager credential.

The credentials modification service can then return this modified credential for use by the calling application. Applications can use this service to add additional information to a user's credential. For example, this additional information can include the credit card number and credit limit of the user. Each credential modification service plug-in is a stand-alone module that is dynamically loaded into the authorization service.

The parameters for configuring Security Access Manager credential modification service plug-ins are declared in the `[aznapi-cred-modification-services]` stanza of each of the configuration files provided with Security Access Manager:

- The `ivmgrd.conf` configuration file for the policy server
- The *`[instance-]`*`ivacld.conf` configuration file for the authorization server
- The configuration file for configured credentials modification service plug-ins for your resource managers

  The `aznAPI.conf` configuration file is provided with Security Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins typically provide the standard functions. Before you implement service plug-ins, read and thoroughly understand the concepts in the *Authorization C API Developer Reference*.

service-id

This stanza entry defines the authorization API service for the credentials attribute list modification service.

Syntax

```
service-id = short_name|path_to_dll [ & params ...]
```

Description

Defines the authorization API service for the credentials attribute list modification service. Each stanza entry defines different types of aznAPI service.

Options

Each entry has the following format:

service-id
: Developer-specified ID of the credential modification service. The service ID string must be unique.

short_name|*path_to_dll*
: The path to the dynamic link library (DLL) that contains the executable code for the service.

If the DLL is in a directory that is normally searched by the system (for example, `/usr/lib` on AIX, Linux, and Solaris operating systems or the value of the PATH environment variable on Windows operating systems), do not specify the full path to the DLL, specify only the DLL name. If you want a platform-independent DLL name, so it can be loaded on any supported platform, provide a short name. The short name is appended with known library prefixes and suffixes for each platform, and each possibility is searched in turn. For example, with a short name of `azn_ent_user`, the following table shows the names that are automatically searched for on each platform:

AIX

```
libazn_ent_user.so
libazn_ent_user.a
```

Linux `libazn_ent_user.so`

Solaris
`libazn_ent_user.so`

Windows
`azn_ent_user.dll`

params
: Optional: The parameters to pass to the service when it is initialized by the aznAPI service. Parameters are considered to be all data that follow the ampersand (&). The service definition is described in more detail in the *Authorization C API Developer Reference*.

Usage

Optional

Default value

There is no default value.

Example

```
AZN_MOD_SVC_RAD_2AB = azn_mod_rad
```

[aznapi-entitlement-services] stanza

An entitlement services plug-in enables authorization API applications to retrieve the entitlements for a user from an entitlements repository.

Each entitlement services plug-in is a stand-alone module that is dynamically loaded into the authorization service.

The stanza entries for configuring Security Access Manager entitlement services plug-ins are declared in the `[aznapi-entitlement-services]` stanza of each of these configuration files provided by Security Access Manager:

- The `ivmgrd.conf` configuration file for the policy server
- The *`[instance-]`*`ivacld.conf` configuration file for the authorization server
- The configuration file for configured entitlement services plug-ins for your resource managers

  The `aznAPI.conf` configuration file is provided with Security Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins typically provide the standard functions. Before you implement service plug-ins, read and thoroughly understand the concepts in the *Authorization C API Developer Reference*.

service-id

This stanza entry defines the authorization API service for the entitlement services of the protected objects.

Syntax

```
service-id = {short_name|path_to_dll} [ & params ...]
```

Description

Defines the authorization API service for the entitlement services of the protected objects. Each stanza entry defines different types of aznAPI service.

Options

Each entry has the following format:

service-id

Developer-specified ID by which the service can be identified by the aznAPI client. The service ID string must be unique.

short_name|*path_to_dll*

The path to the dynamic link library (DLL) that contains the executable code for the service.

If the DLL is in a directory that is normally searched by the system (for example, `/usr/lib` on AIX, Linux, and Solaris operating systems or the value of the PATH environment variable on Windows operating systems), do not specify the full path to the DLL, specify only the DLL name. If you want a platform-independent DLL name, so it can be loaded on any supported Security Access Manager platform, provide a short form library name. The short name is appended with known library prefixes and suffixes for each platform, and each possibility is searched in turn. For example, with a short form library name of `azn_ent_user`, the following names that are automatically searched for on each platform:

AIX

```
libazn_ent_user.so
libazn_ent_user.a
```

Linux `libazn_ent_user.so`

Solaris

```
libazn_ent_user.so
```

Windows

```
azn_ent_user.dll
```

params Optional: One or more parameters to pass to the service when it is initialized by the aznAPI service. Parameters are considered to be all data that follow the ampersand (&). The service definition is described in more detail in the *IBM Security Access Manager for Web: Authorization C API Developer Reference*.

Usage

Optional

Default value

There is no default value.

Example

```
credattrs_ent_svc = azn_ent_cred_attrs
```

[aznapi-external-authzn-services] stanza

An external authorization service plug-in is an optional extension of the Security Access Manager authorization service that you can use to impose additional authorization controls and conditions.

You can use an external authorization service plug-in to force authorization decisions to be made based on application-specific criteria that are not known to the Security Access Manager authorization service. Each external authorization service plug-in is a stand-alone module that is dynamically loaded into the authorization service.

The parameters for configuring Security Access Manager external authorization service plug-ins are declared in the `[aznapi-external-authzn-services]` stanza of this configuration file provided by Security Access Manager:

- The `ivmgrd.conf` configuration file for the policy server
- The *`[instance-]`*`ivacld.conf` configuration file for the authorization server
- The configuration file for configured external authorization service plug-ins for your resource managers

  The `aznAPI.conf` configuration file is provided with Security Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins typically provide the standard functions. Before you implement service plug-ins, read and thoroughly understand the concepts in the *Authorization C API Developer Reference*.

policy-trigger

This stanza entry defines the authorization API service for external authorization service definitions that force authorization decisions to made based on application-specific criteria.

Syntax

```
policy-trigger = {short_name|path_to_dll} [-weight number]
     [ & params ...]
```

Description

Defines the authorization API service for external authorization service definitions that force authorization decisions to made based on application-specific criteria. Each stanza entry defines different types of aznAPI service, and each entry is the same format.

Options

policy-trigger
: The policy trigger is the way that an external authorization service is started. It is either a service ID or an access control list (ACL) action string. For example, it can be `my_service_1` or `Trx`. If the service is defined an ID, the service ID is used as an extended attribute on a POP that triggers the external authorization service when an object has this POP attached to it. If the service is defined with an ACL action string, the service is started when this ACL action mask is requested as part of an authorization decision.

  The policy trigger can be any string that is recognized as a valid key name. The *policy-trigger* is case-sensitive, because the actions themselves are case-sensitive. However, the policy trigger is not case-sensitive if the trigger is a POP attribute.

short_name|*path_to_dll*
: The path to the dynamic link library (DLL) that contains the executable code for the service.

  If the DLL is in a directory that is normally searched by the system (for example, `/usr/lib` on AIX, Linux, and Solaris operating systems or the value of the PATH environment variable on Windows operating systems), do not specify the full path to the DLL. Specify only the DLL name. If you want a platform-independent DLL name, so it can be loaded on any supported platform, provide a short name. The short name is appended with known library prefixes and suffixes for each platform, and each possibility is searched in turn. For example, with a short name of `azn_ent_user`, the following names that are automatically searched for on each platform:

  AIX

  ```
  libazn_ent_user.so
  libazn_ent_user.a
  ```

  Linux `libazn_ent_user.so`

  Solaris

  ```
  libazn_ent_user.so
  ```

Windows
`azn_ent_user.dll`

[-weight *number*]
Optional: Specifies the weight assigned in the access decision process of the external authorization service. This option is an unsigned **size_t** value. This value signifies the weight. In the entire decision process, the weight is specified in any decision that is returned by the external authorization service. The default value is `101`.

params Optional: Additional initialization information to pass to the external authorization service in the form of arguments. The arguments must be preceded by the ampersand (&); for example, `& -server fred`. The service definition is described in more detail in the *Authorization C API Developer Reference*.

Usage

Optional

Default value

There is no default value.

[aznapi-pac-services] stanza

A PAC services plug-in gives authorization API applications the ability to move Security Access Manager credentials back and forth between the native Security Access Manager credentials format and an alternate format called *privilege attribute certificate* (PAC).

Each PAC services plug-in is a stand-alone module that is dynamically loaded into the authorization service.

Identity information can be obtained from a PAC. Applications can convert user credentials to PACs for use within other authorization domains. Applications can then pass the PACs to a server in another authorization domain and do an operation.

The stanza entries for configuring Security Access Manager PAC services plug-ins are declared in the `[aznapi-pac-services]` stanza of each of these configuration files provided by Security Access Manager:

- The configuration file for configured PAC services plug-ins for your resource managers

The `aznAPI.conf` configuration file is provided with Security Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins typically provide the standard functions. Before you implement service plug-ins, read and thoroughly understand the concepts in the *Authorization C API Developer Reference*.

service-id

This stanza entry defines the authorization API service for the Security Access Manager privilege attribute certificate (PAC) encoding service.

Syntax

```
service-id = {short_name|path_to_dll}
      [ & params ... ]
```

Description

Defines the authorization API service for the Security Access Manager privilege attribute certificate (PAC) encoding service. Each stanza entry defines different types of aznAPI authorization service.

Options

Each entry has the following format:

service-id
: Developer-specified ID of the PAC service that produces the PAC. The service ID string must be unique.

short_name | *path_to_dll*
: The path to the dynamic link library (DLL) that contains the executable code for the service executable.

  If the DLL is in a directory that is normally searched by the system (for example, `/usr/lib` on AIX, Linux, and Solaris operating systems or the value of the PATH environment variable on Windows operating systems), do not specify the full path to the DLL, specify only the DLL name. If you want a platform-independent DLL name, so it can be loaded on any supported platform, provide a short name. The short name is appended with known library prefixes and suffixes for each platform, and each possibility is searched in turn. For example, with a short form library name of `azn_ent_user`, the following names that are automatically searched for on each platform:

  AIX

  ```
  libazn_ent_user.so
  libazn_ent_user.a
  ```

  Linux `libazn_ent_user.so`

  Solaris
  `libazn_ent_user.so`

  Windows
  `azn_ent_user.dll`

params
: Optional: Parameters to pass to the service when it is initialized by the aznAPI service. Parameters are considered to be all data that follow the ampersand (&). The service definition is described in more detail in the *IBM Security Access Manager for Web: Authorization C API Developer Reference*.

Usage

Optional

Default value

There is no default value.

[configuration-database] stanza

The stanza entry defines the name and location of the Security Access Manager obfuscated password configuration file.

Security Access Manager creates a configuration file that contains all the obfuscated entries. For example, All bind (log in) passwords are obfuscated and placed in the configuration file. Both the existing configuration file and the obfuscated configuration file have the same file name, except that `.obf` is appended to the file name (for example, `ivmgrd.conf.obf`).

In addition, Security Access Manager creates the `[configuration-database]` stanza, as needed, whenever an obfuscated entry is automatically added to the obfuscated configuration file. This stanza has a stanza entry that points to the name and location of the obfuscated configuration file. The `[configuration-database]` stanza can be in every configuration file, including the `pd.conf` configuration file, if an obfuscated value is added to the file.

Never edit the entry in the `[configuration-database]` stanza. The one exception might be if the file is to be moved permanently to a different location. This scenario is the only circumstance in which you modify the file name and location. Remember that whenever the configuration file is moved to a different location, you must move the obfuscated file also.

file

This stanza entry specifies the file name and location of the obfuscated configuration file information.

Syntax

```
file = fully_qualified_path
```

Description

File name and location of the obfuscated configuration file information.

Note: The obfuscated password is generated and set by the configuration utility. Do not edit this stanza entry.

The name of the obfuscated configuration file is the same name as the related configuration file name. The file extension can be anything, but the extension is usually `.conf.obf`. For example, the obfuscated configuration file for `ldap.conf` is `ldap.conf.obf`.

Options

fully_qualified_path

Represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). Windows operating systems path names, however, can have a backward slash (\) or a colon (:). For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

Usage

Conditional. This stanza entry is required only if, during configuration, passwords were obfuscated.

Default value

The following table shows the default installation location by platform.

Platform	File name
Linux or UNIX	`/opt/PolicyDirector/etc/`*`server_name`*`.conf.obf`

Windows	`c:\program files\tivoli\policy director\etc\`
	`server_name``.conf.obf`

Example

The following example of setting the location of the obfuscated configuration file when using Microsoft Active Directory as the user registry on a Windows operating system:

```
c:\program files\tivoli\policy director\etc\ldap.conf.obf
```

[delegated-admin] stanza

The Security Access Manager configuration can require that the user is authorized to view each group that is returned in the group list. Or, the user can be authorized to return the list without authorizing first.

For delegated administration, use one type of interface throughout the entire process for optimal results. Use the Web Portal Manager or **pdadmin** utility. This stanza relates only to the **pdadmin** utility.

The stanza entries for turning on or off the setting for authorization checks for the delegated management of groups and users are in the `[delegated-admin]` stanza of the following configuration file:

- The `ivmgrd.conf` configuration file for the policy server

authorize-group-list

This stanza entry specifies whether authorization checks on the **group list** and **group list-dn** commands are made.

Syntax

```
authorize-group-list = {yes|no}
```

Description

Specification of whether authorization checks on the **group list** and **group list-dn** commands are made.

This keyword is provided as a performance feature.

Options

yes Enables authorization checks.

no Disables authorization checks.

Usage

Optional

Default value

no

Example

```
authorize-group-list = yes
```

[domains] and [domain=*domain_name*] stanzas

The [domains] stanza contains a list of domains.

Each domain specified under this stanza must have its own [domain=*domain_name*] stanza. The following example shows domains named d and mydomain:

```
[domains]
domain = d
domain = mydomain

[domain=d]

[domain=mydomain]
```

The stanza entries for configuring multiple domains are in the [domains] and the [domain=*domain_name*] stanzas of the following configuration file:
- The ivmgrd.conf configuration file for the policy server

allowed-registry-substrings

This stanza entry specifies the Distinguished Name (DN) substring that restricts which registry locations that users can be created in or be imported from.

Syntax

```
allowed-registry-substrings = dn
```

Description

Distinguished name (DN) substring that restricts which registry locations that users can be created in or be imported from.

The DN of the user that is created or imported must contain the substring value specified. The DN substring value restrictions are registry-dependent. Most user registries allow an alphanumeric string that is not case-sensitive. String values are expected to be characters that are part of the local code set.

You can specify one or more relative DNs to use when creating users. By specifying one or more substrings, you can restrict creating and importing users and groups to the relative DNs that are identified by the substrings. For example, you can specify the DN substring dc=mkt to restrict users who are created or imported into a domain named Marketing:

As a management domain administrator, complete the following tasks:

1. Manually add the *dn* value for each domain created, except the Management (policy server) domain.
2. Notify the domain administrator, after this key value pair is added, to add this string to the DN option when creating and importing users or groups.

Options

dn The distinguished name substring

Usage

Optional

Default value

There is no default value.

Example

```
allowed-registry-substrings = dc=mkt
```

database-path

This stanza entry specifies the file name and location of the policy database for the specified domain.

Syntax

```
database-path = fully_qualified_path
```

Description

File name and location of the policy database for the specified domain. The name of the database is the same as the domain name. The file extension can be anything, but the extension is usually `.db`.

Note: Do not edit this entry.

Options

The *fully_qualified_path* value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of valid characters in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). Windows operating systems path names, however, can have a backward slash (\) or a colon (:).

For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

Usage

Conditional. This stanza entry is required when the user creates at least one domain.

Default value

The following table shows the default value by platform.

Table 23. database-path default value by platform

Platform	File name
Linux or UNIX	`/var/PolicyDirector/db/`*`domain_name`*`.db`
Windows	`c:\program files\tivoli\policy director\db\`*`domain_name`*`.db`

Example

The following example shows the setting of the database path on a Windows operating system:

```
d:\programs\ibm\am\db\dname1.db
```

domain

This stanza entry specifies the name of the domain.

Syntax

```
domain = domain_name
```

Description

Name of the domain that was created.

Options

The *domain_name* value is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set.

Usage

Conditional. This stanza entry is required when the user creates at least one domain.

Default value

There is no default value.

Example

```
domain = mydomain1
```

[ivacld] stanza

The stanza entries for configuring authorization server-related information are in the `[ivacld]` stanza in the following configuration file:

- The *`[instance-]`*`ivacld.conf` configuration file for the authorization server

log-file

This stanza entry specifies the location and name of the log file.

Syntax

```
log-file = fully_qualified_path
```

Description

Location and name of the log file. Messages are redirected from STDOUT and STDERR and sent to the server log file as defined in the authorization server routing file (`pdacld_routing`). The authorization server relies on the routing file to determine the log file names and path.

At startup of the authorization server, a check is made to see whether the routing file exists. If it exists, the routing file is used and this stanza entry is ignored; otherwise, this stanza entry is used.

Options

The *fully_qualified_path* value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of valid characters in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). Windows operating systems path names, however, can have a backward slash (\) or a colon (:). For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

During installation of Security Access Manager, if you enabled Tivoli Common Directory to specify one common directory location for all your base component log files, the default installation directory is different. For example:

```
log-file = TCD/HPD/logs
/msg__pdacld_utf8.log
```

The three-character identifier used in the example is `HPD`, which specifies that the log files are for the Security Access Manager common components.

Usage

Required

Default value

The following table shows the default value by platform.

Table 24. log-file default value by platform

Platform	File name
Linux or UNIX	`/var/PolicyDirector/log/msg__pdacld_utf8.log`
Windows	`c:\program files\tivoli\policy director\log\ msg__pdacld_utf8.log`

Example

The following example sets the log file as Tivoli Common Directory on a Windows operating system:

```
log-file = C:\pd\log\msg__pdacld_utf8.log
```

The following example sets the log file as Tivoli Common Directory on a AIX, Linux, or Solaris operating system:

```
/PolicyDirector/TAMBase/HPD/logs/msg__pdacld_utf8.log
```

logcfg

This stanza entry enables logging and auditing for the authorization component.

Syntax

```
logcfg = audit.azn:{log-agent} path=path
     flush_interval=interval log_id
```

Description

Enables logging and auditing for the authorization component.

Each server provides its own event logging setting in its corresponding configuration file.

Options

audit.azn
Category that specifies auditing of the authorization component.

log-agent
Specifies that the destination where *log-agent* is one of the following values:
- stdout
- stderr
- file
- pipe
- remote

path = *path*
Specifies the name and location of the log file that is used for the *log-agent*.

flush_interval = *interval*
Specifies the frequency for flushing log file buffers.

log_id Specifies the identifier for directing events from additional categories to the same *log-agent*.

Remove the number signs (#) at the beginning of the configuration file lines to enable authentication or authorization auditing (or both) for the application.

Usage

Optional

Default value

There is no default value.

Example

The following example shows the configuration for authentication and authorization auditing:

```
logcfg = audit.azn:file path=/var/PolicyDirector/audit/
      ivacld.log,flush_interval=20,log_id=PDAclAudit
logcfg = audit.authn:file log_id=PDAclAudit
```

permit-unauth-remote-caller

This stanza entry specifies whether authorization API clients must be authorized by the authorization server before their requests are processed.

Syntax

```
permit-unauth-remote-caller= {true|false}
```

Description

Specification of whether authorization API clients must be authorized by the authorization server before their requests are processed.

Options

true Authorization API clients are not authorized.

Attention: Specifying `true` exposes the policy database in the domain for all clients to read, not just those clients that were properly authorized with membership in the `remote-acl-users` group. Depending on the policy within the domain security, system planners must consider the ability for any client to read system-defined policy to be a security problem.

false Authorization API clients are authorized.

Usage

Optional

Default value

```
false
```

Example

```
permit-unauth-remote-caller= false
```

pid-file

This stanza entry specifies the location and name of the PID file.

Syntax

pid-file = *fully_qualified_path*

Description

Location and name of the PID file.

Options

fully_qualified_path

Represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of valid characters in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

Usage

Required

Default value

The following table shows the default value by platform.

Table 25. pid-file default value by platform

Platform	File name
Linux or UNIX	`/var/PolicyDirector/log/ivacld.pid`
Windows	`c:\progra files\tivoli\policy director\log\ivacld.pid`

Example

Example for a Windows operating system:

```
pid-file = C:\pd\log\ivacld.pid
```

tcp-req-port

This stanza entry specifies the Transmission Control Protocol (TCP) port on which the server is listening for requests.

Syntax

```
tcp-req-port = {0|port}
```

Description

Transmission Control Protocol (TCP) port on which the server is listening for requests.

Options

0 Disable the port number.

port Enable the port number. Use any valid port number. A valid port number is any positive number that is allowed by TCP/IP and that is not currently being used by another application. Use the default value, or use a port over 1000 that is not currently being used.

Usage

Required

Default value

`7136`

Example

```
tcp-req-port = 7136
```

unix-user

This stanza entry specifies the Linux or UNIX user account for this server.

Syntax

```
unix-user = user_name
```

Description

The Linux or UNIX user account for this server. The server will run as this user account.

Options

The *user_name* value represents an alphabetic string for the name associated with the user account.

Usage

Required

Default value

```
ivmgr
```

Example

```
unix-user = ivmgr
```

unix-group

This stanza entry specifies the Linux or UNIX group account for this server.

Syntax

```
unix-group = group_name
```

Description

The Linux or UNIX group account for this server. The server runs as this account.

Options

The *group_name* value represents an alphabetic string for the group associated with the user account.

Usage

Required

Default value

```
ivmgr
```

Example

```
unix-group = ivmgr
```

[ivmgrd] stanza

The stanza entries for configuring the policy server and policy database are in the `[ivmgrd]` stanza in this configuration file:

- The `ivmgrd.conf` configuration file for the policy server

provide-last-login

This stanza entry specifies whether to report information about the last login instance of a user.

Syntax

```
provide-last-login = {yes|true|no|false}
```

Description

Use the `provide-last-login option` for reporting information about the last login instance of a user.

To record the last login information for LDAP-based registries, set `[ldap] enable-last-login` to `yes`.

For Microsoft Active Directory registry, Security Access Manager uses the Active Directory user attribute `lastLogonTimestamp` to report the last login time of the user. This attribute is a system attribute and is updated automatically by Active Directory. Security Access Manager has no control over this attribute except reporting the value when required. This attribute is not updated every time a user logs in successfully. When a user logs in successfully, this attribute is only updated if its current value is older than the current time minus the value of the `msDS-LogonTimeSyncInterval` attribute.

The value that Security Access Manager reports for the last login of a user might not be the exact time that a user last logged in. The reported time might be the actual last login time minus the configurable value of `msDS-LogonTimeSyncInterval`. You can configure the default value of `msDS-LogonTimeSyncInterval` to suit the user domain policy.

To use the `lastLogonTimestamp` attribute, the Active Directory domains must be at or greater than Microsoft Windows 2003 domain functional level. For more information about `lastLogonTimestamp` and `msDS-LogonTimeSyncInterval`, visit the Microsoft support website.

Options

yes | true
Set the **provide-last-login** option to **yes**, to specify that the policy server reports the time of last login of a user.

no | false
Set the **provide-last-login** option to **no**, to disable reporting of the last login information about a user.

provide-last-pwd-change

This stanza entry specifies whether to report information about the last password change instance of a user.

Syntax

```
provide-last-pwd-change = {yes|true|no|false}
```

Description

Use the **provide-last-pwd-change** option to permit reporting of information about the last password change instance of a user.

Options

yes | true
: Set the **provide-last-pwd-change** option to **yes**, to specify that the policy server reports the last password change instance of a user.

no | false
: Set the **provide-last-pwd-change** option to **no**, to disable reporting of the last password change instance of a user.

auto-database-update-notify

This stanza entry specifies automatic or manual update notification for policy database replicas.

Syntax

```
auto-database-update-notify = {yes|true|no|false}
```

Description

Specification of automatic or manual update notification for policy database replicas.

Options

yes | true
: Enable automatic update notification. This automatic setting is appropriate for environments where database changes are few and infrequent. When you configure update notification to be automatic, you must also correctly configure the `max-notifier-threads=` and `notifier-wait-time=` stanza entries.

no | false
: Enable manual update notification.

Usage

Required

Default value

`yes`

Example

```
auto-database-update-notify = yes
```

ca-cert-download-enabled

This stanza entry specifies whether the CA certificate can be downloaded.

Syntax

```
ca-cert-download-enabled = {yes|no}
```

Description

The policy server always allows the download of the CA certificate. It is up to the client application to allow whether the CA certificate can be downloaded. Independent of the defined value, the policy server ignores this configuration setting.

Usage

Ignored

database-path

This stanza entry specifies the location and name of the master policy database.

Syntax

```
database-path = fully_qualified_path
```

Description

Location and name of the master policy database. The file extension can be anything, but the extension is usually `.db`.

Note: Do not edit this stanza entry.

Options

fully_qualified_path
Represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of valid characters in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). Windows operating systems path names, however, can have a backward slash (\) or a colon (:). For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

Usage

Required

Default value

The following table shows the default value by platform.

Platform	File name
Linux or UNIX	`/var/PolicyDirector/db/master_authzn.db`
Windows	`c:\program files\tivoli\policy director\db\master_authzn.db`

Example

The following example set the path to the master policy database on a Windows operating system:

```
database-path = C:\pd\db\master_authzn.db
```

log-file

This stanza entry specifies the location and name of the log file.

Syntax

```
log-file = fully_qualified_path
```

Description

Location and name of the log file. Messages are redirected from STDOUT and STDERR and sent to the server log file as defined in the policy server routing file (*[instance-]*pdmgrd_routing). The policy server relies on the routing file to determine the log file names and path.

At startup of the policy server, a check is made to see whether the routing file exists. If it exists, the routing file is used and this stanza entry is ignored; otherwise, this stanza entry is used.

Options

The *fully_qualified_path* value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of valid characters in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). Windows operating systems path names, however, can have a backward slash (\) or a colon (:).

For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

Usage

Required

Default value

The following table shows the default value by platform.

Table 26. [ivmgrd] stanza log-file default value by platform

Platform	File name
Linux or UNIX	/var/PolicyDirector/log/msg__pdmgrd_utf8.log
Windows	c:\program files\tivoli\policy director\log\ msg__pdmgrd_utf8.log

During an installation of Security Access Manager, if you enabled Tivoli Common Directory to specify one common directory location for all your log files, the default installation directory is different.

Example

The following example sets the log file on a Windows operating system without Tivoli Common Directory:

```
log-file = C:\pd\log\msg__pdmgrd_utf8.log
```

The following example sets the log file on a AIX, Linux, or Solaris operating system with Tivoli Common Directory:

```
log-file = TCD_directory/HPD/logs/msg__pdmgrd_utf8.log
```

The three-character identifier used in the example is `HPD`, which specifies that the log files are for Security Access Manager.

logcfg

This stanza entry enables logging and auditing for the application.

Syntax

```
logcfg = audit.azn:{log-agent} path=path
     flush_interval=interval log_id
```

Description

Enables logging and auditing for the application. Category, destination, and other parameters are used to capture Security Access Manager auditing and logging events.

Each server provides its own event logging setting in its corresponding configuration file.

Options

audit.azn
Category that specifies auditing of the authorization component.

log-agent
Specifies that the destination where *log-agent* is one of the following values:

- `stdout`
- `stderr`
- `file`
- `pipe`
- `remote`

path = *path*
Specifies the name and location of the log file that is used for the *log-agent*.

flush_interval = *interval*
Specifies the frequency for flushing log file buffers.

log_id
Specifies the identifier for directing events from additional categories to the same *log-agent*.

Remove the number signs (#) at the beginning of the configuration file lines to enable authentication or authorization auditing (or both) for the application.

Usage

Optional

Default value

There is no default value.

Example

The following example sets the configuration for authentication and authorization auditing only:

```
logcfg = audit.azn:file path=/var/PolicyDirector/audit/
     pdmgrd.log,flush_interval=20,log_id=PDMgrAudit
logcfg = audit.authn:file log_id=PDMgrAudit
#logcfg = audit.mgmt:file log_id=PDMgrAudit
```

max-notifier-threads

This stanza entry specifies the maximum number of event notifier threads.

Syntax

```
max-notifier-threads = number_threads
```

Description

Maximum number of event notifier threads. The policy server is responsible for synchronizing all database replicas in the secure domain. When a change is made to the master database, notification threads do the work of announcing this change to all replicas. Each replica then has the responsibility to download the new information from the master.

When the update notification stanza entry is set to `yes`, you must correctly configure this stanza entry and also the `notifier-wait-time=` stanza entry.

Options

number_threads
: Set this value to equal the number of existing replicas. Specify a valid, positive whole number. Valid range for the number of threads is 1 - 128.

Usage

Conditional. This stanza entry is required when `auto-database-update-notify = yes`.

Default value

`10`

Example

```
max-notifier-threads = 20
```

notifier-wait-time

This stanza entry specifies the time in seconds that the authorization policy database is idle before notification is sent to replicas.

Syntax

```
notifier-wait-time = time_seconds
```

Description

Time in seconds that the authorization policy database is idle before notification is sent to replicas. When the policy server is instructed to change the master policy database, it waits for a default period before sending out notifications to database replicas. This time delay is reset with each subsequent change to the database.

When the update notification stanza entry is set to `yes`, you must correctly configure this stanza entry and also the `max-notifier-threads=` stanza entry.

Options

time_seconds
: The number of seconds the authorization policy database is idle before notification is sent to replicas.

Usage

Conditional. This stanza entry is required when the `auto-database-update-notify = yes`.

Default value

```
15
```

Example

```
notifier-wait-time = 30
```

pid-file

This stanza entry specifies the location and name of the PID file.

Syntax

```
pid-file = fully_qualified_path
```

Description

Location and name of the PID file.

Options

fully_qualified_path
: Represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of valid characters in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). Windows operating systems path names, however, can have a backward slash (\) or a colon (:).

  For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

Usage

Required

Default value

The following table shows the default value by platform.

Table 27. [ivmgrd] stanza pid-file default value by platform

Platform	File name
Linux or UNIX	/var/PolicyDirector/log/ivmgrd.pid
Windows	c:\program files\tivoli\policy director\log\ivmgrd.pid

Example

Example for a AIX, Linux, or Solaris operating system:

```
pid-file = /var/PolicyDirector/log/ivmgrd.pid
```

standby

This stanza entry specifies the number of standby policy servers.

Syntax

```
standby = {0|number}
```

Description

Specifies the number of standby policy servers.

Note: The number of standby servers is generated and set by the configuration utility. Do not edit this stanza entry.

Options

0 Zero specifies that no policy servers are standby servers.

number
The number of standby policy servers. Use a number that is a positive whole number. Currently, this number is only 1.

Usage

Required

Default value

0

Example

```
standby = 1
```

tcp-req-port

This stanza entry specifies the TCP port on which the server is listening for requests.

Syntax

```
tcp-req-port = {0|port}
```

Description

TCP port on which the server is listening for requests.

Options

0 Disables the port number.

port Enables the port number. Specify any valid port number. A valid port number is any positive number that is allowed by TCP/IP and that is not currently being used by another application. Use the default port number, or use a port number over 1024 that is currently not being used.

Usage

Required

Default value

8135

Example

```
tcp-req-port = 8135
```

unix-user

This stanza entry specifies the Linux or UNIX user account for this server.

Syntax

```
unix-user = user_name
```

Description

The Linux or UNIX user account for this server. The server runs as this account.

Options

user_name
Represents an alphabetic string for the name associated with the user account.

Usage

Required

Default value

ivmgr

Example

```
unix-user = ivmgr
```

unix-group

This stanza entry specifies the Linux or UNIX group account for this server.

Syntax

```
unix-group = group_name
```

Description

The Linux or UNIX group account for this server. The server runs as this account.

Options

group_name
: Represents an alphabetic string for the group associated with the user account.

Usage

Required

Default value

```
ivmgr
```

Example

```
unix-group = ivmgr
```

am610compat

This stanza entry maintains compatibility with applications developed for previous Security Access Manager versions.

Syntax

```
am610compat = yes
```

Description

Maintains compatibility with applications developed for previous Security Access Manager versions. The policy server causes the system to follow the previous behavior of applications that assume no errors are thrown.

Options

yes If set to yes, the command does not return an error if the specified user does not exist. The system returns the value `none` instead.

no If set to no, the command returns an error if the user does not exist.

Usage

The `am610compat` configuration item is for the **`ivmgrd.conf`** policy server configuration file.

Th configuration item affects the **`pdadmin`** commands where the **`-user`** <*user-name*> is supplied.

See the *pdadmin commands policy get section* of the IBM® Security Access Manager Command Reference Guide for details.

If the user specified by the command does not exist, it returns the value `none`.

Default value

yes for migrated systems

no for new installations

Example

```
am610compat = yes
```

[ldap] stanza

This stanza defines configuration key value pairs that are required for the Security Access Manager servers to communicate with the server that is associated with an LDAP user registry.

The value for the user registry stanza entry (ldap-server-config) is determined by the pd.conf file. The pd.conf file is created when the IBM Security Access Manager runtime component is configured.

The key value pairs that are used only for the LDAP registry server are in the ldap.conf configuration file in the [ldap] stanza. The LDAP server stanza entries are described separately in "[ldap] stanza for ldap.conf" on page 253.

The key value pairs that are for the server configuration files are in the [ldap] stanza of each of the following configuration files:

- The ivmgrd.conf configuration file for the policy server
- The *[instance-]*ivacld.conf configuration file for the authorization server
- Your resource managers' configuration file for configured LDAP entries

  The aznAPI.conf configuration file is provided with Security Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins typically provide the standard functions. Before you implement service plug-ins, read and thoroughly understand the concepts in the *Authorization C API Developer Reference*.

enhanced-pwd-policy

This stanza entry specifies whether the LDAP registries that Security Access Manager uses provide password policy enforcement for LDAP accounts.

Syntax

```
enhanced-pwd-policy = {yes|true|no|false}
```

Description

Note: The appliance embedded LDAP server does not support this configuration option.

Security Access Manager uses LDAP account passwords for authentication. This means that Security Access Manager is subject to LDAP registry password policies. When the enhanced-pwd-policy option is enabled, Security Access Manager efficiently identifies the underlying LDAP registry password policy and reacts appropriately. The Security Access Manager password policy is enforced concurrently and is not affected by the enhanced-pwd-policy option.

This option is supported for Sun Directory Server 6.3.1 and Security Directory Server. For Security Directory Server with the `enhanced-pwd-policy` option disabled, Security Access Manager provides only limited support for handling LDAP registry password policies. The `enhanced-pwd-policy` option enhances such support.

When you set the `auth-using-compare` option to `no`, a user password is authenticated by creating a connection to the LDAP registry and binding the connection with the user password. Success or failure of the binding is noted and the connection is closed. If you set the `enhanced-pwd-policy` option is set to `yes` when `auth-using-compare` is set to `no`, the user password changes occur on the connection that is used to authenticate the user.

Such behavior increases the duration of the connection and might cause the number of simultaneous instances to increase. If the increase in simultaneous connections is not acceptable, use the `max-auth-connections` option to limit the number of simultaneous connections. For detailed information about the `max-auth-connections` option, see the **max-auth-connections** section.

Note: Only Security Directory Server supports enabling of the `auth-using-compare` option. For other LDAP servers, Security Access Manager considers this option disabled.

Security Access Manager WebSEAL takes advantage of `enhanced-pwd-policy`.

The password policies and account states supported by Security Access Manager are:

- Password reset
- Locked accounts
- Expired accounts
- Grace login for expired accounts
- Accounts whose passwords are going to expire

Options

yes|true

When the `enhanced-pwd-policy` option is set to `true`, Security Access Manager efficiently identifies the underlying LDAP registry password policy and reacts appropriately.

no|false

When the `enhanced-pwd-policy` option is set to `false`, the behavior of Security Access Manager towards LDAP registry password policy enforcement remains unchanged.

Default value

The default value of `enhanced-pwd-policy` is `no|false`

Example

An example of this feature is: LDAP reports that an account is expired and allows grace login. The user is informed that the account is expired, and is provided a grace login page and an option to change the password.

Using enhanced-pwd-policy with Security Directory Server

If you enable enhanced-pwd-policy for the Security Directory Server when using Security Directory Server for the registry, you must take several steps.

About this task

- Manually update the access control lists (ACL) of the server so that users can change their passwords.
- Set auth-using-compare to no in each configuration file where you set enhanced-pwd-policy to yes.

To ensure that users can change their passwords, suffixes that contain or will contain Security Access Manager user accounts must have an LDAP ACL that permits users to change their passwords. An example of the suffix that you create is o=ibm,c=us. An example of a suffix that Security Access Manager creates is secAuthority=Default. Each of these suffixes requires an LDAP ACL to let the users change their passwords.

Complete the following steps to update LDAP access control lists:

Procedure

1. For the suffix that you created, create a file, for example, addacl1.ldif, that contains the following statements:

```
dn:o=ibm,c=us
changetype:modify
add:aclEntry
aclEntry:access-id:cn=this:at.userPassword:rwsc
```

2. Run the command:

```
idsldapmodify -D "cn=root" -w "password"
 -h your.ldap.host.name -f "addacl1.ldif"
```

Behavior of Security Access Manager policy server LDAP accounts and policies:

The pwdMustChange option in the LDAP policy prevents the policy server from starting during configuration.

The account used for configuration does not exist before the configuration starts. You cannot set a policy to override the global policy. To create Security Access Manager LDAP server accounts, you might temporarily disable the global policies before configuration.

After you configure the Security Access Manager server LDAP accounts and the policy server, set a policy for each Security Access Manager server LDAP account to override any global policy that affects the use of the LDAP account.

max-auth-connections

This stanza entry specifies how many simultaneous connections to your LDAP server are permitted for user authentication.

Syntax

```
max-auth-connections = {0|unlimited number of simultaneous connections used for user
authentication|any number higher than 0|actual number of simultaneous
connections used for user authentication}
```

Description

Use the max-auth-connections option to determine how many simultaneous connections to your LDAP server are permitted for user authentication. This option has no effect if auth-using-compare is enabled. The benefit of the max-auth-connections option is greater if enhanced-pwd-policy is enabled. See **enhanced-pwd-policy** for details.

Options

0|unlimited number of simultaneous connections used for user authentication
When you set max-auth-connections to 0 (zero), you can use unlimited LDAP server connections simultaneously to authenticate users.

any number higher than 0|actual number of simultaneous connections used for user authentication
If you set max-auth-connections to a value greater than 0 (zero), the number of simultaneous connections used for user authentication is limited to the number you specify.

Default value

By default, max-auth-connections is set to 0 (zero).

enable-last-login

This stanza entry specifies whether to store the last login information of a user in LDAP.

Syntax

```
enable-last-login = {yes|true|no|false}
```

Description

For LDAP-based registries, each Security Access Manager server that provides a login service requires you to set the enable-last-login option to store the last login date of a user in LDAP. Examples of such servers are:

- WebSEAL.
- Policy Server.
- Authorization Server.
- Local-mode authorization and Java Authorization applications that allow user authentication directly to the registry.

Options

yes|true
Set the value of the enable-last-login option to yes if you want the last login information of users to be recorded and displayed to users.

no|false
Set the value of the enable-last-login option to no if you do not want the last login information of users to be recorded and displayed to users.

auth-using-compare

This stanza entry specifies whether ldap_compare() is used instead of the ldap_bind() call to verify the password and authenticate the user.

Syntax

```
auth-using-compare = {yes|true|no|false}
```

Description

Choice of whether `ldap_compare()` is used instead of the `ldap_bind()` call to verify the password and authenticate the user. For those LDAP servers that allow it, a compare operation might run faster than a bind operation. The value for each server can be different, depending on how that server is configured.

This option changes the method used by the following authorization API calls:
- `azn_util_client_authenticate()`
- `azn_util_password_authenticate()`

Options

yes|true
A compare operation is used to authenticate LDAP users.

no|false
A bind operation is used to authenticate LDAP users.

Any value other than `yes|true`, including a blank value, is interpreted as `no|false`.

To use this key value pair for performance tuning, see the *IBM Security Access Manager for Web: Performance Tuning Guide*.

Usage

Optional

Default value

The default values are server-dependent.

Example

```
auth-using-compare = yes
```

authn-timeout

This stanza entry specifies the amount of time (in seconds) for authentication operations before the LDAP server is considered down.

Syntax

```
authn-timeout = {0|number_seconds}
```

Description

Amount of time (in seconds) that is allowed for authentication operations before the LDAP server is considered to be down. If specified, this value overrides any value set for the `timeout` entry for authentication operations.

Note: Do not specify this stanza entry in the `ldap.conf` server configuration file.

Options

0 No timeout (synchronous).

number_seconds
The number of seconds allowed for authentication operations. Specify a positive whole number. There is no range limitation for timeout values.

Usage

Optional

Default value

```
0
```

Example

```
authn-timeout = 0
```

bind-dn

This stanza entry specifies the LDAP user distinguished name (DN) that is used when binding (signing on) to the LDAP server.

Syntax

```
bind-dn = LDAP_dn
```

Description

LDAP user distinguished name (DN) that is used when binding (signing on) to the LDAP server. The *LDAP_dn* value is created, based on the server name that was specified with the **–n** *server_name* option and the local host of the computer.

Use the **svrsslcfg** utility to set the *LDAP_dn* value.

To use this key value pair for performance tuning, see the *IBM Security Access Manager for Web: Performance Tuning Guide*.

Options

LDAP_dn
Distinguished name that is used to bind to the LDAP server

Usage

Conditional. This stanza entry is required when using an LDAP user registry.

Default value

The default value is server-dependent.

Example

The following example sets the distinguished name for the policy server:

```
bind-dn = cn=ivmgrd/master,cn=SecurityDaemons,secAuthority=Default
```

cache-enabled

This stanza entry specifies whether LDAP client-side caching is used to improve performance for similar LDAP queries.

Syntax

```
cache-enabled = {yes|true|no|false}
```

Description

Specification of whether LDAP client-side caching is used to improve performance for similar LDAP queries.

Options

yes|true
Enables LDAP client-side caching.

no|false
Disables LDAP client-side caching. This value is the default value. Anything other than `yes|true`, including a blank value, is interpreted as `no|false`.

To use this key value pair for performance tuning, see the *IBM Security Access Manager for Web: Performance Tuning Guide*.

Usage

Optional

Default value

`no`

Example

```
cache-enabled = no
```

cache-group-expire-time

This stanza entry specifies the amount of time (in seconds) until a group entry in the cache is considered stale and is discarded.

Syntax

cache-group-expire-time = *number_seconds*

Description

Amount of time (in seconds) until a group entry in the cache is considered stale and is discarded. This stanza entry is ignored if the cache is not enabled.

Options

number_seconds
The amount of time specified in seconds. Specify a positive whole number.

Usage

Optional

Default value

`300` (5 minutes)

Example

```
cache-group-expire-time = 600
```

cache-group-membership

This stanza entry specifies whether group membership information is cached.

Syntax

```
cache-group-membership = {yes|true|no|false}
```

Description

Specification of whether group membership information is cached. This stanza entry is ignored if the cache is not enabled.

Options

yes|true
: Group membership is cached.

no|false
: Group membership is not cached. Anything other than `yes|true`, including a blank value, is interpreted as `no|false`.

Usage

Optional

Default value

`yes|true`

Example

```
cache-group-membership = no
```

cache-group-size

This stanza entry specifies the number of entries in the LDAP group cache.

Syntax

```
cache-group-size = group_entries
```

Description

Number of entries in the LDAP group cache. This stanza entry is ignored if the cache is not enabled.

Options

group_entries
: A positive whole number that represents the number of entries is the LDAP group cache.

Usage

Optional

Default value

```
64
```

Example

```
cache-group-size = 100
```

cache-policy-expire-time

This stanza entry specifies the amount of time in seconds until a policy entry in the cache is considered stale and is discarded.

Syntax

cache-policy-expire-time = *number_seconds*

Description

Amount of time in seconds until a policy entry in the cache is considered stale and is discarded. This stanza entry is ignored if the cache is not enabled.

Options

number_seconds
: The amount of time specified in number of seconds. Specify a positive whole number.

Usage

Optional

Default value

```
30
```

Example

```
cache-policy-expire-time = 60
```

cache-policy-size

This stanza entry specifies the number of entries in the LDAP policy cache.

Syntax

cache-policy-size = *policy_entries*

Description

Number of entries in the LDAP policy cache. This stanza entry is ignored if the cache is not enabled.

Options

policy_entries
: A positive whole number that represents the number of entries is the LDAP policy cache.

Usage

Optional

Default value

```
20
```

Example

```
cache-policy-size = 50
```

cache-return-registry-id

This stanza entry specifies whether the LDAP cache returns the Security Access Manager user identity as it is stored in the registry or the value entered by the user.

Syntax

```
cache-return-registry-id = {yes|no}
```

Description

Specifies whether the LDAP cache returns the Security Access Manager user identity as it is stored in the registry or the value entered by the user.

Note: Refer to Appendix A, “Guidelines for changing configuration files,” on page 185 for guidelines on changing configuration file properties.

Options

yes Return the Security Access Manager user identity as it is stored in the registry. This option returns the user identity exactly as it was created and preserved in the registry.

no Return the Security Access Manager user identity as the value is entered by the user.

Usage

Optional

Default value

```
no
```

Example

```
cache-return-registry-id = no
```

cache-use-user-cache

This stanza entry specifies whether to use the user cache information.

Syntax

```
cache-use-user-cache = {yes|true|no|false}
```

Description

Specification of whether to use the user cache information. This stanza entry is ignored if the cache is not enabled.

Options

yes|true
Use user information from the cache.

no|false
Do not use user information from the cache. Anything other than `yes|true`, including a blank value, is interpreted as `no|false`.

Usage

Optional

Default value

`yes|true`

Example

```
cache-use-user-cache = no
```

cache-user-expire-time

This stanza entry specifies the amount of time in seconds until a user entry in the cache is considered stale and is discarded.

Syntax

```
cache-user-expire-time = number_seconds
```

Description

Amount of time in seconds until a user entry in the cache is considered stale and is discarded. This stanza entry is ignored if the cache is not enabled.

Options

number_seconds
The amount of time specified in number of seconds. Use a number that is a positive whole number.

Usage

Optional

Default value

`30`

Example

```
cache-user-expire-time = 120
```

cache-user-size

This stanza entry specifies the number of entries in the LDAP user cache.

Syntax

```
cache-user-size = user_entries
```

Description

Number of entries in the LDAP user cache. This stanza entry is Ignored if the cache is not enabled.

Options

user_entries
A positive whole number that represents the number of entries is the LDAP user cache.

Usage

Optional

Default value

```
256
```

Example

```
cache-user-size = 1000
```

default-policy-override-support

This stanza entry specifies whether user-level policy support can be used.

Syntax

```
default-policy-override-support = {yes|true|no|false}
```

Description

Specification of whether user-level policy support can be used.

Options

yes | true
User policy support is disabled, and only the global (default) policy is checked. This option allows the user policy to not be checked, even if it is specified.

no | false
User policy support is enabled. When a user policy is specified by the administrator, it overrides the global policy. If no value is specified, `default-policy-override-support = no` becomes the value.

To use this key value pair for performance tuning, see the *IBM Security Access Manager for Web: Performance Tuning Guide*.

Usage

Optional

Default value

no

Example

```
default-policy-override-support = yes
```

ldap-server-config

This stanza entry specifies the location of the `ldap.conf` configuration file.

Syntax

```
ldap-server-config = fully_qualified_path
```

Description

Location of the `ldap.conf` configuration file.

Note: When the `ldap-server-config` entry is specified in the configuration file, the values for `enabled`, `host`, `port`, `max-search-size`, and `replica` are obtained from the `ldap.conf` file. If any of these entries exist in the configuration file, their values are overridden by the values from the `ldap.conf` file.

Options

fully_qualified_path

Represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of valid characters in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). Windows operating systems path names, however, can have a backward slash (\) or a colon (:).

For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

Usage

This stanza entry is required for `ivmgrd.conf`.

Default value

The following table shows the default value by platform.

Platform	File name
Linux or UNIX	`/opt/PolicyDirector/etc/ldap.conf`
Windows	`c:\Program Files\tivoli\Policy Director\etc\ldap.conf`

Example

The following example set the location of the LDAP server for a AIX, Linux, or Solaris operating system:

```
ldap-server-config = /opt/PolicyDirector/etc/ldap.conf
```

login-failures-persistent

This stanza entry specifies whether the tracking of login failures is persistent (maintained in the registry) or done in the local process cache.

Syntax

```
login-failures-persistent = {yes|no}
```

Description

Specifies whether the tracking of login failures is persistent (maintained in the registry) or done in the local process cache.

Options

yes Tracking of failures is maintained in the registry.

no Tracking of failures is done in the local process cache.

Usage

Optional

Default value

`no`

Example

```
login-failures-presistent = yes
```

max-search-size

This stanza entry specifies the maximum search size, as the number of entries, that can be returned from the LDAP server.

Syntax

`max-search-size = [0|`*`number_entries`*`]`

Description

Limit for the maximum search size, specified as the number of entries, that can be returned from the LDAP server. The value for each server can be different, depending on how the server was configured.

Options

`0` The number is unlimited. There is no limit to the maximum search size.

number_entries
The maximum number of entries for search, specified as an integer whole number. This value can also be limited by the LDAP server itself.

Usage

Optional

Default value

The default value is server-dependent, but defaults to 2048 if not configured.

Example

```
max-search-size = 2048
```

port

This stanza entry specifies the non-SSL IP port number that is used for communicating with the LDAP server.

Syntax

```
port = port
```

Description

Non-SSL IP port number that is used for communicating with the LDAP server.

Options

port The port number configured for the LDAP server.

Usage

Required for the authorization server; not required for the policy server.

Default value

389

Example

```
port = 389
```

prefer-readwrite-server

This stanza entry specifies whether the client can question the read/write LDAP server before querying any replica read-only servers that are configured in the domain.

Syntax

```
prefer-readwrite-server = {yes|true|no|false}
```

Description

Specification of whether the client can question the read/write LDAP server before querying any replica Read-only servers that are configured in the domain.

The default value can be different. For example, the default value for ivmgrd.conf is yes while the default value for ivacld.conf is no.

Options

yes|true
Enables the client to be able to question the read/write LDAP server.

no|false
Disables the client. Anything other than `yes|true`, including a blank value, is interpreted as `no|false`.

Usage

Optional

Default value

There is no default value. The default value is server-dependent.

Example

```
prefer-readwrite-server = no
```

search-timeout

This stanza entry specifies the amount of time in seconds that is allowed for search operations before the LDAP server is considered to be down.

Syntax

```
search-timeout = {0|number_seconds}
```

Description

Amount of time in seconds that is allowed for search operations before the LDAP server is considered to be down. If specified, this value overrides any value that is set for the `timeout` entry for search operations.

Note: Do not specify this stanza entry in the `ldap.conf` server configuration file.

Options

0 No timeout (synchronous).

number_seconds
The number of seconds allowed for search operations. Specify a positive whole number. There is no range limitation for timeout values.

Usage

Optional

Default value

0

Example

```
search-timeout = 0
```

ssl-enabled

This stanza entry specifies whether the Security Access Manager server uses SSL to communicate with the LDAP server.

Syntax

```
ssl-enabled = {yes|true|no|false}
```

Description

Specification of whether the Security Access Manager server uses SSL to communicate with the LDAP server. The value for each Security Access Manager server can be different, depending on how that server was configured. If this value is set to `yes` and Federal Information Processing Standards (FIPS) mode is enabled (`ssl-compliance=yes`), LDAP uses whatever secure communication protocol it chooses for FIPS enablement.

If you specify that the authorization API (aznAPI) should use SSL to communicate with the LDAP server, you must enable SSL using this stanza entry.

If you enable SSL communication, you must specify an SSL key file name and, if there are multiple keys in the file, the key file DN.

Options

yes | true
: Enables SSL communication.

no | false
: Disables SSL communication. Anything other than `yes` or `true`, including a blank value, is interpreted as `no` or `false`.

Usage

Required to enable SSL communication. When `ssl-enabled = yes`, the `LdapSSL` entry in the `ldap.conf` file must be set to `useSSL`.

Default value

There is no default value. The default values are server-dependent.

Example

```
ssl-enabled = yes
```

ssl-keyfile

This stanza entry specifies the SSL key file name and location.

Syntax

```
ssl-keyfile = ldap-ssl-key-filename
```

Description

SSL key file name and location. Use the SSL key file to handle certificates that are used in LDAP communication. The file extension can be anything, but the extension is usually `.kdb`.

The certificate files in a directory need to be accessible to the server user (or all users). Make sure that server user (for example, `ivmgr`) or all users have permission to access the `.kdb` file and the folder that contains the `.kdb` file.

Options

ldap-ssl-key-filename
: A valid file name is an alphanumeric string that is not case-sensitive. String values are expected to be characters that are part of the local code set. For

Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks. Windows operating systems path names, however, can have a backward slash (\) or a colon (:). For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

Usage

Conditional. This stanza entry is required only when the LDAP server is configured to do client authentication (`ssl-enabled = yes`).

Default value

The following table shows the default value by platform.

Table 28. [ldap] stanza ssl-keyfile default value by platform

Platform	**File name**
Linux or UNIX	`/opt/PolicyDirector/keytab/`*`server_name`*`.kdb`
Windows	`c:\program files\tivoli\policy director\keytab\`*`server_name`*`.kdb`

Example

The following example sets the SSL key file for a UNIX policy server:

```
ssl-keyfile = /ldap52kdb/a17jsun.kdb
```

ssl-keyfile-dn

This stanza entry specifies the key label of the client certificate within the SSL key file.

Syntax

```
ssl-keyfile-dn = ldap-ssl-keyfile-label
```

Description

Key label of the client certificate within the SSL key file.

Options

ldap-ssl-keyfile-label
    Identifies the client certificate that is presented to the LDAP server.

Usage

Conditional. This stanza entry is required only when the LDAP server is configured to do client authentication.

Default value

If the default policy server key database is being used, the default client certificate value is `PDLDAP`.

Example

```
ssl-keyfile-dn = "PDLDAP"
```

ssl-keyfile-pwd

This stanza entry is deprecated. The `ssl-keyfile-pwd` entry is deprecated in the `[ldap]` stanza. Although this entry might exist in a configuration file, it is ignored.

Syntax

```
ssl-keyfile-pwd = ldap-ssl-keyfile-password
```

Description

Deprecated: The `ssl-keyfile-pwd` entry is deprecated in the [ldap] stanza. Although this entry might exist in a configuration file, it is ignored.

user-and-group-in-same-suffix

This stanza entry specifies whether the groups in which a user is a member are defined in the same LDAP suffix as the user definition.

Syntax

```
user-and-group-in-same-suffix = {yes|true|no|false}
```

Description

Specification of whether the groups in which a user is a member are defined in the same LDAP suffix as the user definition.

When a user is authenticated, the groups in which the user is a member must be determined to build a credential. Normally, all LDAP suffixes are searched to locate the groups of which the user is a member.

Options

yes|true
The groups that are assumed to be defined in the same LDAP suffix as the user definition. Only that suffix is searched for group membership. This behavior can improve the performance of group lookup, because only a single suffix is searched. Use this option only if group definitions are restricted to the same suffix as user definitions.

no|false
The groups might be defined in any LDAP suffix. Anything other than `yes|true`, including a blank value, is interpreted as `no|false`.

To use this key value pair for performance tuning purposes, see the *IBM Security Access Manager for Web: Performance Tuning Guide*.

Usage

Optional

Default value

`no`

Example

```
user-and-group-in-same-suffix = yes
```

[ldap] stanza for ldap.conf

This stanza defines the configuration key value pairs that are required for the LDAP server. For example, you can find the configuration keys and values for LDAP failover, including the use of master server and replica servers, in this stanza.

The user registry type value is determined by the `pd.conf` file. The `pd.conf` file is created when the IBM Security Access Manager runtime is configured.

To use the key value pairs in this stanza for performance tuning, see the *IBM Security Access Manager for Web: Performance Tuning Guide*.

cache-enabled

This stanza entry specifies whether LDAP client-side caching is used to improve performance for similar LDAP queries.

Syntax

```
cache-enabled = {yes|true|no|false}
```

Description

Specification of whether LDAP client-side caching is used to improve performance for similar LDAP queries.

Options

yes|true
Enables LDAP client-side caching.

no|false
Disables LDAP client-side caching. This value is the default value. Anything other than `yes|true`, including a blank value, is interpreted as `no|false`.

To use this key value pair for performance tuning purposes, see the *IBM Security Access Manager for Web: Performance Tuning Guide*.

Usage

Optional

Default value

`no`

Example

```
cache-enabled = no
```

cache-account-policy

The cache-account-policy is a configuration file parameter in the `[ldap]` stanza for `ldap.conf`. The policy determines whether the LDAP client-side caching is enabled.

Syntax

```
cache-account-policy = {yes|true|no|false}
```

Default value

yes

Option descriptions

yes|true
Enables LDAP client-side caching of expired account or expired password. Anything other than `no|false`, including a blank value, is interpreted as `yes|true`.

no|false
Disables LDAP client-side caching of expired account or password.

Usage

Optional. Enable this configuration file parameter if you want Security Access Manager to cache an expired account or password. Otherwise, set this parameter to no.

Example

This example enables the LDAP caching of expired account or password.

```
cache-account-policy = yes
```

connection-inactivity

This stanza entry specifies the number of seconds of inactivity allowed on an LDAP connection before the connection is taken down.

Syntax

connection-inactivity = *number_seconds*

Description

Specifies the number of seconds of inactivity allowed on an LDAP connection before the connection is taken down.

This parameter is not available with the pdconfig utility. The parameter must be modified manually with the **pdadmin** command line (local login). For more information, see "pdadmin commands" in the *IBM Security Access Manager for Web: Command Reference*.

Options

number_seconds
The number of seconds of inactivity allowed on an LDAP connection. The valid range for this parameter is `0` to `31536000`. A connection-inactivity value of `0` specifies that connection inactivity is not tracked and the connections, after they are established, are left connected permanently.

Usage

Optional

Default value

If this parameter is not specified, the default value is `0`.

Example

```
connection-inactivity = 0
```

dynamic-groups-enabled

This stanza entry specifies whether dynamic groups are supported.

Syntax

```
dynamic-groups-enabled = {yes|true|no|false}
```

Description

Specification of whether dynamic groups are supported. This key value pair applies to supported LDAP registries. Security Access Manager supports dynamic groups with Security Directory Server regardless of this setting.

Note: This stanza entry can be used only in the `ldap.conf` configuration file.

Options

yes|true
Security Access Manager attempts to resolve dynamic group membership.

no|false
Security Access Manager does not attempt to resolve dynamic group membership. Anything other than `yes|true`, including a blank value, is interpreted as `no|false`.

Usage

Optional

Default value

```
no
```

Example

```
dynamic-groups-enabled = no
```

enabled

This stanza entry specifies whether LDAP is the user registry.

Syntax

```
enabled = {yes|true|no|false}
```

Description

Specification of whether LDAP is being used as the user registry. Only one user registry can be specified at a time.

If enabled, other required stanza entries are an LDAP server host name, and port with which to bind to the server, a bind user DN, and bind user password (obfuscated).

Options

yes|true
: Enables LDAP user registry support.

no|false
: Disables LDAP user registry support and specifies that LDAP is not the user registry that is used. Anything other than yes or true, including a blank value, is interpreted as no or false.

Usage

Conditional. This stanza entry is required when LDAP is the user registry.

Default value

The default value can be different, depending on how the server is configured.

Example

```
enabled = yes
```

host

This stanza entry specifies the host name of the LDAP server.

Syntax

```
host = host_name
```

Description

Host name of the LDAP server. Valid values for *host_name* include any valid Internet Protocol (IP) host name.

The host that is specified by this entry is assumed to be a readwrite type of server with a preference of 5. For a general description of server types and preferences, see "replica" on page 259.

Options

host_name
: The value is taken from the pd.conf file. The pd.conf file is created when the IBM Security Access Manager runtime is configured.

Usage

Required

Default value

There is no default value. The value is taken from the pd.conf file.

Example

```
host = libra
host = libra.dallas.ibm.com
```

ignore-suffix

This stanza entry specifies the LDAP server suffix that is to be ignored when searching for user and group information.

Syntax

```
ignore-suffix = suffix_dn
```

Description

LDAP server suffix that is to be ignored when searching for user and group information. By default, all defined suffixes in the LDAP server are searched when acquiring User and group information.

Note: This stanza entry can be used only in the `ldap.conf` configuration file.

Options

suffix_dn
: Specifies the suffix distinguished name (DN) that you want to be ignored. Repeat this stanza entry for each suffix you want to be ignored. For example, if you specify `ignore-suffix = o=tivoli,c=us`, any user, or group that includes `o=tivoli,c=us` as part of the DN is ignored.

Usage

Optional

Default value

All defined suffixes are searched.

Example

```
ignore-suffix = o=tivoli,c=us
```

max-search-size

This stanza entry specifies the maximum search size, as the number of entries, that can be returned from the LDAP server.

Syntax

```
max-search-size = [0|number_entries]
```

Description

Limit for the maximum search size, specified as the number of entries, that can be returned from the LDAP server. The value for each server can be different, depending on how the server was configured.

Options

0 The number is unlimited. There is no limit to the maximum search size.

number_entries
: The maximum number of entries for search, specified as an integer whole number. This value can be limited by the LDAP server itself.

Usage

Optional

Default value

The default value is server-dependent, but it defaults to 2048 if it is not configured.

Example

```
max-search-size = 2048
```

max-server-connections

This stanza entry specifies the maximum number of connections for the LDAP server.

Syntax

```
max-server-connections = number_connections
```

Description

Specifies the maximum number of connections that are allowed with the LDAP server. The Security Access Manager run time maintains a pool of connections for each LDAP server. From this pool, an available connection is chosen to do requests to the LDAP server. If all connections are busy, a new connection is established with the LDAP server, up to the maximum server connection pool size.

Options

number_connections
The maximum number of connections allowed with the LDAP server. The valid range for this parameter is 2-16. Values greater than 16 are set to 16.

Usage

Optional

Default value

If this parameter is not specified, the default pool size is 16.

Example

```
max-server-connections = 16
```

novell-suffix-search-enabled

This stanza entry specifies whether Security Access Manager searches the entire directory namespace when the Novell eDirectory LDAP server is used as the user registry.

Syntax

```
novell-suffix-search-enabled = {yes|true|no|false}
```

Description

When the Novell eDirectory LDAP server is used as the user registry, Security Access Manager uses this option to determine whether to search the entire directory namespace. The search is for user, group, and policy information which uses a global root search. Otherwise, the search is to automatically determine the set of naming contexts hosted by the LDAP server. The search examines each defined naming context individually for user, group, and policy information.

Options

yes|true
Security Access Manager does naming context (suffix/partition) discovery and searches each naming context for user, group, and policy information. The optional `ignore-suffix` parameters are honored.

no|false
Security Access Manager does a baseless (global root) search of the entire namespace for user, group, and policy information. The optional `ignore-suffix` parameters are ignored.

Usage

Optional; this stanza entry can be used only in the `ldap.conf` configuration file.

Default value

`no`

Example

```
novell-suffix-search-enabled = no
```

port

This stanza entry specifies the non-SSL IP port number that is used for communicating with the LDAP server.

Syntax

```
port = port
```

Description

Non-SSL IP port number that is used for communicating with the LDAP server.

Options

port The value configured for the LDAP server.

Usage

Required

Default value

`389`

Example

```
port = 389
```

replica

This stanza entry specifies the LDAP user registry replicas in the domain.

Syntax

```
replica = ldap-server, port, type, pref
```

Description

Definition of the LDAP user registry replicas in the domain.

Options

ldap-server
The network name of the server.

port
The port number for the LDAP server. A valid port number is any positive number that is allowed by TCP/IP and that is not currently being used by another application.

type
Either `readonly` or `readwrite`.

pref
A number between `1` and `10`, where `10` is the highest preference. The server with the highest preference is chosen for LDAP operations. If multiple servers have the same preference value, then load balancing occurs among the least busy of the servers.

Usage

Optional

Default value

No replicas are specified.

Example

The following example shows one replica that is specified and two replicas that are commented out:

```
replica = freddy,390,readonly,1
#replica = barney,391,readwrite,2
#replica = benny,392,readwrite,3
```

secauthority-suffix

This stanza entry provides a suffix under which the `secAuthorityInfo` object is located. This parameter serves as a starting search location for the `secAuthorityInfo` object when Security Access Manager is started.

Syntax

```
secauthority-suffix = suffix
```

Description

Provides a suffix under which the `secAuthorityInfo` object is located. This parameter serves as a starting search location for the `secAuthorityInfo` object when Security Access Manager is started. If this parameter is set, the specified suffix is searched first to locate the `secAuthorityInfo` object for the domain. If this parameter is not set, or if the `secAuthorityInfo` object is not located within the suffix specified by the parameter, then the entire set of suffixes is searched.

Options

suffix Suffix under which the secAuthorityInfo object is located.

Usage

Optional.

Default value

No suffixes are specified.

Example

```
secauthority-suffix = c=US
```

ssl-port

This stanza entry specifies the SSL IP port that is used to connect to the LDAP server.

Syntax

```
ssl-port = port
```

Description

SSL IP port that is used to connect to the LDAP server.

Options

port Any valid port number. A valid port number is any positive number that is allowed by TCP/IP and that is not currently being used by another application.

Usage

Conditional. This stanza entry is required when Security Access Manager is configured to use SSL or TLS to communicate with the LDAP server (`ssl-enabled = yes`).

Default value

636

Example

```
ssl-port = 636
```

cache-account-policy

The cache-account-policy is a configuration file parameter in the [ldap] stanza for `ldap.conf`. The policy determines whether the LDAP client-side caching is enabled or not.

Syntax

```
cache-account-policy = {yes|true|no|false}
```

Default value

```
yes
```

Option descriptions

yes|true
Enables LDAP client-side caching of expired account or expired password. Anything other than `no|false`, including a blank value, is interpreted as `yes|true`.

no|false
Disables LDAP client-side caching of expired account or password.

Usage Notes

Optional. Enable this configuration file parameter if you want Security Access Manager to cache an expired account or password. Otherwise, set this parameter to no.

Return Values

None.

Example

This example enables the LDAP caching of expired account or password.

```
cache-account-policy = yes
```

user-search-filter

The user-search-filter is a configuration file parameter in the `[ldap]` stanza for `ldap.conf` that specifies the LDAP search filter used by Security Access Manager.

Syntax

```
user-search-filter = <ldap search filter>
```

Default value

For AD LDS and Sun Directory Servers:

```
user-search-filter =
(|(objectclass=ePerson)(objectclass=Person)(objectclass=User))
```

For all other LDAP Directory Servers:

```
user-search-filter = (|(objectclass=ePerson)(objectclass=Person))
```

Option descriptions

Specifies the LDAP search filter used by Security Access Manager to locate users in the LDAP directory server. This filter must be a valid LDAP string search filter as described by the Request for Comments (RFC) 2254 document.

Use the `user-search-filter` option with `user-objectclass` so that the Security Access Manager can locate LDAP users created with the LDAP object classes.

Note: Do not update the unsupported option with the same name under the `[ldap-generic-general]` stanza.

Usage

Optional: Use this configuration file parameter to specify how to locate Security Access Manager users in LDAP.

Example

This example specifies a search for a `User` or `Person` under `objectclass`.

```
[ldap]
user-search-filter = (|(objectclass=User)(objectclass=Person))
```

[manager] stanza

The stanza entries for configuring the master server settings are in the `[manager]` stanza of each of the following configuration files:

- The *[instance-]*`ivacld.conf` configuration file for the authorization server
- The `pd.conf` configuration file when you use the authorization server

management-domain

This stanza entry specifies the name of the management domain.

Syntax

```
management-domain = {default|domain_name}
```

Description

Name of the management domain. This value is created and set by one of the following utilities:

- For the `pd.conf` file, the value is set with the **bassslcfg** utility.
- For other configuration files, the value is set with the **svrsslcfg** utility.

Note: If the value is not specified, then the value of `Default` is used.

Options

Default
Specifies the Management domain. This value is the default value for all servers.

domain_name
Specifies the user-specified domain. Use this value when you configure your own name for the domain.

The *domain_name* value is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set.

Valid characters for domain names for US English are the letters a-Z, the numbers 0-9, a period (.), an underscore (_), a plus sign (+), a hyphen (-), an "at" symbol (@), an ampersand (&), and an asterisk (*). You cannot use a space in the domain name.

Usage

Required

Default value

```
Default
```

Example

```
management-domain = mymgmtdomain
```

master-host

This stanza entry specifies the host name of the Security Access Manager server.

Syntax

master-host = *server_hostname*

Description

Host name of the Security Access Manager server. The following host names are valid:

- `mycomputer.city.company.com`
- `mycomputer`

Options

server_hostname
    Represents the valid name for the host.

Usage

Required

Default value

There is no default value.

Example

```
master-host = ammaster
```

master-port

This stanza entry specifies the TCP port on which the server listens for requests.

Syntax

master-port = *port*

Description

TCP port on which the server listens for requests. This value is created and set by one of the following utilities:

- For the `pd.conf` file, the value is set with the **bassslcfg** utility.
- For all other configuration files, the value is set with the **svrsslcfg** utility.

Options

port Any valid port number. A valid port number is any positive number that is allowed by TCP/IP and that is not currently being used by another application. Use the default port number value, or use a port over 1000 that is currently not being used.

Usage

Required

Default value

The default value is server-dependent.

Example

```
master-port = 7135
```

[meta-info] stanza

The stanza entry for configuring Security Access Manager version information is in the `[meta-info]` stanza of each of the following configuration files:

- The `ivmgrd.conf` configuration file for the policy server
- The *[instance-]*`ivacld.conf` configuration file for the authorization server
- The `pd.conf` configuration file when you use the authorization server

version

This stanza entry specifies the version of Security Access Manager in decimal format.

Syntax

```
version = number
```

Description

Version of Security Access Manager in decimal format.

Note: This value is generated. Do not change it.

[pdconfig] stanza

This stanza defines the configuration key value pairs that are required for the LDAP server. The entries in this stanza are for internal use only. Do not modify the values in this file. To properly configure these entries, use the **`pdconfig`** utility.

LdapSSL

This stanza entry specifies whether to enable SSL communication on the LDAP server.

Syntax

```
LdapSSL = {ssl|nossl}
```

Description

Specification of whether to enable SSL communication on the LDAP server. If the LDAP server is not SSL enabled, any Security Access Manager server that is SSL enabled cannot communicate with the LDAP server.

Note: The entries in this stanza are for internal use only. Do not modify the values in this file. To properly configure these entries, use the **pdconfig** utility.

Options

ssl Enables SSL communication. SSL is automatically configured.

nossl Disables SSL communication. Anything other than `ssl`, including a blank value, is interpreted as `nossl`.

Usage

Optional

Default value

The default value is server-dependent.

Example

```
LdapSSL = nossl
```

LdapSSLKeyFile

This stanza entry specifies the SSL key file name and location.

Syntax

```
LdapSSLKeyFile = ldap-ssl-key-filename
```

Description

SSL key file name and location. Use the SSL key file to handle certificates that are used in LDAP communication. The file extension can be anything, but the extension is usually `.kdb`.

The certificate files in a directory need to be accessible to the server user (or all users). Make sure that the server user (for example, `ivmgr`) or all users have permission to access the `.kdb` file and the folder that contains the `.kdb` file.

Note: The entries in this stanza are for internal use only. Do not modify the values in this file. To properly configure these entries, use the **pdconfig** utility.

Options

ldap-ssl-key-filename
The file name and location that represents an alphanumeric string that is not case-sensitive. String values are expected to be characters that are part of the local code set. The set of valid characters in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). Windows operating

systems path names, however, can have a backward slash (\) or a colon (:). For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

Usage

Conditional. This stanza entry is required when `LdapSSL = ssl`.

Default value

The following table shows the default value by platform.

Table 29. [pdconfig] stanza LdapSSLKeyFile default value by platform

Platform	File name
Linux or UNIX	`/opt/PolicyDirector/keytab/ivmgrd.kdb`
Windows	`c:\program files\tivoli\policy director\keytab\ivmgrd.kdb`

Example

```
LdapSSL = ssl
LdapSSLKeyFile = /opt/PolicyDirector/keytab/ivmgrd.kdb
```

LdapSSLKeyFileDn

This stanza entry specifies the key label of the client certificate within the SSL key file.

Syntax

```
LdapSSLKeyFileDn = keyLabel
```

Description

Key label of the client certificate within the SSL key file. This stanza entry is used when the LDAP server is configured to do client authentication.

Note: The entries in this stanza are for internal use only. Do not modify the values in this file. To properly configure these entries, use the **pdconfig** utility.

Options

keyLabel

Identifies the client certificate that is presented to the LDAP server.

Usage

Conditional. This stanza entry is required when `LdapSSL = ssl`.

Default value

There is no default value.

Example

```
LdapSSL = ssl
LdapSSLKeyFileDn = "PD_LDAP"
```

LdapSSLKeyFilePwd

This stanza entry specifies the password to access the SSL key file.

Syntax

```
LdapSSLKeyFilePwd = ldap-ssl-keyfile-password
```

Description

Password to access the SSL key file.

Note: The entries in this stanza are for internal use only. Do not modify the values in this file. To properly configure these entries, use the **pdconfig** utility.

Options

ldap-ssl-keyfile-password
The password that is associated with the SSL key file. The default SSL key file is key4ssl.

Usage

Conditional. This stanza entry is required when LdapSSL = ssl.

Default value

There is no default value.

Example

```
LdapSSL = ssl
LdapSSLKeyFilePwd = mysslpwd
```

[pdaudit-filter] stanza

The stanza entry for Security Access Manager auditing support is in the [pdaudit-filter] stanza of the pdaudit.conf configuration file.

logcfg

This stanza entry enables logging and auditing for the application.

Syntax

```
logcfg = audit.azn:[log-agent][[param[=value]] ...]
```

Description

Enables logging and auditing for the application. Category, destination, and other parameters are used to capture Security Access Manager auditing and logging events.

Each server provides its own setting for event logging in its corresponding configuration file.

Options

audit.azn:*log-agent*
Category of auditing event. Also specifies that the destination where *log-agent* is one of the following agents:

- `stdout`
- `stderr`
- `file`
- `pipe`
- `remote`

param=value

Allowable parameters. The parameters vary, depending on the category, the destination of events, and the type of auditing you want to do.

See *IBM Security Access Manager for Web: Troubleshooting Guide* for information about the log agents and the configuration parameters.

Usage

Optional

Default value

Remove the number signs (#) at the beginning of the configuration file lines to enable authentication or authorization auditing (or both) for the application.

Example

```
logcfg = audit.azn:file path=audit.log,flush_interval=20,log_id=audit_log
```

[pdrte] stanza

When the policy server is installed, the policy server automatically starts after each system reboot. When the authorization server is installed, the authorization server daemon automatically starts after each system reboot.

The stanza entries for automating server startup when using any of the user registries are in the `[pdrte]` stanza of the following configuration file:

- The `pd.conf` configuration file when you use the authorization server

When you use the Security Access Manager authorization server, you must have the `pd.conf` configuration file.

boot-start-[instance-]ivacld

This stanza entry specifies whether to start the authorization server instance at system startup.

Syntax

```
boot-start-[instance-]ivacld = {yes|no}
```

Description

Specifies whether to start the authorization server instance at system startup. The syntax contains:

- The `boot-start-ivacld` default instance stanza
- Other entries corresponding to configured authorization server instances

When you configure any authorization server with a specified (non-empty string) instance name, the configuration file adds a `boot-start-[`*`instance`*`-]ivacld` entry for that server.

The configuration file always contains an entry for the empty string instance name authorization server, `boot-start-ivacld`.

Options

instance-
: The instance name created during configuration of an authorization server.

yes Start the authorization server at system startup.

no Do not start the authorization server at system startup.

Usage

Conditional. This stanza entry is required for AIX, Linux, and Solaris operating systems only.

Default value

`no`

Example

```
boot-start-ivacld = yes
boot-start-otherinst-ivacld = yes
```

boot-start-ivmgrd

This stanza entry specifies whether to start the policy server at system boot.

Syntax

```
boot-start-ivmgrd = {yes|no}
```

Description

Specification of whether to start the policy server at system boot.

Options

yes Start the policy server at system boot.

no Do not start the policy server at system boot.

Usage

Conditional. This stanza entry is required for AIX, Linux, and Solaris operating systems only.

Default value

`no`

Example

```
boot-start-ivmgrd = yes
```

configured

This stanza entry specifies whether the IBM Security Access Manager runtime package was configured.

Syntax

```
configured = {yes|no}
```

Description

Specification of whether the IBM Security Access Manager runtime package was configured.

Note: This value is generated. Do not change it.

ivacld-instances

This stanza entry specifies a list of configured authorization server instance names.

Syntax

```
ivacld-instances = [ 'instance1' [ 'instance2' [ ...] ] ]
```

Description

A list of configured authorization server instance names.

Options

Each authorization server instance name must be set in single quotation marks and separated by space characters. An empty string instance name is set as the default authorization server instance. This default name must not be present in the list of `ivacld-instances`.

Usage

Conditional. This stanza entry is required for Linux and UNIX operating systems only.

Do not modify this value directly. The authorization server configuration tools maintain this value.

Default value

```
no
```

Example

```
ivacld-instances = 'otherinst' 'otherinst2'
```

tivoli_common_dir

This stanza entry specifies the file name and location for message files and trace log files.

Syntax

tivoli_common_dir = *fully_qualified_path*

Description

File name and location for message files and trace log files. Specifies whether Tivoli Common Directory is used.

Options

fully_qualified_path

Represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of valid characters in a file name can be determined by the file system and by the local code set. For Windows operating systems, file names cannot have a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). Windows operating systems path names, however, can have a backward slash (\) or a colon (:).

For AIX, Linux, and Solaris operating systems, path names and file names are case-sensitive.

Usage

Conditional. This stanza entry is required when you configure the IBM Security Access Manager Runtime for Java environment for Tivoli Common Directory (TCD) logging.

Default value

See *IBM Security Access Manager for Web: Troubleshooting Guide*.

user-reg-host

This stanza entry specifies the user registry host name.

Syntax

```
user-reg-host = hostname
```

Description

User registry host name.

Note: This value is generated during configuration. Do not change it.

user-reg-hostport

This stanza entry specifies the non-SSL IP port number that is used for communicating with the user registry server.

Syntax

```
user-reg-hostport = port
```

Description

Non-SSL IP port number that is used for communicating with the user registry server.

Note: This value is generated during configuration. Do not change it.

user-reg-server

This stanza entry specifies the user registry server name.

Syntax

```
user-reg-server = server_name
```

Description

User registry server name.

Note: This value is generated during configuration. Do not change it.

user-reg-type

This stanza entry specifies the user registry type.

Syntax

```
user-reg-type = {ldap|active_directory}
```

Description

User registry type.

Note: This value is generated during configuration. Do not change it.

[ssl] stanza

The `[ssl]` stanza in the configuration file defines the Secure Sockets Layer (SSL) configuration settings for the Security Access Manager servers.

The stanza entries for configuring Security Access Manager SSL settings are in the `[ssl]` stanza of each of the following configuration files:

- The `ivmgrd.conf` configuration file for the policy server
- The *`[instance-]`*`ivacld.conf` configuration file for the authorization server
- The `pd.conf` configuration file when you use the authorization server
- Your configuration file of the resource manager for configured SSL entries

  The `aznAPI.conf` configuration file is provided with Security Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins must provide the standard functions. Before you implement service plug-ins, read and thoroughly understand the concepts in the *Authorization C API Developer Reference*.

ssl-authn-type

This stanza entry specifies the type of authentication.

Syntax

```
ssl-authn-type = certificate
```

Description

Type of authentication.

Note: This value is created and its value set during configuration for the authentication server and the policy proxy server. This stanza entry is not used for the policy server.

Do not edit this stanza entry.

Default value

```
certificate
```

ssl-auto-refresh

This stanza entry specifies whether automatic refresh of the SSL certificate and the key database file password occur.

Syntax

```
ssl-auto-refresh = {yes|no}
```

Description

Specification of whether automatic refresh of the SSL certificate and the key database file password occur.

This value is created and its value set with one of the following utilities:

- For the `ivmgrd.conf` configuration file, it is set with the **`mgrsslcfg`** utility.
- For the `pd.conf` configuration file, it is set with the **`bassslcfg`** utility.
- For all other configuration files, it is set with the **`svrsslcfg`** utility.

Note: This value is set with a configuration utility. Do not edit this stanza entry.

Options

`yes` Enables automatic certificate and password refresh.

`no` Disables automatic certificate and password refresh.

ssl-cert-life

This stanza entry specifies the number of days in the lifetime of a certificate.

Syntax

ssl-cert-life = *number_days*

Description

Value for number of days in the lifetime of a certificate. Any issued or renewed certificates must use this value.

For the `ivmgrd.conf` configuration file, set this value with the **`mgrsslcfg`** utility to a value 1 - 7299. (The default is `1460`, or four years.) The name and path are fixed for this configuration file. Use this utility to modify this value after initial configuration.

To increase or decrease the value, change the value and restart the policy server. The new value is in effect only for certificates that are issued or that are renewed from that point on. If both the certificate and the password to the key database file that contains the certificate expire, the password must be refreshed first.

Note:

1. Only the policy server uses this value.
2. The password value is set with the **`mgrsslcfg`** utility.
3. Do not edit this stanza entry.

disallow-trailing-spaced-usernames

The disallow-trailing-spaced-usernames is a configuration file parameter in the [ssl] stanza for pd.conf that determines whether trailing spaces in usernames are accepted.

Syntax

```
disallow-trailing-spaced-usernames = {yes|no}
```

Default value

no

Option descriptions

yes Indicates that trailing spaces in usernames are not accepted.

no Indicates that trailing spaces in usernames are accepted.

Usage notes

Enable this configuration file parameter if trailing spaces in usernames are not accepted. This configuration file parameter is optional.

Example

This example indicates that trailing spaces in usernames are not accepted.

```
disallow-trailing-spaced-usernames = yes
```

ssl-compliance

Syntax

```
ssl-compliance = { none | fips | sp800-131-transition | sp800-131-strict
                 | suite-b-128 | suite-b-192 }
```

Description

Determines which compliance mode is enabled.

Options

none Indicates that no special compliance modes are applied to the TLS communication protocol. This setting is equivalent to `[ssl] ssl-enable-fips = no`, which is a deprecated option.

fips Enables FIPS 140-2 compliance. This setting is equivalent to `[ssl] ssl-enable-fips = yes`, which is a deprecated option.

sp800-131-transition
Enables NIST SP 800-131a support at the transition level. The transition level has fewer restrictions than the strict level.

sp800-131-strict
Enables NIST SP 800-131a support at the strict level. This enforcement is required by some federal agencies and enterprises that work with the federal government starting in 2014.

suite-b-128
Enables NSA Suite B at the 128-bit support level.

suite-b-192
Enables NSA Suite B at the 192-bit support level.

Usage

Required.

This setting is used for secure communication between Security Access Manager processes, secure communication from Security Access Manager to the LDAP registry servers, and secure communication from Security Access Manager to syslog servers.

When a Security Access Manager Java component is running in WebSphere Application Server, then WebSphere Application Server must be running with the same compliance standard as Security Access Manager. For details on configuring WebSphere Application Server for various compliance modes, see http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.was_v8/was/8.0.0.3/Security/WASV8003_SecurityCryptoSignatureAlgorithm/player.html.

To configure Security Access Manager with a specific compliance, set the `ssl-compliance` value in `pd.conf` immediately before you configure the Security Access Manager policy server. The `ssl-compliance` option takes precedence over the deprecated `ssl-enable-fips` option if both are present.

Default value

none

Example

```
ssl-compliance = suite-b-128
```

ssl-enable-fips (deprecated)

This stanza entry determines whether Federal Information Process Standards (FIPS) mode is enabled. This entry is deprecated and replaced by the `ssl-compliance` entry.

Syntax

```
ssl-enable-fips = {yes|no}
```

Description

Determines whether Federal Information Process Standards (FIPS) mode is enabled. If enable, set to `yes`, Transport Layer Security (TLS) version 1 (TLSv1) is the secure communication protocol used. If not enabled, set to `no`, SSL version 3 (SSLv3) is the secure communication protocol used.

Options

yes Specifies that TLSv1 is the secure communication protocol.

no Specifies that SSLv3 is the secure communication protocol.

Usage

Required

Default value

There is no default value. This value is set by the configuration utility that is associated with each server.

Example

```
ssl-enable-fips = no
```

ssl-enhanced-security

Syntax

```
ssl-enhanced-security = {yes|true|no|false}
```

Description

Controls the enhanced security added to the proprietary Inter-Process Communication (IPC) in Access Manager. It affects the following functions:

- All pdadmin server task commands that are sent from the Policy Server to Tivoli® Access Manager applications, such as WebSEAL (webseald), at the Tivoli Access Manager application side of the communication.
- Remote logging at the Authorization Server (pdacld) side of the communication.
- The policy database change notification sent to Tivoli Access Manager local mode applications at the application side.

Options

yes | true
Enables enhanced security.

no | false
Disables enhanced security. Anything other than `yes` or `true`, including a blank value, is interpreted as `no` or `false`.

Default value

This option is enabled by default.

Example

```
ssl-enhanced-security = yes
```

ssl-io-inactivity-timeout

This stanza entry specifies the time in seconds that an SSL connection waits for a response before timing out.

Syntax

```
ssl-io-inactivity-timeout = {0|number_seconds}
```

Description

Duration in seconds that an SSL connection waits for a response before timing out. For certain administration requests, you might receive the HPDBA0219E error message when the timeout value is too small. For example, a request might be to look up members in a large user registry group over an SSL connection. To resolve this problem, increase this timeout value in the `pd.conf` configuration file.

This timeout value is created, and the value is set with one of the following utilities:

- For the `ivmgrd.conf` configuration file, it is set with the **mgrsslcfg** utility.
- For the `pd.conf` configuration file, it is set with the **bassslcfg** utility.
- For all other configuration files, it is set with the **svrsslcfg**.

Note: The timeout value is set with the configuration utility. Do not edit this stanza entry.

Options

0 No timeout is allowed.

number_seconds
Specifies the number of seconds that an SSL connection waits for a response before timing out. There is no range limitation for this timeout value.

Usage

Required

Default value

There is no default value. This value is set by the configuration utility that is associated with each server.

Example

```
ssl-io-inactivity-timeout = 300
```

ssl-keyfile

This stanza entry specifies the file name and location on the local system of the SSL key file.

Syntax

```
ssl-keyfile = key-path
```

Description

File name and location on the local system of the SSL key file. If the key-value pair does not exist in the configuration file, the application fails. The file extension can be anything, but it is usually `.kdb`. By default, the key file is in one of the following operating system-specific directories:

AIX, Linux, and Solaris operating systems
`/var/PolicyDirector/keytab`

Windows operating systems
`c:\program files\tivoli\policy director\keytab`

The certificate files in a directory need to be accessible to the server user (or all users). Make sure that server user (for example, `ivmgr`) or all users have permission to access the `.kdb` file and the folder that contains the `.kdb` file.

This file is created, and the value is set with one of the following utilities:

- For the `ivmgrd.conf` configuration file, it is set with the **mgrsslcfg** utility.

- For the `pd.conf` configuration file, it is set with the **`bassslcfg`** utility.
- For all other configuration files, it is set with the **`svrsslcfg`** utility.

Note: The file name, including extension, is generated and set by the configuration utility. Do not edit this stanza entry.

ssl-keyfile-label

This stanza entry specifies the label of the key, other than the default key that you can use.

Syntax

```
ssl-keyfile-label = label
```

Description

Label of the key to use other than the default. Quotation marks that surround the *label* value are not permitted.

This label is created, and the value is set with one of the following utilities:

- For the `ivmgrd.conf` configuration file, it is set with **`mgrsslcfg`** utility
- For the `pd.conf` configuration file, this entry does not apply.
- For all other configuration files, it is set with the **`svrsslcfg`** utility.

Note: The label is set by the configuration utility. Do not edit this stanza entry.

ssl-keyfile-stash

This stanza entry specifies the file name and location of the SSL password stash file that protects private keys in the key file.

Syntax

```
ssl-keyfile-stash = stash-path
```

Description

File name and location of the SSL password stash file that is used to protect private keys in the key file. The password might be stored encrypted in the stash file.

The file extension can be anything, but it is usually `.sth`. By default, the key file is in one of the following operating system-specific directories:

AIX, Linux, and Solaris operating systems

```
/var/PolicyDirector/keytab
```

Windows operating systems

```
c:\program files\tivoli\policy director\keytab
```

This file is created, and the value is set with one of the following utilities:

- For the `ivmgrd.conf` configuration file, it is set with the **`mgrsslcfg`** utility.
- For the `pd.conf` configuration file, it is set with the **`bassslcfg`** utility.
- For all other configuration files, it is set with the **`svrsslcfg`** utility. The path is defined by the **–d** option, and the name is defined by the **–n** option.

Note: The file name, including extension, is generated and set by the configuration utility. Do not edit this stanza entry.

ssl-listening-port

This stanza entry specifies the TCP port to listen on for incoming requests.

Syntax

```
ssl-listening-port = {0|port}
```

Description

TCP port to listen on for incoming requests.

Note: The policy server does not use this stanza entry.

Options

0
Disables listening. The value is specified during configuration by with the **svrsslcfg** utility.

Note: Do not change this parameter directly; modify the parameter only by issuing the **scrsslcfg -chgport** command so that the policy server knows that the listening port is changed. Otherwise, the resource manager cannot receive policy update notifications or pdadmin server task commands.

port
Enables listening at the specified port number. The valid range for *port* is any positive number that is allowed by TCP/IP and is not currently being used by another application.

There is no one default value, because the configuration programs for each daemon specify its own default value.

Usage

Required, except for the policy server.

Default value

If not specified during configuration, the default value is `0`. Otherwise, the value is server-dependent.

Example

```
ssl-listening-port = 8139
```

ssl-local-domain

This stanza entry specifies the name of the local domain.

Syntax

```
ssl-local-domain = {Default|domain_name}
```

Description

The name of the local domain. The server runs on this domain. If this value is not in the configuration file, then operations that rely on its presence fail.

The domain name value is created during configuration, but you can change it using one of the following utilities:

- For the `ivmgrd.conf` configuration file, change it with the **mgrsslcfg** utility.

- For the `pd.conf` configuration file, change it with the **bassslcfg** utility.
- For all other configuration files, change it with the **svrsslcfg** utility.

Note: This value is set during configuration or set with the configuration utility. Do not edit this stanza entry.

ssl-maximum-worker-threads

This stanza entry specifies the number of threads that the server can create to handle incoming requests.

Syntax

```
ssl-maximum-worker-threads = number_threads
```

Description

Number of threads that can be created by the server to handle incoming requests.

Options

number_threads
Number of threads that can be specified. The valid range must be equal to or greater than 1. The maximum number varies, because it is dependent on available system resources.

Usage

Required

Default value

The default value is server-dependent.

Example

```
ssl-maximum-worker-threads = 50
```

ssl-pwd-life

This stanza entry specifies the password lifetime in days for the key database file.

Syntax

```
ssl-pwd-life = number_days
```

Description

Password lifetime for the key database file, specified in the number of days. For automatic password renewal, the value for the lifetime of a password is controlled by the *number_days* value when the server is started.

The number of days 1 - 7299 is created, and the value is set with one of the following utilities:

- For the `ivmgrd.conf` configuration file, it is set with the **mgrsslcfg** utility.
- For the `pd.conf` configuration file, it is set with the **bassslcfg** utility.
- For all other configuration files, it is set with the **svrsslcfg** utility.

For manual password renewal, the value is dictated by the value supplied to the **svrsslcfg –chgpwd** utility. This value is also written into the appropriate configuration file.

Note:

1. If a certificate and the password to the key database file that contains that certificate are both expired, the password must be refreshed first.
2. The password value is set with the configuration utility. Do not edit this stanza entry.

ssl-v3-timeout

This stanza entry specifies the session timeout in seconds for SSL Version 3 connections between clients and servers.

Syntax

```
ssl-v3-timeout = number_seconds
```

Description

Session timeout in seconds for SSL Version 3 connections between clients and servers. This timeout value controls how often a full SSL handshake is completed between Security Access Manager clients and servers.

This timeout value is created, and the value is set with one of the following utilities:

- For the `ivmgrd.conf` configuration file, it is set with the **mgrsslcfg** utility.
- For the `pd.conf` configuration file, it is set with the **bassslcfg** utility.
- For all other configuration files, it is set with the **svrsslcfg**. The path is defined by the **–d** option, and the name is defined by the **–n** option.

Note:

1. Security Access Manager components might not function with small timeout values in some network environments.
2. The timeout value is set with the configuration utility. Do not edit this stanza entry.

ssl-v2-enable

This stanza entry specifies the Secure Sockets Layer (SSL), Version 2, settings for Security Access Manager server communication.

Syntax

```
ssl-v2-enable = {yes|true|no|false}
```

Description

The `ssl-v2-enable` option has an effect only when `ssl-compliance = none`. The setting is for compatibility with version 6.1.1 of Security Access Manager.

Enabling SSL Version 2 is not required. By default, the SSL Version 2 protocol is disabled, which increases communication security.

Options

yes|true
: Enables the SSL v2 protocol. The protocols SSL v3 and TLS are also choices for Security Access Manager communication and are chosen by Security Access Manager in preference to SSL Version 2, even if this setting is enabled.

no|false
: Disables the SSL Version 2 protocol. Anything other than `yes` or `true`, including a blank value, is interpreted as `no` or `false`.

Usage

Optional. If used, must be set to the same value in all Security Access Manager server and application configuration files.

Default value

The default value is no.

Example

The following example disabled the SSL Version 2 protocol.

```
ssl-v2-enable = no
```

ssl-session-cache-size

This stanza entry specifies the number of concurrent sessions supported by a Security Access Manager server or service.

Syntax

```
ssl-session-cache-size = numeric value
```

Description

A larger cache size can improve the Secure Sockets Layer (SSL) security performance of the server if you add additional memory.

The cache size can be 1-524,288 number of concurrent sessions.

Note: You might need to double the SSL session cache size for a target maximum cache size. For example: if you want a cache size of 8,192, set the `ssl-session-cache-size` parameter value to 16,384.

Options

numeric value
: The cache size can be 1-524,288 number of concurrent sessions.

Usage

Optional.

Default value

The default value is 1024.

Example

The following example sets the cache size at 1024.

```
ssl-session-cache-size = 1024
```

ssl-v3-cipher-specs

This stanza entry defines the Secure Sockets Layer (SSL), Version 3, ciphers.

Syntax

ssl-v3-cipher-specs = *configuration string*

Description

To specify the ciphers for SSL Version 3, modify the `ssl-v3-cipher-specs` parameter value in the appropriate configuration file. You can specify one or more ciphers. Enter multiple ciphers in a comma-separated list.

Any SSL Version 3 connection to a Security Access Manager server or service is limited to the set of ciphers defined in the `ssl-v3-cipher-specs` parameter.

Options

configuration string
Allowed SSLV30 CipherSpecs:

```
TLS_RSA_WITH_NULL_NULL, TLS_RSA_WITH_NULL_MD5,
TLS_RSA_WITH_NULL_SHA, TLS_RSA_EXPORT_WITH_RC4_40_MD5,
TLS_RSA_WITH_RC4_128_MD5, TLS_RSA_WITH_RC4_128_SHA,
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5, TLS_RSA_WITH_DES_CBC_SHA,
TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
(Deprecated), TLS_RSA_EXPORT1024_WITH_RC4_56_SHA (Deprecated),
SSL_RSA_FIPS_WITH_DES_CBC_SHA (Deprecated),
SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (Deprecated),
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA
```

FIPS and NIST SP800-131a allowed SSLV30 CipherSpecs:

```
None
```

Usage

Optional.

Default value

Default SSLV30 CipherSpecs:

```
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_RC4_128_MD5, TLS_RSA_WITH_DES_CBC_SHA,
TLS_RSA_EXPORT_WITH_RC4_40_MD5, TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5,
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA,
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA, TLS_RSA_WITH_NULL_SHA,
TLS_RSA_WITH_NULL_MD5
```

Example

```
ssl-v3-cipher-specs = TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA
```

tls-v10-cipher-specs

This stanza entry specifies the Transport Layer Security (TLS), Version 1.0, ciphers to use.

Syntax

```
tls-v10-cipher-specs = configuration string
```

Description

To specify the ciphers for TLS Version 1.0, modify the `tls-v10-cipher-specs` parameter value in the appropriate configuration file. Enter multiple ciphers in a comma-separated list.

Any TLS Version 1.0 connection to a Security Access Manager server or service is limited to the set of ciphers defined in the `tls-v10-cipher-specs` parameter.

Options

configuration string

Allowed TLSV10 CipherSpecs:

```
TLS_RSA_WITH_NULL_NULL, TLS_RSA_WITH_NULL_MD5,
TLS_RSA_WITH_NULL_SHA, TLS_RSA_EXPORT_WITH_RC4_40_MD5,
TLS_RSA_WITH_RC4_128_MD5, TLS_RSA_WITH_RC4_128_SHA,
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5, TLS_RSA_WITH_DES_CBC_SHA,
TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
(Deprecated), TLS_RSA_EXPORT1024_WITH_RC4_56_SHA (Deprecated),
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA
```

FIPS and NIST SP800-131a allowed TLSV10 CipherSpecs:

```
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_3DES_EDE_CBC_SHA
```

Usage

Optional.

Default value

Default TLSV10 CipherSpecs:

```
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_RC4_128_MD5 TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5 TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5,
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA,
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA, TLS_RSA_WITH_NULL_SHA,
TLS_RSA_WITH_NULL_MD5
```

Example

```
tls-v10-cipher-specs = TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA
```

tls-v11-cipher-specs

Defines the Transport Layer Security (TLS), Version 1.1, ciphers.

Syntax

```
tls-v11-cipher-specs = configuration string
```

Description

To specify the ciphers for TLS Version 1.1, modify the `tls-v11-cipher-specs` parameter value in the appropriate configuration file. You can specify one or more ciphers. Enter multiple ciphers in a comma-separated list.

Any TLS Version 1.1 connection to a Security Access Manager server or service is limited to the set of ciphers defined in the `tls-v11-cipher-specs` parameter.

Options

configuration string

Allowed TLSV11 CipherSpecs:

```
TLS_RSA_WITH_NULL_NULL, TLS_RSA_WITH_NULL_MD5,
TLS_RSA_WITH_NULL_SHA, TLS_RSA_WITH_RC4_128_MD5,
TLS_RSA_WITH_RC4_128_SHA, TLS_RSA_WITH_DES_CBC_SHA,
TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA,
TLS_RSA_WITH_AES_256_CBC_SHA
```

FIPS and NIST SP800-131a allowed TLSV11 CipherSpecs:

```
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_3DES_EDE_CBC_SHA
```

Usage

Optional.

Default value

Default TLSV11 CipherSpecs:

```
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_RC4_128_MD5, TLS_RSA_WITH_DES_CBC_SHA,
TLS_RSA_WITH_NULL_SHA, TLS_RSA_WITH_NULL_MD5
```

Example

```
tls-v11-cipher-specs = TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA
```

tls-v12-cipher-specs

This stanza entry specifies the Transport Layer Security (TLS), Version 1.2, ciphers.

Syntax

```
tls-v12-cipher-specs = configuration string
```

Description

To specify the ciphers for TLS Version 1.2, modify the `tls-v12-cipher-specs` parameter value in the appropriate configuration file. You can specify one or more ciphers. Enter multiple ciphers in a comma-separated list.

Any TLS Version 1.2 connection to a Security Access Manager server or service is limited to the set of ciphers defined in the `tls-v12-cipher-specs` parameter.

Options

configuration string

Allowed TLSV12 CipherSpecs: The allowed CipherSpecs are the same as the Default TLSV12 CipherSpecs with the addition of:

```
TLS_RSA_WITH_NULL_NULL TLS_RSA_WITH_RC4_128_SHA
TLS_ECDHE_RSA_WITH_RC4_128_SHA TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
```

FIPS and NIST SP800-131a allowed TLSV12 CipherSpecs:

```
TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
```

Suite B Allowed TLSV12 CipherSpecs:

```
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
```

Usage

Optional.

Default value

Default TLSV12 CipherSpecs:

```
TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
```

```
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_NULL_SHA256
TLS_RSA_WITH_NULL_SHA TLS_ECDHE_RSA_WITH_NULL_SHA
TLS_ECDHE_ECDSA_WITH_NULL_SHA
```

Note: TLS Version 1.2 CipherSpecs that do not explicitly indicate a SHA256 or SHA384 hash implicitly use a SHA256 or SHA384 hash. However, the use of CipherSpecs that do not explicitly indicate a SHA256 or SHA384 hash with TLS Version 1.2 might result in interoperability problems with SSL and TLS stacks. CipherSpecs with explicit SHA256 or SHA384 hashes must be used.

Example

```
tls-v12-cipher-specs = TLS_RSA_WITH_AES_256_GCM_SHA384,TLS_RSA_WITH_AES_128_CBC_SHA256
```

[xmladi-attribute-definitions] stanza

The stanza entries for configuring the Access Decision Information Extensible Markup Language (ADI XML) document attribute definitions are in the [xmladi-attribute-definitions] stanza.

This stanza can be found or placed into any of the Security Access Manager configuration files, except for the pd.conf configuration file.

The aznAPI.conf configuration file is provided with Security Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins typically provide the standard functions. Before you implement service plug-ins, read and thoroughly understand the concepts in the *Authorization C API Developer Reference*.

AttributeName

This stanza entry defines ADI XML document attributes that are inserted into the XML ADI element start tag.

Syntax

```
AttributeName = "AttributeValue"
```

Description

Definition of ADI XML document attributes that are inserted into the XML ADI element start tag. The entry enables attributes to be defined for the entire XML ADI document and for all ADI defined in the XML ADI document.

The ADI XML model requires that the XML document contains the following top-level XML element into which all target ADI for a particular rule evaluation is inserted. The XMLADI element is created automatically as part of the rule evaluation process:

```
<XMLADI>
<!--  XML formatted ADI are inserted here.  -->
</XMLADI>
```

Usage

Required

Example

```
xmlns:myNS = "http://myURI.mycompany.com"
appID = '"Jupiter" - Account Management Web Portal Server #1.'
```

The attribute value must be enclosed in either double or single quotation marks.

The following XMLADI element start tag defines these attributes:

```
<XMLADI xmlns:myNS="http://myURI.mycompany.com"
appID='"Jupiter" - Account Management Web Portal Server #1.'>
```

See Chapter 10, "Authorization rules management," on page 117.

Appendix D. User registry differences

Each user registry presents unique concerns when integrated with Security Access Manager. This release of Security Access Manager supports the specified LDAP user registries.

Security Access Manager supports the following LDAP user registries:

- Active Directory Lightweight Directory Service (AD LDS)
- IBM z/OS® Security Server LDAP Server
- Novell eDirectory Server
- Sun Java System Directory Server
- Security Directory Server

General concerns

There are several concerns specific to all supported user registries.

- Do not use the forward slash (/) character when defining the names for users and groups when distinguished names strings are used to define the name. Each user registry treats this character differently.
- Do not use leading and trailing blanks in user and group names. Each user registry treats blanks differently.

LDAP concerns

There are several concerns specific to all supported LDAP user registries.

- There are no configuration steps needed to make Security Access Manager support the password policy of the LDAP. Security Access Manager does not assume that the LDAP has its own password policy. Security Access Manager first enforces its own Password Policy. Security Access Manager attempts to update the password in LDAP only if the provided password meets the requirements of the Security Access Manager password policy.
- Next, Security Access Manager implements the password policy of the LDAP by using the return code that it gets from LDAP during a password-related update.
- If Security Access Manager can map the return code without ambiguity with the corresponding error code, then it maps the code and returns an error message.
- To take advantage of the multi-domain support in Security Access Manager, you must use an LDAP user registry.
- 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.
- 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.

Modifying Sun Java System Directory Server look-through limit

When the directory server is installed, the default value is 5000. You can modify this value.

About this task

If the user registry contains more entries than the defined look-through limit, then the directory server returns the following status:

```
LDAP_ADMINLIMIT_EXCEEDED
```

Security Access Manager treats this status as an error.

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.

   Note: 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 Service (AD LDS) concerns

The following concerns are specific to Microsoft Active Directory Lightweight Directory Service (AD LDS).

- In the Policy Server configuration, you can select either a standard data model or a minimal data model for the user registry. If you use AD LDS, you must select the minimal data model, because AD LDS allows only a single naming attribute when creating LDAP objects. When AD LDS is selected as the user registry, Security Access Manager always uses the minimal data model even if the standard data model is selected during the Policy Server configuration.
- The common name (`cn`) attribute is a single-value attribute and can store only one value. The AD LDS registry requires the value of `cn` to be the same as the `cn` naming attribute in the distinguished name (`dn`) attribute. When creating a user or group in Security Access Manager, specify the same value for `cn` as the `cn` naming attribute in the `dn`. Security Access Manager ignores the value of the `cn` attribute if it is different from the value of the `cn` naming attribute in the `dn`. For example, you cannot use the following command to create a user because the value of the `cn` attribute, *`fred`*, is different from the `cn` naming attribute in the `dn`, *`user1`*:

  ```
  pdadmin user create user1 cn=user1,o=ibm,c=us fred smith password1
  ```

Federated registry support

Consider the following points when you configure federated registry support.

The federated registry support feature provides the following benefits:

- It provides the ability to use Security Access Manager with a user registry without requiring the addition of Security Access Manager schema changes, accounts, and access controls in the user registry. These are instead stored in an separate registry leaving the user registry untouched.
- Multiple user registries can be federated into one common Security Access Manager registry.
- It introduces a method to support Active Directory for policy servers running on the appliance.

The following limitations apply to supported federated registries:

- IBM Security Directory Server nested groups are not supported for foreign user DN members.
- Dynamic groups are only supported for user DNs in the same registry in which the group exists.

  Note: Each domain in Active Directory Forest is considered a different registry.
- IBM Security Directory Server supports foreign member DNs. This means that Active Directory native users could be added to Security Access Manager registry groups such as **su-admins**. However the nested group limitation applies.
- Active Directory:
  - Active Directory groups do not support adding foreign user DN members. For example, users in the Security Access Manager registry such as **sec_master** can not be made members of Active Directory groups.
  - When a native Active Directory user is removed by a process outside of Security Access Manager, and that user was also a Security Access Manager enabled user, Security Access Manager does not automatically detect and remove the Security Access Manager component of this user.
  - Active Directory Forest: The Global Catalog is not an option for locating user credentials, only the Security Access Manager `principalName` is used.
  - The connection to the registry must be over SSL/TLS for password operations to be permitted by Active Directory.
- Security Access Manager must be provided with an account (**bind-dn**) with sufficient privileges for the operations required on the suffixes of registry being federated to it. In some cases the actual users being federated need certain privileges.
- The **bind-dn** account is shared by all Security Access Manager processes on the same appliance. Thus it requires the permissions that will satisfy all these processes. For example, if both WebSEAL and the policy server are on the same appliance, then the **bind-dn** account needs all the permissions required by both.
- For all registry types and use cases: The **bind-dn** account needs permission to read attributes from the root DSE entry ("").
- Password Change:
  - If options **bind-auth-and-pwdchg = yes** or **enhanced-pwd-policy = yes** are set, then Security Access Manager managed federated users need permission to modify their own password. For example, the IBM Security Directory Server LDAP server requires an ACL to be attached, such as:

    ```
    aclEntry: access-id:cn=this:at.userPassword:grant:w
    ```

  - If **bind-auth-and-pwdchg = no**, **enhanced-pwd-policy = no**, **auth-using-compare = yes**, and **auth-using-compare** is supported for the LDAP server, then the **bind-dn** account needs permission to be able to do an LDAP compare operation on the user's **userPassword** attribute.
  - If neither of the previous two configurations are in force, then the **bind-dn** account needs **password reset** permissions discussed as follows.
    - The configuration property **bind-auth-and-pwdchg** is configurable per federated registry stanza and for the Security Access Manager registry itself. It defaults to `FALSE` for all federated registries except Active Directory.
    - The configuration property **enhanced-pwd-policy** is a single global setting common to all federated registries and the Security Access Manager registry itself. The Active Directory federated registry implementation does not provide all the features of **enhanced-pwd-policy**.
- Password Reset:

  If you reset the password with the `pdadmin> user modify <user> password <password>` command, then the **bind-dn** account needs permission to be able to set or reset a user's **userPassword** attribute (or **unicodePwd** for Active Directory).
- Credential Construction:
  - Active Directory:
    - The **bind-dn** account needs permission to be able to read the **memberOf** attribute of a user.
    - If **dynamic-groups-enabled = yes**, then **bind-dn** account needs permission to read the authorization store containing the groups with attribute **groupType=34** and read the group attribute **msDS-AzLDAPQuery**, which contains a search filter. The **bind-dn** account then needs permission to search for user entries under the specified suffixes using this search filter.
  - IBM Security Directory Server: The **bind-dn** account needs permission to be able to read the **ibm-allGroups** attribute of a user.
  - Oracle Sun Directory Server: If **dynamic-groups-enabled = yes**, then **bind-dn** account needs permission to search under the specified suffix for group entries with an **objectClass=groupOfURLs**. It also needs permission to read the group attribute **memberURL**, which contains a search suffix, scope, and filter. The **bind-dn** account then needs permission to search for user entries under the specified suffix using this search filter and scope.
  - For other supported registries, the **bind-dn** account needs permission to search for group entries under the specified suffixes and read their member entries.

    Notes:
    - The configuration property **dynamic-groups-enabled** is configurable per federated registry stanza and for the Security Access Manager registry itself. It defaults to `FALSE` except for IBM Security Directory Server, which automatically provides dynamic group memberships using the **ibm-allGroups** attribute.
    - Read access to additional user entry attributes might be required if Security Access Manager is configured to fetch and add their values to the credential.
- Import User or Group: The **bind-dn** account needs permission to read the user or group **objectclass** attribute.
- If you do not plan to view, add, remove, or modify the federated registry native user and group accounts, then no additional permissions are required.

- Federated registry native user and group administration via **pdadmin**: The **bind-dn** account must have permission to view, add, remove, and modify users and groups and modify group memberships under the specified federated registry suffix.

AD registry support configuration

Follow these steps to set up the Security Access Manager to support federated registries.

1. Set up a Security Access Manager policy server using either the appliance internal LDAP server or an external TDS LDAP server.

   Note: You cannot set up the policy server directly in AD.
2. Federate the AD registry suffixes into the Security Access Manager registry. The AD registry must be accessed using its LDAP SSL interface. You can use either of the following methods:
   - Using the Federated Directories management page in the appliance LMI
     a. From the top menu, select **Secure Web Settings** > **Manage** > **Runtime Component**.
     b. Select **Manage** > **Federated Directories**.
     c. Click **New**.
     d. Enter the information for the AD registry suffix to add.
     e. Click **Save**.
   - Modifying the ldap.conf configuration file manually. For example:

     ```
     [server:myadserver]
     bind-dn = CN=Administrator,CN=Users,DC=home,DC=com
     bind-pwd = P@ssw0rd
     host = myad.ibm.com
     port = 636
     ssl-enabled = yes
     suffix = DC=home,DC=com
     #dynamic-groups-enabled = no
     bind-auth-and-pwdchg = yes
     ```

     Note: You can federate multiple servers. So for an AD forest, you would need multiple **[server:<inst>]** stanzas for each AD server that contains users or groups you want to access. Some of the options such as **dynamic-groups-enabled** and **bind-auth-and-pwdchg** can only be set by manually modifying the ldap.conf configuration file. You cannot use the LMI method to set these options.
3. If you are not using SSL to your Security Access Manager registry (internal or TDS), then you also need to set up a keyfile for the SSL access to AD. You can use either of the following methods:
   - Using the Federated Directories management page in the appliance LMI
     a. From the top menu, select **Secure Web Settings** > **Manage** > **Runtime Component**.
     b. Select **Manage** > **Federated Directories**.
     c. Click **SSL Settings**.
     d. Provide values for the displayed fields.
     e. Click **Save**.
   - Modifying the ldap.conf configuration file manually. For example:

     ```
     [ldap]
     ssl-keyfile = /opt/PolicyDirector/etc/ldapssl.kdb
     ssl-keyfile-pwd = passw0rd
     ```

Note: The **bind-pwd** and **ssl-keyfile-pwd** should be placed in the ldap.conf obfuscation file. The **ssl-keyfile** settings in ivmgrd.conf, ivacld*.conf or webseald*.conf take precedence for their respective processes over the values in ldap.conf.

4. After you have federated AD suffixes into your Security Access Manager registry, you can manipulate the AD users and groups in those suffixes as normal Security Access Manager users. For example, you can import an AD user into Security Access Manager:

   ```
   pdadmin sec_master> user import testuser CN=testuser,CN=Users,DC=home,DC=com
   ```

   There are some restrictions as to foreign group memberships (cross-federated registry). For example, AD groups do not allow adding Security Access Manager registry (non-AD) user members, but appliance internal or TDS registry groups do allow AD user members. You can add the AD user Administrator to su-admins (cn=su-admins,cn=SecurityGroups,secAuthority=Default), but you cannot add the Security Access Manager user sec_master to the AD group Administrators (CN=Administrators,CN=Builtin,DC=home,DC=com).

Length of names

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 30 for a comparison of the maximum lengths that are allowed and the recommended maximum length to use to ensure compatibility with all the user registries that are supported by Security Access Manager.

Table 30. Maximum lengths for names by user registry and the optimum length across user registries

Name	IBM Security Directory Server	IBM z/OS Security Server	Novell eDirectory Server	Sun Java System Directory Server	Microsoft Active Directory Server	Active Directory Lightweight Directory Service (AD LDS)	Optimum length
Given name (LDAP CN)	256	256	64	256	64	64	64
Middle name	128	128	128	128	64	64	64
Family name	128	128	128	128	64	64	64
Registry UID (LDAP DN)	1024	1024	1024	1024	2048	1024	255
Security Access Manager user identity	256	256	256	256	64	64	64
User password	unlimited	unlimited	unlimited	unlimited	256	128	256
User description	1024	1024	1024	1024	1024	1024	1024
Group name	256	256	256	256	64	64	64
Group description	1024	1024	1024	1024	1024	1024	1024

Table 30. Maximum lengths for names by user registry and the optimum length across user registries (continued)

Name	IBM Security Directory Server	IBM z/OS Security Server	Novell eDirectory Server	Sun Java System Directory Server	Microsoft Active Directory Server	Active Directory Lightweight Directory Service (AD LDS)	Optimum length
Single sign-on resource name	240	240	240	240	60	240	60
Single sign-on resource description	1024	1024	1024	1024	1024	1024	1024
Single sign-on user ID	240	240	240	240	60	240	60
Single sign-on password	unlimited	unlimited	unlimited	unlimited	256	unlimited	256
Single sign-on group name	240	240	240	240	60	240	60
Single sign-on group description	1024	1024	1024	1024	1024	1024	1024
Action name	1	1	1	1	1	1	1
Action description, action type	unlimited	unlimited	unlimited	unlimited	unlimited	unlimited	unlimited
Object name, object description	unlimited	unlimited	unlimited	unlimited	unlimited	unlimited	unlimited
Object space name, object space description	unlimited	unlimited	unlimited	unlimited	unlimited	unlimited	unlimited
ACL name, ACL descriptions	unlimited	unlimited	unlimited	unlimited	unlimited	unlimited	unlimited
POP name, POP description		unlimited	unlimited	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.

If you configure Security Access Manager to use multiple Active Directory domains, the maximum length of the user identity and group name does not include the domain suffix. When using multiple domains, the format of a user identity is *user_id@domain_suffix*. The maximum length of 64 characters applies only to the *user_id* portion. When using an email address or other format for the Security Access Manager user identity in the Active Directory, then the maximum name length remains the same but it includes the suffix.

Although the lengths of some names can be unlimited, the excessive lengths can result in a policy that is difficult to manage and might result in poor system performance. Choose maximum values that are logical for your environment.

Appendix E. pdadmin to Web Portal Manager equivalents

This appendix shows the mapping of the administration **pdadmin** commands to Web Portal Manager.

Information about the **pdadmin** utility can be found in the *IBM Security Access Manager for Web: Command Reference*.

Table 31. Mapping between the pddamin utility and Web Portal Manager

pdadmin utility	Web Portal Manager
acl attach *object_name acl_name*	**ACL** → **List ACL** → click ACL name → **Attach** tab → **Attach** → type protected object path → **Attach**
acl create *acl_name*	**ACL** → **Create ACL** → complete the form → **Create**
acl delete *acl_name*	**ACL** → **List ACL** → select ACL names → **Delete**
acl detach *object_name*	**ACL** → **List ACL** → click ACL name → **Attach** tab → select protected object → **Detach**
acl find *acl_name*	**ACL** → **List ACL** → click ACL name → **Attach** tab
acl list	**ACL** → **List ACL**
acl list *acl_name* **attribute**	**ACL** → **List ACL** → click ACL name → **Extended Attribute** tab
acl modify *acl_name* **delete attribute** *attribute_name*	**ACL** → **List ACL** → select ACL name → **Extended Attribute** tab → select attributes → **Delete**
acl modify *acl_name* **delete attribute** *attribute_name attribute_value*	Not supported
acl modify *acl_name* **description** *description*	**ACL** → **List ACL** → click ACL name → modify description → **Set**
acl modify *acl_name* **remove any-other**	**ACL** → **List ACL** → click ACL name → select **Any-other** → **Delete**
acl modify *acl_name* **remove group** *group_name*	**ACL** → **List ACL** → click ACL name → select group name → **Delete**
acl modify *acl_name* **remove unauthenticated**	**ACL** → **List ACL** → click ACL name → select **Unauthenticated** → **Delete**
acl modify *acl_name* **remove user** *user_name*	**ACL** → **List ACL** → click ACL name → select user name → **Delete**
acl modify *acl_name* **set any-other** *permissions*	**ACL** → **List ACL** → click ACL name → select **Any-other** → **Create** → select permissions → **Apply**
acl modify *acl_name* **set attribute** *attribute_name attribute_value*	**ACL** → **List ACL** → click ACL name → **Extended Attribute** tab → **Create** → complete the form → **Apply**
acl modify *acl_name* **set group** *group_name permissions*	**ACL** → **List ACL** → click ACL name → **Create** → select **Group** → specify group name → select permissions → **Apply**

Table 31. Mapping between the pddamin utility and Web Portal Manager (continued)

pdadmin utility	**Web Portal Manager**
acl modify *acl_name* **set unauthenticated** *permissions*	**ACL** → **List ACL** → click ACL name → **Create** → select **Unauthenticated** → select permissions → **Apply**
acl modify *acl_name* **set user** *user_name permissions*	**ACL** → **List ACL** → click ACL name → **Create** → select **User** → specify user name → select permissions → **Apply**
acl show *acl_name*	**ACL** → **List ACL** → click ACL name
acl show *acl_name* **attribute** *attribute_name*	**ACL** → **List ACL** → click ACL name → **Extended Attribute** tab
action create *name description action_type*	**ACL** → **List Action Groups** → click `primary` action group → **Create** → complete the form → **Create**
action create *name description action_type action_group_name*	**ACL** → **List Action Groups** → click action group → **Create** → complete the form → **Create**
action delete *name*	**ACL** → **List Action Groups** → click `primary` action group → select actions → **Delete**
action delete *name action_group_name*	**ACL** → **List Action Groups** → click action group → select actions → **Delete**
action group create *action_group_name*	**ACL** → **Create Action Group** → type group name → **Create**
action group delete *action_group_name*	**ACL** → **List Action Groups** → select action groups → **Delete**
action group list	**ACL** → **List Action Groups**
action list	**ACL** → **List Action Groups** → click `primary` action group
action list *action_group_name*	**ACL** → **List Action Groups** → click action group
admin show configuration	Not supported
authzrule attach *object_name ruleid*	**AuthzRule** → **List AuthzRule** → click authorization rule name → **Attach** tab → **Attach** → type protected object path → **Attach**
authzrule create *ruleid* { *ruletext*} [**–desc** *description*] [**–failreason** *failreason*]	**AuthzRule** → **Create AuthzRule** → complete the form → **Create**
authzrule delete *ruleid*	**AuthzRule** → **List AuthzRule** → select authorization rule name → **Delete**
authzrule detach *object_name*	**AuthzRule** → **List AuthzRule** → click authorization rule name → **Attach** tab → select object names → **Detach**
authzrule find *ruleid*	**AuthzRule** → **List AuthzRule** → click authorization rule name → **Attach** tab
authzrule list	**AuthzRule** → **List AuthzRule**
authzrule modify *ruleid* { **ruletext** *rule_text* | **description** *description* | **failreason** *failreason*	**AuthzRule** → **List AuthzRule** → click authorization rule name → modify fields → **Apply**
authzrule show *ruleid*	**AuthzRule** → **List AuthzRule** → click authorization rule name
config modify svrpassword *config_file password*	Not supported

Table 31. Mapping between the pddamin utility and Web Portal Manager (continued)

`pdadmin` utility	Web Portal Manager
`config modify` **keyvalue set** [**–obfuscate**] *config_file stanza key value*	Not supported
`config modify` **keyvalue append** [**–obfuscate**] *config_file stanza key value*	Not supported
`config modify` **keyvalue remove** *config_file stanza key value*	Not supported
`config modify` **keyvalue remove** *config_file stanza key*	Not supported
`config show` *config_file stanza key*	Not supported
`context show`	Not supported
`domain create` *domain domain_admin_id domain_admin_password* [**–desc** *description*]	**Secure Domain** → **Create Secure Domain** → complete the form → **Create**
`domain delete` *domain* [**–registry**]	**Secure Domain** → **List Secure Domain** → select secure domain names → **Delete**
`domain list`	**Secure Domain** → **List Secure Domain**
`domain modify` *domain* **description** *description*	**Secure Domain** → **List Secure Domain** → click secure domain name → modify description → **Apply**
`domain show` *domain*	**Secure Domain** → **List Secure Domain** → click secure domain name
`errtext` *error_number*	Not supported
`exit`	Not supported
`group create` *group_name dn cn* [*group_container*]	**Group** → **Create Group** → complete the form → **Create**
`group delete` [**–registry**] *group_name*	**Group** → **Search Groups** → type pattern and maximum results → **Search** → select group names → **Delete**
`group import` *group_name dn* [*group_container*]	**Group** → **Import Group** → complete the form → **Import**
`group list` *pattern max_return*	**Group** → **Search Groups** → type pattern and maximum results → **Search**
`group list-dn` *pattern max_return*	Not supported
`group modify` *group_name* **add** *user* `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**
`group modify` *group_name* **description** *description*	**Group** → **Search Groups** → type pattern and maximum results → **Search** → click group name → type description → **Apply**
`group modify` *group_name* **remove** *user* `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**
`group show` *group_name*	**Group** → **Search Groups** → type pattern and maximum results → **Search** → click group name

Table 31. Mapping between the pddamin utility and Web Portal Manager (continued)

`pdadmin` utility	Web Portal Manager
`group show-dn` *dn*	Not supported
`group show-members` *group_name*	**Group** → **Search Groups** → type pattern and maximum results → **Search** → click group name → **Members** tab
`help` {*topic* | *command*}	Not supported
`login` **–a** *admin_id* **–p** *password* [**–d** *domain* | **–m**]	Not supported
`login` **–l**	Not supported
`logout`	Not supported
`object access` *object_name permissions*	Not supported
`object create` *object_name description type* **ispolicyattachable** {`yes` | `no`}	**Object Space** → **Create Object** → complete the 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.
`object delete` *object_name*	**Object Space** → **Browse Object Space** → expand and click object name → **Delete**
`object exists` *object_name*	Not supported
`object list`	**Object Space** → **Browse Object Space** → expand
`object list` *object_name*	**Object Space** → **Browse Object Space** → expand and click object name
`object list` *object_name* **attribute**	**Object Space** → **Browse Object Space** → expand and click object name → **Extended Attributes** tab
`object listandshow` *object_name*	Not supported
`object modify` *object_name* **delete** *attribute_name*	**Object Space** → **Browse Object Space** → expand and click object name → **Extended Attributes** tab → select attribute → **Delete**
`object modify` *object_name* **delete** *attribute_name attribute_value*	Not supported
`object modify` *object_name* **set attribute** *attribute_name attribute_value*	**Object Space** → **Browse Object Space** → expand and click object name → **Extended Attributes** tab → **Create** → complete the form → **Apply**
`object modify` *object_name* **set description** *description*	**Object Space** → **Browse Object Space** → expand and click object name → modify description → **Apply**
`object modify` *object_name* **isPolicyAttachable** {`yes` | `no`}	**Object Space** → **Browse Object Space** → expand and click object name → select or clear check box→ **Apply**
`object modify` *object_name* **type** *type*	Not supported
`object show` *object_name*	**Object Space** → **Browse Object Space** → expand and click object name

Table 31. Mapping between the pddamin utility and Web Portal Manager (continued)

pdadmin utility	**Web Portal Manager**
object show *object_name* **attribute** *attribute_name*	**Object Space** → **Browse Object Space** → expand and click object name → **Extended Attributes** tab
objectspace create *objectspace_name*	**Object Space** → **Create Object Space** → complete the form → **Create**
objectspace delete *objectspace_name*	**Object Space** → **Browse Object Space** → click object space name → **Delete**
objectspace list	**Object Space** → **Browse Object Space**
policy get *policy_name*	**User** → **Show Global User Policy**
policy get *policy_name* **–user** *user_name*	**User** → **Search Users** → type pattern and maximum results → **Search** → click user name → **Policy** tab
policy set *policy_name policy_value*	**User** → **Show Global User Policy** → modify value → **Apply**
policy set *policy_name policy_value* **–user** *user_name*	**User** → **Search Users** → type pattern and maximum results → **Search** → click user name → **Policy** tab → modify value → **Apply**
pop attach *object_name pop_name*	**POP** → **List POP** → click POP name → **Attach** tab → **Attach** → type protected object path → **Attach**
pop create *pop_name*	**POP** → **Create POP** → complete the form → **Create**
pop delete *pop_name*	**POP** → **List POP** → select POP names → **Delete**
pop detach *object_name*	**POP** → **List POP** → click POP name → **Attach** tab → select object → **Detach**
pop find *pop_name*	**POP** → **List POP** → click POP name → **Attach** tab
pop list	**POP** → **List POP**
pop list *pop_name*	**POP** → **List POP** → click POP name
pop list *pop_name* **attribute**	**POP** → **List POP** → click POP name → **Extended Attributes** tab
pop modify *pop_name* **delete attribute** *attribute_name*	**POP** → **List POP** → click POP name → **Extended Attributes** tab → select attributes → **Delete**
pop modify *pop_name* **delete attribute** *attribute_name attribute_value*	Not supported
pop modify *pop_name* **set attribute** *attribute_name attribute_value*	**POP** → **List POP** → click POP name → **Extended Attributes** tab → **Create** → complete the form → **Apply**
pop modify *pop_name* **set audit-level** {all | none | *audit_level_list*}	**POP** → **List POP** → click POP name → select or clear appropriate check boxes → **Apply**
pop modify *pop_name* **set description** *description*	**POP** → **List POP** → click POP name → modify description → **Apply**
pop modify *pop_name* **set ipauth add** *network netmask authentication_level*	**POP** → **List POP** → click POP name → **IP Auth** tab → **Create** → type the network, net mask, and authentication level → **Apply**

Table 31. Mapping between the pddamin utility and Web Portal Manager (continued)

`pdadmin` utility	Web Portal Manager
`pop modify` *pop_name* **set ipauth add** *network netmask* **forbidden**	**POP** → **List POP** → click POP name → **IP Auth** tab → **Create** → type network and net mask and select **Forbidden** check box → **Apply**
`pop modify` *pop_name* **set ipauth anyothernw** *authentication_level*	**POP** → **List POP** → click POP name → **IP Auth** tab → **Create** → select **Any Other Network** check box and type authentication level → **Create**
`pop modify` *pop_name* **set ipauth anyothernw forbidden**	**POP** → **List POP** → click POP name → **IP Auth** tab → **Create** → select **Any Other Network** and **Forbidden** check boxes → **Create**
`pop modify` *pop_name* **set ipauth remove** *network netmask*	**POP** → **List POP** → click POP name → **IP Auth** tab → select IP authorization entries → **Delete**
`pop modify` *pop_name* **set qop** {`none` | `integrity` | `privacy`}	**POP** → **List POP** → click POP name → select appropriate quality of protection → **Apply**
`pop modify` *pop_name* **set tod-access** {`anyday` | `weekday` | *day_list*}:{`anytime` | *time_spec-time_spec*}[:`utc` | `local`]	**POP** → **List POP** → click POP name → define time of day access → **Apply**
`pop modify` *pop_name* **set warning** {`yes` | `no`}	**POP** → **List POP** → click POP name → select or clear **Warn Only On Policy Violation** check box → **Apply**
`pop show` *pop_name*	**POP** → **List POP** → click POP name
`pop show` *pop_name* **attribute**	**POP** → **List POP** → click POP name → **Extended Attributes** tab
`quit`	Not supported
`rsrc create` *resource_name* [**–desc** *description*]	**GSO Resource** → **Create GSO** → complete the form → **Create**
`rsrc delete` *resource_name*	**GSO Resource** → **List GSO** → select resources → **Delete**
`rsrc list`	**GSO Resource** → **List GSO**
`rsrc show` *resource_name*	**GSO Resource** → **List GSO** → click resource
`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** → complete the form → **Create**
`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** → complete the form → **Create**
`rsrccred delete` *resource_name* **rsrctype** {`web` | `group`} **user** *user_name*	**User** → **Search Users** → **Search** → click user name → **GSO Credentials** tab → select credentials → **Delete**
`rsrccred delete` *resource_group_name* **rsrctype** {`web` | `group`} **user** *user_name*	**User** → **Search Groups** → **Search** → click user name → **GSO Credentials** tab → select credentials → **Delete**
`rsrccred list` **user** *user_name*	**User** → **Search Users** → **Search** → click user name → **GSO Credentials** tab

Table 31. Mapping between the pddamin utility and Web Portal Manager (continued)

pdadmin utility	**Web Portal Manager**
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**
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**
rsrccred show *resource_name* **rsrctype** {web | group} **user** *user_name*	**User** → **Search Users** → **Search** → click user name → **GSO Credentials** tab
rsrccred show *resource_group_name* **rsrctype** {web | group} **user** *user_name*	**User** → **Search Groups** → **Search** → click user name → **GSO Credentials** tab
rsrcgroup create *resource_group_name* [**–desc** *description*]	**GSO Resource** → **Create GSO Group** → complete the form → **Create**
rsrcgroup delete *resource_group_name*	**GSO Resource** → **List GSO Groups** → select resource groups → **Delete**
rsrcgroup list	**GSO Resource** → **List GSO Groups**
rsrcgroup modify *resource_group_name* **add rsrcname** *resource_name*	**GSO Resource** → **List GSO Groups** → select resource group → select members → **Add**
rsrcgroup modify *resource_group_name* **remove rsrcname** *resource_name*	**GSO Resource** → **List GSO Groups** → select resource group → select members → **Remove**
rsrcgroup show *resource_group_name*	**GSO Resource** → **List GSO Groups** → select resource group
server list	Not supported
server listtasks *server_name*	Not supported
server replicate *server_name*	Not supported
server show *server_name*	Not supported
server task *server_name* {help | stats | trace}	Not supported
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.*
user create [**–gsouser**] [**–no-password-policy**] *user_name dn cn sn password* [*group1* [*group2* ...]]	**User** → **Create User** → complete the form → **Create**
user delete [**–registry**] *user_name*	**User** → **Search Users** → type pattern and maximum results → **Search** → select user names → **Delete**
user import [**–gsouser**] *user_name dn* [*group_name*]	**User** → **Import User** → complete the form → **Import**
user list *pattern max_return*	**User** → **Search Users** → type pattern and maximum results → **Search**
user list-dn *pattern max_return*	Not supported
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**

Table 31. Mapping between the pddamin utility and Web Portal Manager (continued)

`pdadmin` utility	Web Portal Manager
`user modify` *user_name* **password** *password*	**User** → **Search Users** → type pattern and maximum results → **Search** → click user name → modify password→ **Apply**
`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**
`user show` *user_name*	**User** → **Search Users** → type pattern and maximum results → **Search** → click user name
`user show-dn` *dn*	Not supported
`user show-groups` *user_name*	**User** → **Search Users** → type pattern and maximum results → **Search** → click user name → **Groups** tab

Appendix F. Managing user registries

This appendix contains a subset of user registry tasks that are specific to installing Security Access Manager.

You might need more information about common administrative tasks for your particular registry (tasks that are not specific to Security Access Manager). See the documentation that came with your user registry product.

LDAP-specific tasks

LDAP is a protocol that runs over TCP/IP. The LDAP protocol standard includes low-level network protocol definitions plus data representation and handling.

A directory that is accessible through LDAP is commonly termed an LDAP directory. An example of an LDAP server product is the Security Directory Server, which is included with Security Access Manager.

This section contains the following topics:

LDAP failover configuration

LDAP failover configuration makes use of the Lightweight Directory Access Protocol (LDAP) standard method for accessing and updating information in a directory.

Directories are accessed with the client/server model of communication. Any server that implements LDAP is an LDAP server. The LDAP distributed architecture supports scalable directory services with server replication capabilities. Server replication improves the availability of a directory service.

Security Directory Server replication is based on a master-subordinate model. Sun Java System Directory Server replication is based on a supplier/consumer model, which Security Access Manager still treats as a master-subordinate or peer-to-peer relationship.

Active Directory Lightweight Directory Service (AD LDS) replication is based on membership in a *configuration set*, which is a group of AD LDS instances that share and replicate a common configuration partition and schema partition. AD LDS uses a multi-master form of replication, which means that any instance in the configuration set is writable and propagates the changes to all other instances in the configuration set.

Note: AD LDS instances cannot replicate with Active Directory. They replicate on a schedule that is independent of the Active Directory replication schedule, even when AD LDS is running in an Active Directory domain.

Security Access Manager treats each AD LDS instance in a configuration set as a replica. The Access Manager directory partition that contains the `secAuthorityInfo` subtree must be replicated to each of the AD LDS instances in the configuration set. The default replication schedule for AD LDS is one time per hour. This schedule can be changed, but the most frequent rate at which AD LDS replicates is four times an hour. Updates to one instance in a configuration set are not propagated for at least 15 minutes. Therefore, when Security Access Manager is used with AD LDS, configure one instance in the configuration set to have a higher read/write preference than all other instances. This way, updates are directed to the AD LDS instance with the highest preference. No other instances are used as failover unless the preferred instance is down.

For information about setting the AD LDS replication schedule, see the *IBM Security Access Manager for Web: Installation Guide*. To set preference values, see "Preference values for replica LDAP servers" on page 310.

Note: For SSL, ensure that the same certificate authority issues the AD LDS certificate for each instance in the configuration set. This way, Security Access Manager can validate the AD LDS certificate from each instance. If the AD LDS instances in the configuration set are on the same system, the instances can share the certificate.

For a generic LDAP server, the failover configuration depends on the specific LDAP server. The LDAP server recognizes the concept of master-subordinate, and Security Access Manager can use this replication support. For information about whether your LDAP server supports replication in this manner, see the documentation for your LDAP server.

The combination of a master server and multiple replicated servers helps to ensure that directory data is always available when needed. If any server fails, the directory service continues to be available from another replicated server. Security Access Manager supports this replication capability.

The master-subordinate replication model

Replication involves two types of directories: master/peer and replica.

LDAP refers to the master as the master server and to the replica as the replica server. Even when peer-to-peer replication is being used, the peer servers can be considered "masters" for the Security Access Manager perspective. All updates are made on the master server and these updates are later propagated to the replica servers. Each replica server directory contains a copy of the data in the master server directory.

Changes to the directory can be made only to a master server, which is always used for write operations to the directory. For Security Access Manager, these types of servers are configured as `readwrite` servers. Either the master or the replicas can be used for read operations. When the original master server is out of service for an extended period, a replica server can be promoted as a master server. The change enables write operations to the directory.

Security Access Manager failover capability for LDAP servers

When it starts, Security Access Manager connects to the LDAP master server indicated by the host key in the `ldap.conf` configuration file.

If the LDAP master server is down for any reason, the Security Access Manager server must be able to connect to an available LDAP replica server for any read operations. For Security Access Manager, these types of servers are configured as `readonly` servers.

Many operations, especially those from regular users, are read operations. These include operations such as user authentication and sign-on to back-end junctioned web servers. After configuration, Security Access Manager fails over to a replica server when it cannot connect to the master server.

You can find the configuration parameters for LDAP failover in the [ldap] stanza of the ldap.conf configuration file. This configuration file is in one of the following operating system-specific locations:

On AIX, Linux, and Solaris operating systems
/opt/PolicyDirector/etc/ldap.conf

On Windows operating systems
install_path\etc\ldap.conf

Master server configuration

Security Directory Server supports a single read/write master LDAP server or multiple peer-to-peer read/write servers.

Sun Java System Directory Server supports multiple read/write LDAP servers. Security Access Manager treats the Sun Java System supplier server as the master server for configuration purposes.

The active configuration lines in the ldap.conf file represent the parameters and values for this master LDAP server. You determine these values during Security Access Manager configuration. For example:

```
[ldap]
enabled = yes
host = outback
port = 389
ssl-port = 636
max-search-size = 2048
```

Table 32. Master server configuration entities and values

Entity	Description
enabled	Security Access Manager uses an LDAP user registry. Values are yes and no.
host	The network name of the computer where the LDAP master server is located. This server is assumed to be a readwrite server with a preference of 5.
port	The TCP listening port of the LDAP master server.
ssl-port	The SSL listening port of the LDAP master server.
max-search-size	The Security Access Manager limit for an LDAP client search of database items. For example, a request for the Web Portal Manager to list users from the LDAP database might reach this limit.

You might change the LDAP database. For example, when you add a user account through the Web Portal Manager, Security Access Manager uses the read/write (master) LDAP server.

Replica server configuration

Security Directory Server supports one or more read-only replica LDAP servers.

Sun Java System Directory Server supports the existence of one or more read-only replica LDAP servers that are termed consumers.

You must add lines to the [ldap] stanza that identifies any replica servers available to Security Access Manager. Use the following syntax for each replica:

```
replica = ldap_server,port,type,preference
```

Table 33. Replica server configuration entities and values

Entity	Description
ldap-server	The network name of the LDAP replica server.
port	The port this server listens on. Generally, use 389 or 636.
type	The functionality of the replica server, which is either readonly or readwrite. Normally, use read-only. A read/write type would represent a master server.
preference	A number from 1 to 10. The server with the highest preference value is chosen for LDAP connections. See "Preference values for replica LDAP servers."

Example:

```
replica = replica1.ldap.tivoli.com,389,readonly,4
replica = replica2.ldap.tivoli.com,389,readonly,4
```

Changes to the ldap.conf file do not take effect until you restart Security Access Manager.

Preference values for replica LDAP servers

Each replica LDAP server must have a preference value (1 to 10) that determines its priority.

Priority is based on one of the following selections:

- The primary read-only access server
- A backup read-only server during a failover

The higher the number, the higher the priority. If the primary read-only server fails for any reason, the server with the next highest preference value is used. If two or more servers have the same preference value, a least-busy load balancing algorithm determines which one is selected.

Remember that the master LDAP server can function as both a read-only and a read/write server. For read-only access, the master server has a hardcoded default preference setting of 5. Use this preference setting to set replica servers at values higher or lower than the master to obtain the required performance. For example, with appropriate preference settings, you can prevent the master server from handling everyday read operations.

You can set hierarchical preference values to allow access to a single LDAP server with failover to the other servers. You can also set equal preferences for all servers and allow load balancing to dictate server selection.

Table 34 on page 311 illustrates some possible preference scenarios. "M" is a reference to the master (read-only/read-write) LDAP server; "R1", "R2" and "R3" are references to the replica (read-only) LDAP servers.

Table 34. Potential preference scenarios

M	R1	R2	R3	Failover preference
5	5	5	5	All servers have the same preference values. Load balancing determines which server is selected for each access operation.
5	6	6	6	The 3 replica servers have the same preference value. This value is higher than the master server value. Load balancing determines server selection among the 3 replicas. The master is used only if all 3 replica servers become unavailable.
5	6	7	8	Server 3 (with the highest preference value) becomes the primary server. If server 3 fails, server 2 becomes the primary server because it has the next highest preference value.

Preference values affect the read-only access to the LDAP database and the read-write access to the replica LDAP servers. Security Access Manager always uses the master (read/write) server when you need to change the LDAP database.

Some Security Access Manager daemons such as the policy server override the preference settings in their configuration files to indicate that the read/write server is preferred. This override occurs because those daemons usually make update operations that go to the master LDAP server.

Server polling

If an LDAP server does fail, Security Access Manager continuously polls the server to check for its return to active duty. The poll time is 10 seconds.

Valid characters for LDAP user and group names

You might use LDAP as the user registry. The set of valid characters allowed within a user or group name is determined by several Internet Engineering Task Force (IETF) Request for Comments (RFC).

Relevant RFCs include:

- *2253 Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*
- *2254 The String Representation of LDAP Search Filters*

The specific LDAP server can also dictate the validity of these characters.

In general, you can use special characters within a distinguished name. However, certain special characters require an additional escape character. The following special characters must be escaped when used in a distinguished name:

- Plus sign (+)
- Semicolon (;)
- Comma (,)
- Backward slash (\)
- Double quote (")
- Less than (<)
- Greater than (>)
- Pound sign (#)

For example, to create a user that contains a semicolon with the **pdadmin** utility:

```
pdadmin> user create "user;one" "cn=user\;one,o=tivoli,c=us"
"user;one" "user;one" password1
```

Note: Avoid the backward slash character (\) as part of a user or group name. For more information, see "Characters disallowed for user and group name" in appendix A of the *IBM Security Access Manager for Web: Command Reference*.

If you use special characters with the **pdadmin** utility, enclose each argument of the user or group command with double quotation marks. The double quotation marks allow the argument to be entered without being subject to interpretation by the operating system shell command processor.

Due to the variability of special character handling in general, avoid the use of special characters.

Applying Security Access Manager ACLs to new LDAP suffixes

The LDAP naming model is maintained in a hierarchical namespace known as the Directory Information Tree (DIT).

Many LDAP server products, such as Security Directory Server, which is included with Security Access Manager, and the Sun Java System Directory Server and Novell eDirectory, maintain the data of the DIT in a hierarchical namespace that is often represented as a tree structure. The top of the tree is termed a *naming context*. Sometimes, this naming context is called a *suffix* because it represents the ending portion of a distinguished name (DN). For example, the `c=us` suffix might be created to represent country-specific data within an organization. An entry within this suffix might have a DN similar to `cn=Joe Williams,ou=austin,o=ibm,c=us`. The set of suffixes that is maintained by the LDAP server can be configured with the vendor-specific LDAP administration tools.

When the Security Access Manager policy server is configured, it attempts to apply appropriate access controls in the form of Access Control Lists (ACLs) to each LDAP suffix that is in the LDAP server. This access control gives appropriate permissions to allow Security Access Manager to create and manage user and group information in these suffixes.

Note: The Security Access Manager policy server does not attempt to apply ACLs to each LDAP suffix when AD LDS is used as the user registry. Access to AD LDS registry entries is controlled by administration groups within AD LDS.

For LDAP server types other than AD LDS, an LDAP administrator might add an LDAP suffix after Security Access Manager is configured. To have Security Access Manager to manage users and groups in this new suffix, the administrator must apply the appropriate ACLs to the new suffix.

To apply the appropriate access controls to a newly created LDAP suffix, use the **ivrgy_tool** utility with the **add-acls** parameter. For more information, see "ivrgy_tool" in the *IBM Security Access Manager for Web: Command Reference*. Alternately, you can manually apply the following ACLs to each new suffix:

`cn=SecurityGroup,secAuthority=Default`
- Full access

`cn=ivacld-servers,cn=SecurityGroups,secAuthority=Default`
- read

- search
- compare
- write for the following attributes:
  - **secAcctValid**
  - **secPwdFailCountTime**
  - **secPwdFailures**
  - **secPwdLastChanged**
  - **secPwdLastFailed**
  - **secPwdLastUsed**
  - **secPwdUnlockTime**
  - **secPwdValid**

`cn=remote-acl-users,cn=SecurityGroups,secAuthority=Default`
- read
- search
- compare
- write for the following attributes:
  - **secAcctValid**
  - **secPwdFailCountTime**
  - **secPwdFailures**
  - **secPwdLastChanged**
  - **secPwdLastFailed**
  - **secPwdLastUsed**
  - **secPwdUnlockTime**
  - **secPwdValid**

When using a generic LDAP server, give the same access controls to the specified groups. For information about how to set access control for a generic LDAP server, see the documentation that is associated with the generic LDAP server.

If a Security Access Manager administrator created a domain other than the initial `\Management` domain, which is created during the configuration of the policy server, apply the following additional ACLs to the new suffix for each domain:

`cn=SecurityGroup,secAuthority=`*`domain_name`*`,cn=Subdomains, secAuthority=Default`
- Full access

`cn=ivacld-servers,cn=SecurityGroups,secAuthority=`*`domain_name`*`,cn=Subdomains, secAuthority=Default`
- read
- search
- compare
- write for the following attributes:
  - **secAcctValid**
  - **secPwdFailCountTime**
  - **secPwdFailures**
  - **secPwdLastChanged**
  - **secPwdLastFailed**
  - **secPwdLastUsed**
  - **secPwdUnlockTime**
  - **secPwdValid**

`cn=remote-acl-users,cn=SecurityGroups,secAuthority=`*`domain_name`*`, cn=Subdomains,secAuthority=Default`
- read
- search
- compare
- write for the following attributes:
  - **secAcctValid**

- **secPwdFailCountTime**
- **secPwdFailures**
- **secPwdLastChanged**
- **secPwdLastFailed**
- **secPwdLastUsed**
- **secPwdUnlockTime**
- **secPwdValid**

Where *domain_name* is the name of the additional administrative domain. For a list of domains, use the **`domain list`** command.

Example procedures

You can use these example procedures for either Security Directory Server or Sun Java System Directory Server, depending on the LDAP server type that is used.

About this task

The example procedures assume that there is a newly created `c=fr` suffix. Substitute your newly created suffix for this value in the procedures.

Security Directory Server:

This procedure describes how to apply the appropriate Security Access Manager access controls in Security Directory Server for a newly created suffix.

About this task

This procedure uses the Security Directory Server Web Administration Tool and assumes that this tool is installed and configured into the WebSphere Application Server.

Procedure

1. Access the login page with a supported web browser. The default login page is the following URL:

   http://*server_name*:12100/IDSWebApp/IDSjsp/Login.jsp

   Where *server_name* is the host name of the application server where the Web Administration Tool is installed.

   If the list of console server contains the LDAP server to be administered, select its host name and go to step 4 on page 315. If this list does not contain the server, add it as a console server.
2. Add an LDAP server to the list of console servers:
   a. Log on as the Console Admin. The default Console Admin identity is `superadmin` and the default password is `secret`.
   b. In the navigation area on the left, click **Console administration** and **Manage console servers**. This action presents a list of LDAP servers that are currently configured for administration.
   c. Click **Add** and type the host name and port number for the LDAP server to be administered.
   d. Click **OK** to add the server.
   e. Click **Close** to complete the action.
   f. From the navigation area, click **Logout**.

3. Access the login page with the URL in step 1 on page 314 and select from the list the LDAP server that you added.
4. In the Login window, type the LDAP server administrator in the **Username** field (for example, `cn=root`) and password in the **password** field, and click **Login**.
5. In the navigation area on the left, click **Directory management** and **Manage entries**. If you see the newly added suffix in the Manage entries window on the right, go to step 7. If you do not see the newly added suffix, add an entry for a newly created suffix.
6. Add a suffix:
   a. Click **Add** to display the Add an entry window.
   b. Select the appropriate structural object class for the newly added suffix. For the `c=fr` suffix, the appropriate object class is **country**.
   c. Click **Next** to display the Select auxiliary object classes window where you can add additional object classes appropriate for the entry type.
   d. Because this example does not use other object classes, click **Next** to define the selected structural object class.
   e. In the **Relative DN** field, type `c=fr` and leave the **Parent DN** field blank. The only required attribute is `c` for country. Enter the value `fr`, and click **Finish** to return to the Manage entries window. You now see the newly added suffix in the list of top-level entries.
7. In the Manage entries window:
   a. From the **Select** column, select the suffix.
   b. From the **Select Action** list, select **Edit ACL**.
   c. Click **Go** to display the Edit ACL window that shows the current ACLs on the suffix.
8. In the Edit ACL window:
   a. Click **Non-filtered ACLs**.
   b. Ensure that the **Propagate ACLs** option is selected.
   c. Click **Add** to display the Add access rights window.
9. In the Add access rights window:
   a. In the **Subject DN** (distinguished name) field, type `cn=SecurityGroup,secAuthority=Default`.
   b. Set the **Add child** right to **grant**.
   c. Set the **Delete entry** right to **grant**.
   d. Set the **normal**, **sensitive**, **critical**, **system** and **restricted** security classes to **grant** for the **read**, **write**, **search** and **compare** actions.
   e. Click **OK** to return to the Edit ACL window.
10. In the Edit ACL window, click **Add** to display the Add access rights window.
11. In the Add access rights window:
   a. In the **Subject DN** (distinguished name) field, type `cn=ivacld-servers,cn=SecurityGroups,secAuthority=Default`.
   b. Set the **Subject Type** to **group**.
   c. Set the **normal** security classes to **grant** for the **read**, **search** and **compare** actions.
   d. From the **Attributes** list, select **secAcctValid** and click **Define**. Repeat this step for each of the following attributes:
      - **secPwdFailCountTime**
      - **secPwdFailures**

- **secPwdLastChanged**
- **secPwdLastFailed**
- **secPwdLastUsed**
- **secPwdUnlockTime**
- **secPwdValid**

e. After defining these attributes, set each of these attributes to **grant** for the **read**, **write**, **search** and **compare** actions.

f. Click **OK** to return to the Edit ACL window.

If you have other domains that need domain ACLs, continue to step 12. If you have no further domains, this step completes the access control. Go to step 17 on page 317. This sample procedure has additional domains the require domain ACLs.

12. In the Edit ACL window, click **Add** to display the Add access rights window.

    In the Add access rights window:

    a. In the **Subject DN** (distinguished name) field, type `cn=SecurityGroup,secAuthority=`*`domain_name`*`,cn=Subdomains, secAuthority=Default`, where *domain_name* is the domain name that is protected.

    b. Set the **Add child** right to **grant**.

    c. Set the **Delete entry** right to **grant**.

    d. Set the **normal**, **sensitive**, **critical**, **system** and **restricted** security classes to **grant** for the **read**, **write**, **search**, and **compare** actions.

    e. Click **OK** to return to the Edit ACL window.

13. In the Edit ACL window, click **Add** to display the Add access rights window.

14. In the Add access rights window:

    a. In the **Subject DN** (distinguished name) field, type `cn=ivacld-servers,cn=SecurityGroups,secAuthority=`*`domain_name`*`,cn=Subdomains, secAuthority=Default`, where *domain_name* is the domain name that is protected.

    b. Set the **Subject Type** to **group**.

    c. Set the **normal** security classes to **grant** for the **read**, **search** and **compare** actions.

    d. From the **Attributes** list, select **secAcctValid** and click **Define**. Repeat this step for each of the following attributes:
       - **secPwdFailCountTime**
       - **secPwdFailures**
       - **secPwdLastChanged**
       - **secPwdLastFailed**
       - **secPwdLastUsed**
       - **secPwdUnlockTime**
       - **secPwdValid**

    e. After defining these attributes, set each of these attributes to **grant** for the **read**, **write**, **search** and **compare** actions.

    f. Click **OK** to return to the Edit ACL window.

15. In the Edit ACL window, click **Add** to display the Add access rights window.

16. In the Add access rights window:

    a. In the **Subject DN** (distinguished name) field, type `cn=remote-acl-users,cn=SecurityGroups,secAuthority=`*`domain_name`*`,cn=Subdomains, secAuthority=Default`, where *domain_name* is the domain name that is protected.

b. Set the **Subject Type** to **group**.

c. Set the **normal** security classes to **grant** for the **read**, **search** and **compare** actions.

d. From the **Attributes** list, select **secAcctValid** and click **Define**. Repeat this step for each of the following attributes:
   - **secPwdFailCountTime**
   - **secPwdFailures**
   - **secPwdLastChanged**
   - **secPwdLastFailed**
   - **secPwdLastUsed**
   - **secPwdUnlockTime**
   - **secPwdValid**

e. After defining these attributes, set each of these attributes to **grant** for the **read**, **write**, **search** and **compare** actions.

f. Click **OK** to return to the Edit ACL window.

This completes the addition of the access control for the suffix.

17. Click **Close**. You do not need to restart the LDAP server for the changes to take effect.
18. If you no longer need to use the Web Administration Tool, click **Logout**.

Sun Java System Directory Server:

This procedure applies the appropriate Security Access Manager access controls to the newly created suffix for Sun Java System Directory Server.

About this task

This procedure uses the Sun Java System Server Console.

Procedure

1. Start the Sun Java System Server Console with one of the following commands:
   - On AIX, Linux, and Solaris operating systems, enter the following command from the Sun Java System Directory Server installation directory:

     ```
     # ./startconsole
     ```
   - On systems that run the Solaris operating environment, when not using the Solaris packaged version:

     a. Change to the server root directory.

     b. Enter the following command:

     ```
     startconsole arguments
     ```

     c. Type `–h` to display a usage message that explains command-line arguments.
   - On Windows operating systems, select **Start** > **Programs** > **Sun Java System Server Products** > **Sun Java System Server Console Version *version_number***.
2. Log on to the Sun Java System Server Console:

   a. Type the LDAP administrator ID, which is usually `cn=Directory Manager`

   b. Type the password for this administrator.

   c. Click **OK**.
3. Select the Sun Java System Domain to be used by Security Access Manager.

4. Expand the server name and **Server Group**.
5. Select **Directory Server** to display the configuration information about the Sun Java System Directory server.
6. Click **Open** to access the Sun Java System Directory server.
7. Click the **Directory** tab. If the newly created suffix is displayed on the left pane, go to step 8. If the newly created suffix is not displayed, create an entry for the new suffix before applying access controls to the suffix.

   Note: These instructions assume an example suffix. Create the entry type and name that corresponds to your actual suffix.

   To create the entry:
   a. Right-click the name of the server at the top of the directory tree, and select **Object** > **New Root Object** to display a list of root suffixes.
   b. Select **c=fr** from the list of root suffixes. The New Object selection window is displayed.
   c. In the New Object selection window, scroll down and select **Country** as the new object entry type.
   d. Click **OK** to display the Property Editor window.
   e. In the **Country** field type `fr`, and click **OK**.
   f. Select **View** > **Refresh** to display the new suffix.
8. Right-click **c=fr** in the left pane, and select **Object** → **Set Access Permissions** to display the Manage Access Control for c=fr window.
9. Click **New** to display the Edit ACI for c=fr window.
10. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type `SECURITY GROUP – ALLOW ALL`.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.
    d. Replace the default ACI text with the following text:
    ```
    (target="ldap:///c=fr")(targetattr="*")
    (version 3.0; acl "SECURITY GROUP – ALLOW ALL";
    allow (all)
    groupdn = "ldap:///cn=SecurityGroup,secAuthority=Default";)
    ```
    e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.
    f. Click **OK** to return to the Manage Access Control for c=fr window.
11. Click **New** to display the Edit ACI for c=fr window.
12. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type `PD Servers GROUP – ALLOW READ`.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.
    d. Replace the default ACI text with the following text:
    ```
    (target="ldap:///c=fr")(targetattr="*")
    (version 3.0; acl "PD Servers GROUP – ALLOW READ";
    allow (read, search, compare)
    groupdn = "ldap:///cn=ivacld-servers,cn=SecurityGroups,
    secAuthority=Default";)
    ```
    e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.
    f. Click **OK** to return to the Manage Access Control for c=fr window.
13. Click **New** to display the Edit ACI for c=fr window.

14. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type SECURITY GROUP– ALLOW WRITE.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.
    d. Replace the default ACI text with the following text:
    ```
    (target="ldap:///c=fr")(targetattr="secAcctValid ||
    secPwdFailCountTime || secPwdFailures || secPwdLastChanged ||
    secPwdLastFailed || secPWDLastUsed || secPwdUnlockTime ||
    secPwdValid")
    (version 3.0; acl "SECURITY GROUP– ALLOW WRITE";
    allow (read, search, compare)
    groupdn = "ldap:///cn=ivacld-servers,cn=SecurityGroups,
    secAuthority=Default";)
    ```
    e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.
    f. Click **OK** to return to the Manage Access Control for c=fr window.
15. Click **New** to display the Edit ACI for c=fr window.
16. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type PD Remote ACL Users GROUP – ALLOW READ.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.
    d. Replace the default ACI text with the following text:
    ```
    (target="ldap:///c=fr")(targetattr="*")
    (version 3.0; acl "PD Remote ACL Users GROUP – ALLOW READ";
    allow (read, search, compare)
    groupdn = "ldap:///cn=remote-acl-users,cn=SecurityGroups,
    secAuthority=Default";)
    ```
    e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.
    f. Click **OK** to return to the Manage Access Control for c=fr window.
17. Click **New** to display the Edit ACI for c=fr window.
18. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type SECURITY GROUP– ALLOW WRITE.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.
    d. Replace the default ACI text with the following text:
    ```
    (target="ldap:///c=fr")(targetattr="secAcctValid ||
    secPwdFailCountTime || secPwdFailures || secPwdLastChanged ||
    secPwdLastFailed || secPWDLastUsed || secPwdUnlockTime ||
    secPwdValid")
    (version 3.0; acl "SECURITY GROUP– ALLOW WRITE";
    allow (read, search, compare)
    groupdn = "ldap:///cn=remote-acl-users,cn=SecurityGroups,
    secAuthority=Default";)
    ```
    e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.
    f. Click **OK** to return to the Manage Access Control for c=fr window.
19. Click **New** to display the Edit ACI for c=fr window.
20. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type PD Deny-Others.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.

   d. Replace the default ACI text with the following text:

      ```
      (targetfilter="(secAuthority=Default)")
      (version 3.0; acl "PD Deny-Others";
      deny(all)
      groupdn != "ldap:///cn=SecurityGroup,secAuthority=Default||
      ldap:///cn=remote-acl-users,cn=SecurityGroups,secAuthority=Default||
      ldap:///cn=ivacld-servers,cn=SecurityGroups,secAuthority=Default";)
      ```

   e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.
   f. Click **OK** to return to the Manage Access Control for c=fr window.

   If you have no further domains, this action completes the access control. You can skip to step 34 on page 322. If you have additional domains that require domain ACLs, continue with step 21.

21. Click **New** to display the Edit ACI for c=fr window.
22. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type `SECURITY GROUP – ALLOW ALL`.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.
    d. Replace the default ACI text with the following text:

       ```
       (target="ldap:///c=fr")(targetattr="*")
       (version 3.0; acl "SECURITY GROUP - ALLOW ALL;
       allow (all)
       groupdn = "ldap:///cn=SecurityGroup,secAuthority=domain_name,
       cn=Subdomains,secAuthority=Default";)
       ```

       where *domain_name* is the name of the domain that is protected.
    e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.
    f. Click **OK** to return to the Manage Access Control for c=fr window.
23. Click **New** to display the Edit ACI for c=fr window.
24. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type `PD Servers GROUP – ALLOW READ`.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.
    d. Replace the default ACI text with the following text:

       ```
       (target="ldap:///c=fr")(targetattr="*")
       (version 3.0; acl "PD Servers GROUP - ALLOW READ";
       allow (read, search, compare)
       groupdn = "ldap:///cn=ivacld-servers,cn=SecurityGroups,
       secAuthority=domain_name,cn=Subdomains,secAuthority=Default";)
       ```

       where *domain_name* is the name of the domain that is protected.
    e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.
    f. Click **OK** to return to the Manage Access Control for c=fr window.
25. Click **New** to display the Edit ACI for c=fr window.
26. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type `SECURITY GROUP– ALLOW WRITE`.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.
    d. Replace the default ACI text with the following text:

```
(target="ldap:///c=fr")(targetattr="secAcctValid ||
secPwdFailCountTime || secPwdFailures || secPwdLastChanged ||
secPwdLastFailed || secPWDLastUsed || secPwdUnlockTime ||
secPwdValid")
(version 3.0; acl "SECURITY GROUP– ALLOW WRITE";
allow (read, search, compare)
groupdn = "ldap:///cn=ivacld-servers,cn=SecurityGroups,
secAuthority=domain_name,cn=Subdomains,secAuthority=Default";)
```

   e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.
   f. Click **OK** to return to the Manage Access Control for c=fr window.

27. Click **New** to display the Edit ACI for c=fr window.
28. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type `PD Remote ACL Users GROUP – ALLOW READ`.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.
    d. Replace the default ACI text with the following text:

```
(target="ldap:///c=fr")(targetattr="*")
(version 3.0; acl "PD Remote ACL Users GROUP - ALLOW READ";
allow (read, search, compare)
groupdn = "ldap:///cn=remote-acl-users,cn=SecurityGroups,
secAuthority=domain_name,cn=Subdomains,secAuthority=Default";)
```

    where *domain_name* is the name of the domain that is protected.
    e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.
    f. Click **OK** to return to the Manage Access Control for c=fr window.
29. Click **New** to display the Edit ACI for c=fr window.
30. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type `SECURITY GROUP– ALLOW WRITE`.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.
    d. Replace the default ACI text with the following text:

```
(target="ldap:///c=fr")(targetattr="secAcctValid ||
secPwdFailCountTime || secPwdFailures || secPwdLastChanged ||
secPwdLastFailed || secPWDLastUsed || secPwdUnlockTime ||
secPwdValid")
(version 3.0; acl "SECURITY GROUP– ALLOW WRITE";
allow (read, search, compare)
groupdn = "ldap:///cn=remote-acl-users,cn=SecurityGroups,
secAuthority=domain_name,cn=Subdomains,secAuthority=Default";)
```

    e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.
    f. Click **OK** to return to the Manage Access Control for c=fr window.
31. Click **New** to display the Edit ACI for c=fr window.
32. In the Edit ACI for c=fr window:
    a. In the **ACI name** field, type `PD Deny-Others`.
    b. Highlight **All Users**, and click **Remove**.
    c. Click **Edit Manually**.
    d. Replace the default ACI text with the following text:

```
(targetfilter="(secAuthority=domain_name)")
(version 3.0; acl "PD Deny-Others";
deny(all)
groupdn != "ldap:///cn=SecurityGroup,secAuthority=Default||
ldap:///cn=SecurityGroup,secAuthority=domain_name,cn=Subdomains,
secAuthority=Default||
ldap:///cn=remote-acl-users,cn=SecurityGroups,secAuthority=domain_name,
cn=Subdomains,secAuthority=Default||
ldap:///cn=ivacld-servers,cn=SecurityGroups,secAuthority=domain_name,
cn=Subdomains,secAuthority=Default";)
```

where *domain_name* is the name of the domain that is protected.

e. Click **Check Syntax** to ensure validate the text. Correct errors until the syntax validates.

f. Click **OK** to return to the Manage Access Control for c=fr window.

33. If there are further domains, repeat steps 21 on page 320 to 32 on page 321 for each domain. When complete, continue with step 34.
34. Click **OK** to close the Manage Access Control for c=fr window.
35. Click **Console** > **Exit** to exit the console.

IBM z/OS Security Server:

This procedure applies the appropriate Security Access Manager access controls to the newly created suffix for IBM z/OS Security Server. Do this procedure after the Security Access Manager policy server is configured.

About this task

Instead of using the manual process described below, you can use the **ivrgy_tool** utility to update the ACLs on suffixes added after the initial policy server configuration. See "ivrgy_tool" in the *IBM Security Access Manager for Web: Installation Guide*.

These steps are specifically for the IBM Security Directory Server for z/OS Version 1.8. This LDAP server is the IBM z/OS LDAP Server.

Procedure

1. Add the new suffix to the LDAP server configuration file. See *z/OS LDAP Server Administration and Use* for your version of z/OS LDAP for details on how to update the server configuration file.
2. Restart the IBM z/OS LDAP Server.
3. To add an entry to the newly created suffix, do the following steps:

   a. Create an LDIF file. This example assumes that the new suffix is o=neworg,c=us:

   ```
   dn: o=neworg,c=us
   objectClass: organization
   objectClass: top
   o: neworg
   ```

   b. Use the appropriate LDIF file as input to the **ldapadd** command:

   ```
   ldapadd -h ldap_host -p ldap_port -D ldap_admin_dn -w ldap_admin_pwd
     -v -f ldif_filename
   ```

4. To apply the appropriate Security Access Manager access controls to the newly created suffix (*suffix*), do either of the following tasks:

   - If no additional Security Access Manager domains were created other than the initial management domain, complete the following steps:

     a. Create the following LDIF file:

```
dn: suffix
aclpropagate: TRUE
aclentry: group:cn=SecurityGroup,secAuthority=Default:object:ad:normal:\
rwsc:sensitive:rwsc:critical:rwsc:restricted:rwsc
aclentry: group:cn=ivacld-servers,cn=SecurityGroups,secAuthority=Defaul\
t:normal:rsc:at.userPassword:wc:at.secAcctValid:rwsc:
at.secPwdFailCountTime:rwsc:at.secPwd\
Failures:rwsc:at.secPwdLastChanged:rwsc:at.secPwdLastFailed:rwsc:at.sec\
PwdLastUsed:rwsc:at.secPwdUnlockTime:rwsc:at.secPwdValid:rwsc
aclentry: group:cn=remote-acl-users,cn=SecurityGroups,secAuthority=Defau\
lt:normal:rsc:at.secAcctValid:rwsc:at.secPwdFailCountTime:rwsc:at.secPwd\
Failures:rwsc:at.secPwdLastChanged:rwsc:at.secPwdLastFailed:rwsc:at.secP\
wdLastUsed:rwsc:at.secPwdUnlockTime:rwsc:at.secPwdValid:rwsc
entryowner: LDAP_admin_dn
entryowner: group:cn=SecurityGroup,secAuthority=Default
ownerpropagate: TRUE
```

The backward slash (\) at the end of a line indicates that this line combines with the next line, without any spaces.

b. Apply the updates in the LDIF file by using it as input to the **ldapmodify** command:

```
ldapmodify -h ldap_host -p ldap_port -D ldap_admin_dn
-w ldap_admin_pwd -v -f ldif_file
```

- If a domain was created in addition to the initial management domain, and if a new suffix is created, apply ACLs for each added domain. Complete the following steps:

a. Add ACLs to the default domain and added domain (*added_domain*) by creating an LDIF file similar to the following one:

```
dn: suffix
aclentry: group:cn=SecurityGroup,secAuthority=Default:object:ad:normal\
:rwsc:sensitive:rwsc:critical:rwsc:restricted:rwsc
aclentry: group:cn=ivacld-servers,cn=SecurityGroups,secAuthority=Defau\
lt:normal:rsc:at.userPassword:wc:at.secAcctValid:
rwsc:at.secPwdFailCountTime:rwsc:at.secP\wdFailures:rwsc:
at.secPwdLastChanged:rwsc:at.secPwdLastFailed:rwsc:at.\secPwdLastUsed:
rwsc:at.secPwdUnlockTime:rwsc:at.secPwdValid:rwsc
aclentry: group:cn=remote-acl-users,cn=SecurityGroups,secAuthority=Def\
ault:normal:rsc:at.secAcctValid:rwsc:at.secPwdFailCountTime:rwsc:at.se\
cPwdFailures:rwsc:at.secPwdLastChanged:rwsc:at.secPwdLastFailed:rwsc:a\
t.secPwdLastUsed:rwsc:at.secPwdUnlockTime:rwsc:at.secPwdValid:rwsc
aclentry: group:cn=SecurityGroup,secAuthority=added_domain,cn=Subdomai\
ns,secAuthority=Default:object:ad:normal:rwsc:sensitive:rwsc:critical:\
rwsc:restricted:rwsc
aclentry: group:cn=ivacld-servers,cn=SecurityGroups,secAuthority=added\
_domain,cn=Subdomains,secAuthority=
Default:normal:rsc:at.userPassword:wc:at.secAcctValid:\
rwsc:at.secPwdFailCountTime:rwsc:at.secPwdFailures:rwsc:at.secPwdLastC\
hanged:rwsc:at.secPwdLastFailed:rwsc:at.secPwdLastUsed:rwsc:at.secPwdU\
nlockTime:rwsc:at.secPwdValid:rwsc
aclentry: group:cn=remote-acl-users,cn=SecurityGroups,secAuthority=add\
ed_domain,cn=Subdomains,secAuthority=
Default:normal:rsc:at.userPassword:wc:at.secAcctVali\
d:rwsc:at.secPwdFailCountTime:rwsc:at.secPwdFailures:rwsc:at.secPwdLas\
tChanged:rwsc:at.secPwdLastFailed:rwsc:at.secPwdLastUsed:rwsc:at.secPw\
dUnlockTime:rwsc:at.secPwdValid:rwsc
aclpropagate: TRUE
entryowner: LDAP_admin_dn
entryowner: group:cn=SecurityGroup,secAuthority=Default
ownerpropagate: TRUE
```

b. Apply the updates in the LDIF file by using it as input to the **ldapmodify** command:

```
ldapmodify -h ldap_host -p ldap_port -D ldap_admin_dn -w ldap_admin_pwd
-v -f ldif_file
```

Note: The **ldapmodify** command returns an error if the following attributes and values are set by default for the newly added suffix:

```
aclpropagate: TRUE
entryowner: LDAP_admin_dn
ownerpropagate: TRUE
```

If the **ldapmodify** command returns the following error, remove these three attribute and value pairs from the LDIF file and run the **ldapmodify** command again:

```
ldapmodify: additional info: R004086 Entry 'suffix' already contains
  attribute 'attribute' with value 'value'
```

Setting the password history policy

If Security Directory Server is your user registry, you can use its password history policy.

About this task

For more information about setting the password policy that is used with Security Directory Server, see the *IBM Security Directory Server: Administration Guide*.

Procedure

1. Access the login page with a supported web browser. The default login page is the following URL:

   http://*server_name*:12100/IDSWebApp/IDSjsp/Login.jsp

   Where *server_name* is the host name of the application server where the Web Administration Tool is installed.
2. Select the LDAP host name to be managed and log on as an LDAP administrator (for example, `cn=root`). The Web Administration Tool starts.
3. In the navigation area, select **Server administration** > **Manage security properties**.
4. In the main window, select **Password validation**.
5. Set the minimum number of passwords that must be used before a password can be reused. Enter a number from 0 to 30. If you enter zero, a password can be reused without restriction.
6. Click **Apply**.
7. In the main window, click **Password policy**.
8. If not already enabled, set the **Password policy enabled** check box to enable password policy.
9. Click **OK**.

Novell-specific tasks

The Novell eDirectory can be configured as a Security Access Manager user registry.

This section describes a few steps that are unique to this configuration.

Updating the eDirectory schema with ConsoleOne

If you are installing a new Security Access Manager secure domain, the Security Access Manager schema is installed automatically on the Novell eDirectory Server (NDS) when the Security Access Manager policy server is configured.

About this task

Before you configure the policy server, modify Novell eDirectory with Novell's ConsoleOne directory management utility or iManager web-based administration console.

Note: The default Novell eDirectory schema assumes that the directory does not use the X.500 object classes of `inetOrgPerson` or `groupOfNames`. By default, these classes are mapped into the eDirectory classes of User and Group. Because Security Access Manager uses the `inetOrgPerson` and `groupOfNames` object classes for creating its own users and groups, modifications to the default eDirectory schema are required.

To update the eDirectory schema with the Novell iManager web-based administration console, see "Updating the eDirectory schema with Novell iManager" on page 326.

Procedure

1. Start the Novell ConsoleOne directory management utility.
2. Select the organization object within your Novell eDirectory tree. A list of objects is displayed on the right side of the ConsoleOne window.
3. Right click the **LDAP group** object (not LDAP server), and click **Properties** from the menu.
4. Click the **Class Map** tab and the table of LDAP class names. The Novell eDirectory class names are displayed.
5. Delete the entries with LDAP classes of `inetOrgPerson` and `groupOfNames`.
6. Click **Apply** and then click **Close**.
7. Click the **Attribute Map** tab and the table of LDAP attribute names. The Novell eDirectory attribute names are displayed.
8. Scroll through the table and find the Novell eDirectory attribute `member`. Check the value of the corresponding LDAP attribute. If the LDAP attribute value is `member`, then no change is needed. If the attribute is showing the default value of `uniqueMember`, you need to modify it as follows.
   - Click **Modify**. The Attribute Mapping window is displayed.
   - Change the **Primary LDAP Attribute** field from `uniqueMember` to `member`.
   - Change the **Secondary LDAP attribute** field from `member` to `uniqueMember`.
   - In the Attribute window, click **OK** to accept the changes.
9. If you are using Solaris, proceed to the next step. If you are using Windows NT, you might have to add another mapping for the LDAP attribute `ndsHomeDirectory` as follows:
   - On the right hand side of the Attribute Mappings window, click **Add**. The Attribute Mapping window repaints and is displayed again.
   - From the Novell eDirectory **NSD Attribute** field menu, click **Home Directory**.
   - In the **Primary LDAP Attribute** field, click `ndsHomeDirectory`.
   - In the Attribute Mapping window, click **OK** to accept the changes.

10. In the Properties window, click **OK**.

Updating the eDirectory schema with Novell iManager

If you install a new Security Access Manager secure domain, the Security Access Manager schema is installed automatically on the Novell eDirectory Server (NDS) when the Security Access Manager policy server is configured.

About this task

Before you configure the policy server, modify Novell eDirectory with Novell's ConsoleOne directory management utility or iManager web-based administration console.

Note: The default Novell eDirectory schema assumes that the directory does not use the X.500 object classes of `inetOrgPerson` or `groupOfNames`. By default, these classes are mapped into the eDirectory classes of User and Group. Because Security Access Manager uses the `inetOrgPerson` and `groupOfNames` object classes for creating its own users and groups, modifications to the default eDirectory schema are required.

Procedure

1. Launch the iManager web page and log on as the administrator for the Novell eDirectory tree to be updated.
2. Click the **Roles and Tasks** icon at the top of the iManager window to open the Roles and Tasks view.
3. In the Roles and Tasks navigation frame, expand the **LDAP** category.
4. In the expanded list, click the **LDAP Options** task.
5. On the LDAP Options page, click the LDAP Group listed.
6. Click **Class Map** to display the Novell eDirectory class to LDAP class mappings.
7. Remove mappings to `inetOrgPerson` and `groupOfNames`.
   a. Scroll through the list and look for mappings of eDirectory classes to the LDAP class `inetOrgPerson`.
   b. If a mapping exists, select the row and click the **Remove Mapping** icon to remove the mapping.
   c. Click **OK** in the pop-up window to confirm the removal of the mapping.
   d. Click **Apply** to apply the changes.
   e. Repeat this step to remove a mapping for the LDAP class `groupOfNames`.
8. Click **OK** to accept the changes.
9. Repeat steps 3-5 to return to the LDAP Group page.
10. Click **Attribute Map** to access the Novell eDirectory attribute to LDAP attribute mappings.
11. Scroll through the table and find the Novell eDirectory attribute member. Check the value of the corresponding LDAP attribute. If the LDAP attribute value is `member`, no change is needed. If the attribute is the default value of `uniqueMember`, modify it as follows:
    a. Select the row and click the **View/Edit Mapping** icon.
    b. Change the **Primary LDAP Attribute** field from `uniqueMember` to `member`.
    c. Change the **Secondary LDAP attribute** field from `member` to `uniqueMember`.
    d. Click **OK** in the pop-up window to confirm the change.

   e. Click **Apply** to apply the changes.

12. If you are using Solaris, proceed to the next step. If you are using Windows NT, you might need to add another mapping for the LDAP attribute ndsHomeDirectory. To add another mapping for the LDAP attribute ndsHomeDirectory:
    a. Click the **Add Mapping** icon in the right side of the window. A pop-up window to define the mapping is displayed.
    b. In the **eDirectory Attribute** field, select **Home Directory**.
    c. In the Primary LDAP Attribute field, type ndsHomeDirectory.
    d. Click **OK** to confirm the mapping and close the pop-up window.
13. Click **OK** in the Attribute Map window to accept the changes.

Novell eDirectory maintenance activities that can damage schema modifications applied by Security Access Manager

Novell eDirectory defines the object classes User and Group as part of its base schema.

Instances of these object classes are created by an eDirectory administrator when defining a user or a group. Both of these object classes are defined by eDirectory as *leaf nodes*. eDirectory adds an attribute X-NDS_NOT_CONTAINER '1' to each of these object class definitions that specifies they are not container objects. Not being a container object means that the objects cannot be defined beneath instances of these object classes.

Security Access Manager requires the ability to append its own objects beneath pre-existing eDirectory users and groups to import them and make them usable by Security Access Manager. When Security Access Manager adds its own object class definitions to the eDirectory schema, it also redefines the eDirectory User and Group object classes. Redefinition allows instances of these classes to be container objects. Novell eDirectory allows this change to its schema definition.

The following Novell eDirectory administrator actions cause Security Access Manager modification to the User object class to be undone. The Group object class is not affected.

- Running the eDirectory database repair tool **ndsrepair** with the **rebuild schema** option.
- Running Basic Repair from the iManager console and running **local database repair** with the **rebuild operational schema** option.
- Applying a patch update to Novell eDirectory.
- Upgrading Novell eDirectory to a more recent version.

To do any of these operations after Security Access Manager was configured into the eDirectory server, run the following command immediately. This action ensures that the definition of the User object class is restored.

```
ivrgy_tool(.exe) -h edir_server_name -p port -D edir_admin_dn
-w edir_admin_password schema
```

The **ivrgy_tool** utility can be found in one of the following Security Access Manager directories:

AIX, Linux, and Solaris operating systems
    /opt/PolicyDirector/sbin

Windows operating systems

```
c:\program files\tivoli\policy director\sbin
```

Security Access Manager does not add the `/sbin` directory to the system PATH. You must run the **`ivrgy_tool`** utility from the `/sbin` directory.

Index

Printed in USA