IBM® SecureWay® Policy Director

Administration Guide

Version 3 Release 0

IBM® SecureWay® Policy Director

Administration Guide

Version 3 Release 0

Note

Before using this information and the product it supports, read the general information under “Appendix B. Notices” on page 293.

First Edition (October 1999)

This edition applies to version 3, release 0, modification 0 of IBM® SecureWay®Policy Director product and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces version 2, release 2, modification 200 of IBM SecureWay Global Sign-On.

Contents

About this book

This book provides information about IBM® SecureWay® Policy Director, such as:

- Policy Director concepts, such as: authentication, authorization, and credentials acquisition.
- General administration tasks by using the Management Console
- WebSEAL administration
- NetSEAL administration
- NetSEAT administration
- Administration resources (the **ivadmin** command)

Who should read this book

This book is written for the administrator who will be managing Policy Director users, groups, GSO resources, GSO resource groups, proxy users, access control lists and permissions, and the object space.

The person administering Policy Director must also manage authentication, authorization, and credentials acquisition and must have some knowledge of these processes.

The administrator should already be knowledgeable in managing the IBM Distributed Computing Environment (DCE) and the IBM SecureWay Directory's lightweight directory access protocol (LDAP). The IBM SecureWay Directory and IBM Distributed Computing Environment servers are used by Policy Director and are included in the Policy Director product.

How this book is organized

This book is organized into the following sections:

- Chapters 1 through describe Policy Director concepts, such as an overview of Policy Director in "Chapter 1. Welcome to Policy Director" on page 1, "Chapter 2. Authentication and credentials acquisition" on page 15, and "Chapter 3. Understanding Authorization" on page 33.
- Chapters 4 through 12 discuss general Policy Director Administration tasks, such as:
  - "Chapter 4. Introducing the Management Console" on page 53
  - "Chapter 5. Managing user accounts and groups" on page 67
  - "Chapter 6. Managing GSO resources, resource groups, and resource credentials" on page 73
  - "Chapter 7. Understanding access control" on page 79
  - "Chapter 8. Applying access control" on page 105
  - "Chapter 9. Managing proxy users" on page 111
  - "Chapter 10. Managing the Policy Director servers" on page 121
  - "Chapter 11. Managing the authorization service" on page 131
  - "Chapter 12. Logging and auditing server activity" on page 143

Chapters 13 through 16 cover WebSEAL administration such as:

- "Chapter 13. WebSEAL: Setting up authentication" on page 157
- "Chapter 14. WebSEAL: General administration tasks" on page 175
- "Chapter 15. WebSEAL: smart junction administration" on page 185
- "Chapter 16. WebSEAL: Application integration" on page 215

Chapters 17 and 18 discuss these NetSEAL administration topics:
- “Chapter 17. NetSEAL: Overview” on page 223
- “Chapter 18. NetSEAL: General administration tasks” on page 235

Chapters 19 through 21 provide information about NetSEAT administration:
- “Chapter 19. NetSEAT: Overview” on page 245
- “Chapter 20. NetSEAT: General administration tasks” on page 249
- “Chapter 21. NetSEAT: Directory Services Broker” on page 263

This book includes appendixes concerning “Appendix A. Policy Director administration using ivadmin” on page 267 and “Appendix B. Notices” on page 293.

What conventions are used in this book

This book uses the following typographical conventions:

Convention	Meaning
bold	User interface elements such as menu names, menu choices, entry fields, icons, folders, list boxes, action buttons, push buttons, radio buttons, spin buttons, and check boxes. Bold highlighting is also used for notes and cautions.
`monospace`	Syntax, sample code, and text that the user must type.
Italic	Emphasis and first use of special terms that are relevant to Policy Director.
→	Shows a series of selections from a menu. For example: Select **File → Run** means click **File**, and then click **Run**.

Year 2000 readiness

These products are Year 2000 ready. When used in accordance with their associated documentation, they are capable of correctly processing, providing, and receiving date data within and between the twentieth and twenty-first centuries, provided that all products (for example, hardware, software, and firmware) used with the products properly exchange accurate date data with them.

Service and support

Contact IBM for service and support for all the products included in the IBM SecureWay FirstSecure offering. Some of these products may refer to non-IBM support. If you obtain these products as part of the FirstSecure offering, contact IBM for service and support.

Prerequisite and related information

See the following documentation for more information on Policy Director prerequisite and related products:

IBM SecureWay Policy Director

Documentation in PDF format: The following books are related to Policy Director and are available in PDF format under /doc on the *IBM SecureWay Policy Director Version 3.0* CD:

This book, *IBM SecureWay Policy Director Administration Guide, Version 3.0*

Documentation in printed format: The following book is also related to Policy Director and is available in printed format and available in the product package:

IBM SecureWay Policy Director Up and Running, Version 3.0 (SCT6-3KNA-00)

Documentation available from the Web: The following books also relate to Policy Director and are available from the Policy Director Web page in PDF format:

- *IBM SecureWay Policy Director Up and Running, Version 3.0 (SCT6-3KNA-00)*
- *IBM SecureWay Policy Director Programming Guide and Reference, Version 3.0*

IBM SecureWay FirstSecure

The following document relates to IBM SecureWay FirstSecure:

- *IBM SecureWay FirstSecure Planning and Integration, Version 2.0 (S564-8D11-00)*

  This book describes FirstSecure and the products that make up FirstSecure. This book helps you start planning to use all the IBM SecureWay products.

IBM Distributed Computing Environment

The following documents describe how to install DCE and are available on the IBM SecureWay Policy Director Security Services CD in PDF format under /doc or at the DCE Web site.

IBM DCE for Windows NT

IBM Distributed Computing Environment for Windows NT Quick Beginnings, Version 2.2, available at the following Web address:

`http://www.software.ibm.com/network/dce/library/publications/dcent.html`

This book describes Distributed Computing Environment (DCE) for Windows NT, Version 2.2, and explains how to plan for, install, and configure the product.

IBM DCE for AIX

IBM Distributed Computing for AIX Quick Beginnings Version 2.2, is available at the following Web address:

`http://www.software.ibm.com/network/dce/library/publications/dceaix.html`

This book describes the IBM Distributed Computing Environment for AIX, Version 2.2 (DCE 2.2 for AIX), and explains how to plan for, install, and configure the product.

Transarc DCE for Solaris

The *Transarc DCE Version 2.0 Release Notes* and the *Installation and Configuration Guide* are available at the following Web address:

`http://www.transarc.com/Library/documentation/dce/2.0/index.html`

The *Transarc DCE Version 2.0 Release Notes* documents the following information about Transarc DCE software and documentation:

- Differences between OSF DCE and the DCE DFS product
- Differences between Version 2.0 and Version 1.1 of DCE DFS
- Known defects and limitations that are associated with DCE DFS

The *Installation and Configuration Guide* provides instructions for installing, configuring, and upgrading the DCE DFS Version 2.0 product.

IBM SecureWay Directory

The following book contains installation and configuration information for IBM SecureWay Directory (LDAP):

- *IBM SecureWay Directory Installation and Configuration, Version 3.1.1*

  There is a separate version of this book in HTML format for each of the supported operating systems. The book for each operating system is on the appropriate CD. The CDs are:

  – *IBM SecureWay Directory Version 3.1.1 for NT*
  – *IBM SecureWay Directory Version 3.1.1 for AIX*
  – *IBM SecureWay Directory Version 3.1.1 for Solaris*

The following documents are also available online for IBM SecureWay Directory:

- *IBM SecureWay Directory Client SDK Programming Reference*
- *IBM SecureWay Directory Server Plug-ins Reference*

See the *IBM SecureWay Policy Director Up and Running* book for more information on how to access the LDAP documentation.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other IBM SecureWay Policy Director documentation, visit our Policy Director home page at:

`http://www.ibm.com/software/security/policy/library`

There you will find the feedback page where you can enter comments and send them. You will also find information about last-minute updates to Policy Director.

Information about updates to other IBM SecureWay FirstSecure products is available at the following Web address:

`http://www.ibm.com/software/security/firstsecure/library`

Chapter 1. Welcome to Policy Director

IBM SecureWay Policy Director (Policy Director) is a complete authorization solution for corporate Web, client/server, and legacy applications. Policy Director authorization allows an organization to securely control user access to protected information. You use Policy Director in conjunction with standard Internet-based applications to build highly secure and well-managed intranets.

This chapter includes:

Understanding enterprise network security

Many organizations now value the public Internet and private intranets as effective and vital mediums for global communication. Electronic commerce is rapidly becoming an essential component of many business marketing strategies. Educational institutions rely on the Internet for long-distance learning. Online services allow individuals to send electronic mail and to tap the Web's vast encyclopedia of resources. Traditional applications, such as Telnet or POP3, still prevail as important network services.

Businesses are realizing that they can use Internet technologies to enhance supply-chain relationships, facilitate collaboration with business partners, and provide increased customer connectivity. Businesses can do these things, provided they can expose corporate resources with a high degree of security.

Businesses want to use the Internet as a global commercial and distribution vehicle. However, the lack of proven security policy mechanisms and management systems has hindered these businesses.

Policy Director is an information policy-management solution that provides organizations with centralized network security services. With centralized network security services, you can put into effect and maintain consistent user and policy information.

Network security terminology and definitions

The following network security services and concepts are important to the discussion of Policy Director throughout this document:

Secure Domain — The group of users, systems, and resources that share common services and usually function with a common purpose.

Authentication — The process of identifying any individual who is attempting to log in to a secure domain.

Credentials — Detailed information acquired during authentication that describes the user, group associations (if any), and other security-related identity attributes.

Authorization — The process of determining whether an individual has the right to perform an operation on a protected resource.

Encryption	The translation of electronic data into secret code that protects the data from examination by unauthorized parties.
Integrity	The condition where electronic data does not change between the time it is sent and the time it is received.
Quality of Protection	The level of data security that is determined by a combination of authentication, integrity, and privacy conditions.
Scalability	The ability of a network system to respond to increasing numbers of users who access resources.

Common concerns of network security

Both the worldwide public Internet and company-private intranets connect to heterogeneous computer systems, applications, and networks. This mixture of dissimilar hardware and software usually affects a network in the following ways:

- No centralized control of security for applications.
- No unified resource-location naming convention.
- No common support for high availability of applications.
- No common support for scalable growth.

New business models require organizations to expose their information resources to a degree not previously thought of. These businesses need to know that they can securely control access to those resources.

Managing policy and users across distributed networks has proven difficult for Information Technology (IT) managers. It is difficult because individual application-and-system vendors put into effect authorization in their own proprietary fashion.

Companies realize that developing new authorization services for each enterprise application is an expensive process that leads to a difficult-to-manage infrastructure. A centralized authorization service, which is accessed by developers by using an application program interface (API), could greatly speed time to market and reduce total cost of ownership.

A centralized, network-security management system needs to fulfill requirements that include:

- Co-existence with and leverage of existing firewall and authenticator architectures.
- Integration or co-existence with network and application management frameworks.
- Application-independence.

Introducing Policy Director

Policy Director is a complete authorization, network security, and policy management solution that provides end-to-end protection of resources over geographically dispersed intranets and extranets. An *extranet* is a virtual private network (VPN) that uses access control and security features to restrict the use of one or more intranets attached to the Internet to selected subscribers.

In addition to its state-of-the-art security policy-management feature, Policy Director supports authentication, authorization, data security, and resource management capabilities. Use Policy Director in conjunction with standard Internet-based applications to build highly secure and well-managed intranets.

With Policy Director, businesses can now securely manage access to private internal network-based resources. Also, businesses can leverage the public Internet's broad connectivity and ease of use. Policy Director, in combination with a corporate firewall system, can protect the enterprise intranet from unauthorized access and intrusion.

Policy Director authorization service API standard

Authorization services are a critical part of an application's security architecture. 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 accessing a Web-based retirement fund can view personal account information. Before that can happen, an authorization server must verify the identity, credentials, and privilege attributes of that user.

The standards-based Policy Director Authorization API allows applications to make calls to the centralized Policy Director authorization service. By using these calls, you eliminate the necessity for developers to write authorization code for each new application.

The Policy Director Authorization API allow businesses to standardize all applications on a trusted authorization framework. Using the Policy Director Authorization API, businesses can provide more control over access to resources on their networks.

See the *Policy Director Programmer's Guide and Reference* for complete information and descriptions of the Policy Director Authorization API.

Policy Director core technologies

The Policy Director network security-management solution provides and supports the following core technologies:

- Authentication
- Authorization
- Quality of data protection
- Scalability
- Accountability

Authentication

This core technology includes support for Policy Director username and password *authentication mechanisms*.

Secret Key:

- Kerberos
- Lightweight directory access protocol (LDAP)

Public/private key:

- Login over a secure socket layer-enabled (SSL) browser using applications-specific username and password:
  - Basic authentication (BA) mechanism—only WebSEAL and the secure socket layer interface HTTPS.
  - Policy Director forms-based mechanism—only WebSEAL and HTTPS.
- Login over SSL by using the client-side X.509 certificate—Policy Director supports the PKIX-compliant public key infrastructure (PKI) products (such as IBM SecureWay Trust Authority, Version 3.1) or Entrust-based PKI products (such as IBM Vault Registry, Version 2.2.2).

  IBM SecureWay Trust Authority Version 3.1 includes client software, a simple registration application, a certificate authority and an integrated directory to support the complete certificate life-cycle, including enrollment and initial certification, key-pair update, certificate update, certificate and certificate revocation list (CRL) publication, and certificate revocation. A graphical user interface (GUI) is provided for managing Certification Authority (CA), Registration Authority (RA), and End Entity (EE) requests. An API library is also provided.

Credentials acquisition:

- Credentials acquisition service (CAS)—customized authentication extensions.

Authorization

This core technology provides support for the following types of Policy Director authorization:

- The Policy Director Authorization Service
- The standards-based Policy Director Authorization API
- An external authorization capability

Quality of data protection

Quality of protection is the degree to which Policy Director protects any information that is transmitted between client and server. The combined effect of tunnel mechanisms, encryption standards, and modification-detection algorithms determine quality of protection.

In the order of increasing security, quality-of-protection levels include:

1. Standard transmission control protocol (TCP) communication (no authentication)
2. Authentication only—verifies the user's identity
3. Authentication + data integrity—protects messages (data stream) from being changed during network communication
4. Authentication + data integrity + data privacy—protects messages from being changed or inspected during network communication

You can specify the required levels of protection you want to use for specific hosts and networks.

Supported encryption standards: Policy Director supports the following data encryption standard (DES) and other encryption ciphers over SSL:

- 40-bit RC2
- 128-bit RC2
- 40-bit RC4
- 128-bit RC4
- 40-bit DES
- 56-bit DES
- 168-bit triple DES

Policy Director NetSEAL and Policy Director WebSEAL support 40-bit DES and 56-bit DES encryption over DCE-Remote Procedure Call (DCE-RPC).

Note: International versions could be subject to export restrictions on encryption technology.

Tunnel mechanisms: Policy Director supports the following protocols for transmitting encrypted data:

- Secure socket layer (SSL) tunneling
- Generic security services (GSS) tunneling

WebSEAL supports data integrity and data privacy that are provided by the SSL-encrypted tunnel. WebSEAL and NetSEAL support RPCs. Use of integrity and timestamps with RPC provides protection against *playback attacks*. A playback attack occurs when a user's data is captured as it flows between that user's client and the server. That data is then replayed or presented back to the server as a means of impersonating that first user.

SSL tunneling: The SSL protocol allows the exchange of signals to set up communications between two workstations. This protocol provides security and privacy over the Internet. SSL works by using public key for authentication and secret key to encrypt data that is transferred over the SSL connection.

Enable SSL when using SSL tunneling to Policy Director NetSEAL servers. This configuration is used when NetSEAT client serves as an SSL client to a Policy Director NetSEAL server that is securing specific ports (for example, the port used by Telnet).

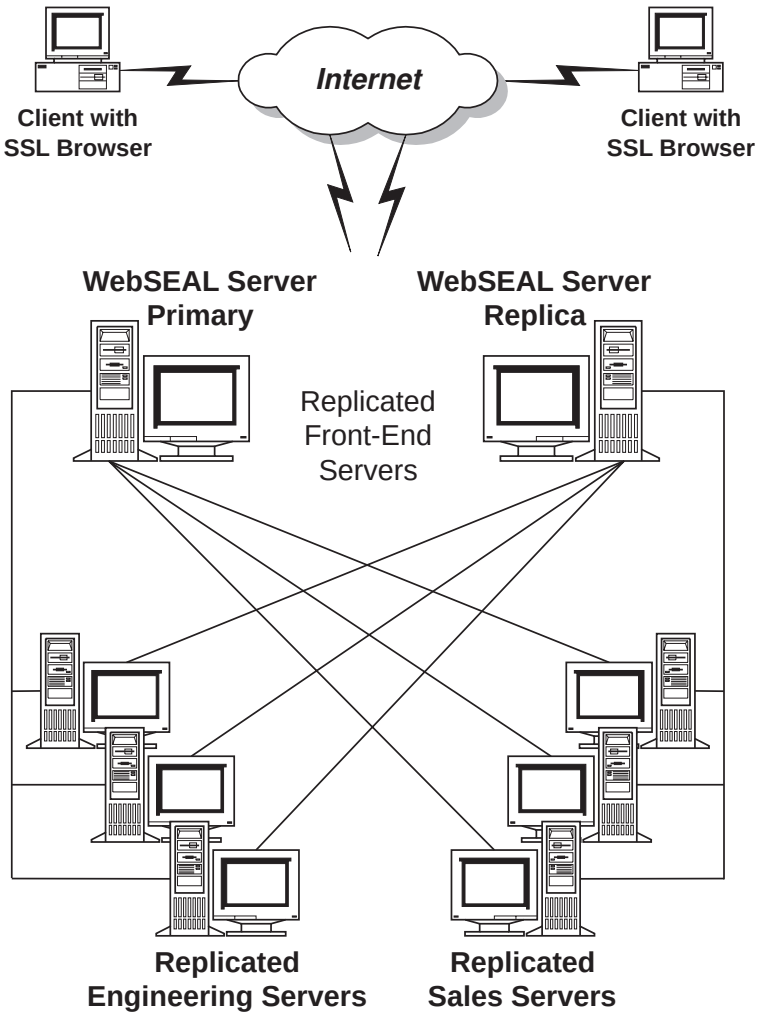
Policy Director WebSEAL supports SSL Versions 2 and 3.

GSS tunneling: The GSS interface (GSS API) is a standard way to allow applications to access security services. GSS tunneling is used over secure RPCs. Enable this option when installing the NetSEAT client as a support module to either Policy Director for Microsoft Windows NT or Policy Director Management Console.

GSS tunneling provides security services to callers in a generic fashion. It is supportable with a range of underlying mechanisms and technologies. It allows source-level portability of applications to different environments. GSS tunneling enables control over the level of protection on traffic traveling in both directions independently of each other. For example, data traveling from the client to the server might be fully protected with bulk data encryption while data traveling from the server to the client might be unprotected.

Scalability

Scalability is the ability to respond to increasing numbers of users who access resources in the secure domain.



Policy Director uses the following techniques to provide scalability:

- Replication of services
  - Authentication services
  - Authorization services
  - Security policies
  - Data encryption services
  - Auditing services
- Front-end replicated WebSEAL servers
  - Mirrored resources for high availability
  - Load balancing of client requests
- Back-end replicated servers
  - Back-end servers, which can be WebSEAL servers or third-party Web servers
  - Mirrored resources (unified namespace) for high availability
  - Additional content and resources
  - Load balancing of incoming requests through Smart Junction technology

- Optimized performance by allowing the off-loading of authentication and authorization services to separate servers
- Scaled deployment of services without increasing management overhead

Accountability

Policy Director provides a number of logging and auditing capabilities. There are log files that capture any error messages and any warning messages that are generated by both Policy Director servers and DCE servers. There are also audit trail files that monitor Policy Director and DCE server activity.

Log files

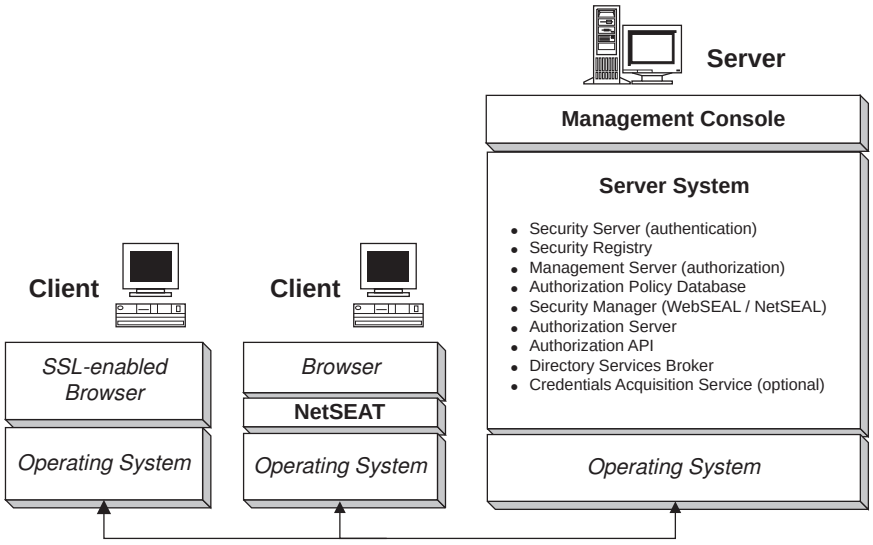
- Policy Director server log files
- DCE server log files
- DCE serviceability messages
- Standard HyperText Transfer Protocol (HTTP) log files

Audit trail files

- Policy Director authorization audit trail files
- WebSEAL audit trail file
- Policy Director management audit trail file
- DCE audit trail file
- LDAP audit trail file

Policy Director components

Policy Director includes software for client systems and server systems. Policy Director provides support on the Sun Solaris, IBM AIX, and Microsoft Windows NT operating system platforms.



Management Console

The Management Console is a Java™-based graphical application that is used to manage security policy for the Policy Director secure domain. From the Management Console, you can perform administrative tasks on the account registry and the primary (also known as *master*) authorization policy database.

Typical Management Console tasks include adding and deleting user accounts and group accounts, and applying access control lists to namespace objects. The Management Console uses secure RPC to perform these management tasks through secure communication channels.

You can delegate management responsibilities to the local level. For example, you can assign a specific security administrator with limited responsibilities. Then, this security administrator is able to manage security policy for only those resources that are located in a designated portion of the protected object namespace.

Security server

The Security server (secd) can be either an LDAP server or a DCE server. The Security server provides authentication services. It also maintains a centralized registry database (LDAP or DCE) that contains account entries for all valid users who participate in the secure domain.

In a DCE environment, registry database users are sometimes referred to as *principals*.

The Security server performs two important roles:

- The Security server defines the groups and organizations to which the user belongs and the roles the user can assume. The centralized registry database stores this information. The Policy Director Authorization Service considers this information when making authorization decisions.
- The Security server provides authentication services for all login attempts.

For DCE, the Security server can replicate the registry database throughout the secure domain to prevent a single point of failure. The Security server is responsible for updating all replica databases whenever a change occurs to the primary registry.

Management server

The Management server (ivmgrd) maintains the primary authorization policy database for the secure domain. It is also responsible for updating all authorization database replicas throughout the secure domain. The Management server also maintains location information about the other Policy Director servers in the secure domain.

Security Manager

The Security Manager (secmgrd) applies access control policy that is based on information from a replica, authorization-policy database. The Security Manager includes:

- A NetSEAL component for coarse-grained Transmission Control Protocol/Internet Protocol (TCP/IP) access control.
- A WebSEAL component for fine-grained HTTP and HTTPS access control.

WebSEAL

WebSEAL is one of the two Security Manager (secmgrd) components.

WebSEAL is a high-performance, multithreaded Web server that accepts HTTP, HTTPS, and NetSEAT client requests. WebSEAL manages access control for such resources as:

- Universal Resource Locations (URLs)
- URL-based regular expression
- Common Gateway Interface (CGI) programs in Perl, C, or C++
- Hypertext Markup Language (HTML) files
- Java servlets
- Java class files

WebSEAL, as a junction server, secures and manages third-party Web servers through Smart Junction technology. *Smart junctions* allow you to attach additional server file systems to the Web space and view the resources as a single unified object namespace.

Use WebSEAL to provide single sign-on capabilities for Web-based resources. The user authenticates to WebSEAL by using standard Kerberos or SSL. WebSEAL then impersonates the user by using HTTP basic authentication and digest authentication. WebSEAL can also pass the user's identity as a CGI variable.

NetSEAL

NetSEAL is one of the two Security Manager (secmgrd) components. NetSEAL is a virtual private network (VPN) solution for securing all incoming TCP/IP communication. NetSEAL performs access control that is based on the destination port and identity of the client. NetSEAL is the security solution for

- Authorizing and securing traditional Internet services, such as Telnet and POP3.
- Authorizing and securing various application packages, such as database systems and network management tools.

NetSEAL is a resource manager that controls a user's ability to connect to a particular port on the server (such as port 23, Telnet). The NetSEAL component also accepts and authorizes TCP/IP traffic that is tunneled from the NetSEAT client.

Using the NetSEAL server allows integration of any network application server with the Policy Director security services. The NetSEAL server provides a secure tunnel endpoint for all network communications. The user's authenticated identity is passed, along with the original protocol request, over this SSL-created or GSS created tunnel. Use NetSEAL SSL tunnels to communicate with the NetSEAT client.

NetSEAT client

NetSEAT is a network support module. This module works seamlessly as a secure proxy for client applications by allowing end-to-end encryption over an SSL or GSS tunnel of all client/server traffic. As a Dynamic Link Library (DLL) implementation of a security client, NetSEAT allows users to take full advantage of Policy Director's features. These features include securing data communications and providing high availability architecture.

NetSEAT ensures full integration with the Policy Director security mechanism and provides resource management for the client. NetSEAT provides protection to TCP/IP applications. NetSEAT transparently encrypts the application data into VPN tunnels (such as SSL or GSS) that can be transported over unsecure links (such as the public Internet).

It can be configured to intercept all outgoing HTTP requests and to forward them to the destination WebSEAL server. It transparently maps logical URLs to physical WebSEAL servers by allowing the relocation or replication of Web resources—without affecting the end-user.

Note: You do not need NetSEAT in order to interact with Policy Director. For example, client users can use SSL-enabled browsers to communicate directly with WebSEAL.

Authorization API

The Policy Director Application Development Kit (ADK) includes an authorization API server (AuthAPI) that lets developers build Policy Director security and authorization directly into corporate applications. The Policy Director authorization API provide direct access to the Policy Director Authorization Service. Use of these authorization API means that developers no longer need to write authorization code for each application.

The Policy Director Authorization API reduces both application-development time and application-development cost. Because the authorization API provide for central management of all network security, the total cost of ownership and the likelihood of security breaches are reduced.

Authorization server

In remote cache authorization mode, applications use the function calls that are provided by the Policy Director Authorization API to communicate to the Policy Director Authorization server (ivacld). The Policy Director Authorization server maintains a replica of the authorization policy database, and functions as the authorization decision-making evaluator.

The Policy Director Authorization API forward an authorization decision request to the Policy Director Authorization server. The Policy Director Authorization server returns a recommendation that is based on security policy. The server can also write an audit record that contains the details of the authorization request.

Directory Services Broker

Policy Director automatically installs and configures the Directory Services Broker (DSB) during the Policy Director installation. Policy Director provides the DSB as part of the Management server (IVMgr) package. No additional steps are necessary to use a DSB.

If NetSEAT clients serve as the support module for Policy Director servers or the Management Console, they will use the DSB. If NetSEAT clients are using only SSL tunnelling, they will not use the DSB.

The DSB acts as a Cell Directory Services (CDS) mid-tier server. The NetSEAT client directs requests for the location of resources and services to the DSB. The DSB, in turn, contacts the secure domain's CDS to resolve the request. Then, the DSB returns the requested information to the system that runs the NetSEAT client.

Credentials Acquisition Service (optional)

The Policy Director Credentials Acquisition Service (CAS) is an optionally configured component.

Credential acquisition is the process of transforming or mapping specific identity information provided by an authentication mechanism into a common, domain-wide representation of the client's identity. This common representation is called the *client's credentials*.

When you need credentials acquisition or mapping, you must configure the Policy Director Credentials Acquisition Service to be used with the Policy Director WebSEAL server. Policy Director users are automatically mapped to credentials by WebSEAL.

Clients accessing Policy Director using client-side X.509 certificates can have *certificate* information mapped to Policy Director identities through the Policy Director Credentials Acquisition Service or by writing your own credentials acquisition service.

If you have users defined in some other external registry, the usernames can be mapped to Policy Director identities through the use of a customized CAS server. You can write and customize your own CAS server to provide a specific solution for the secure domain and to process authentication information, such as: client certificates, user names, or tokens. The Policy Director Credentials Acquisition Service developer or designer determines entirely the specifics of this authentication and mapping service. Policy Director stores mapping rules in a database external to Policy Director. Policy Director provides the Interface Definition Language (IDL) interface between WebSEAL and the Policy Director Credentials Acquisition Service. Policy Director also provides the general server framework that handles Policy Director Credentials Acquisition Service server functions such as startup, server registration, and signal handling. It is the Policy Director Credentials Acquisition Service developer's responsibility to extend the credentials acquisition service framework to carry out the identity mapping functions that are required by the particular application.

Understanding the security model

Policy Director security means controlled access to information. Policy Director technology maps security policy for an organization onto the objects of your protected namespace.

You can base this access on business policies without the restriction of network topology. Users can be permitted or denied access that is based on who they are and what their role is—rather than their physical location.

Policy Director components are client and server-based applications. By using mutual authentication and the assignment of access rights, you can make specific resources available for general availability. At the same time, you can restrict sensitive internal resources to more secure authorized access. Your information is secure, regardless of whether an authorized user accesses the data from within the secure domain or using a remote Internet connection.

Defining a security policy

Policy Director security software lets you create a secure domain where all communication is protected from unauthorized access and undetected corruption.

The administrator of a secure domain must identify:

- Who can participate in the secure domain and request access to objects in the protected object namespace.
- What objects you should protect.
- What rules protect those objects.

Policy Director processes a client request in the following manner:

- Prove who the client is using authentication.
- Acquire rights in the form of authorization credentials.
- Perform an authorization decision that is based on these credentials.

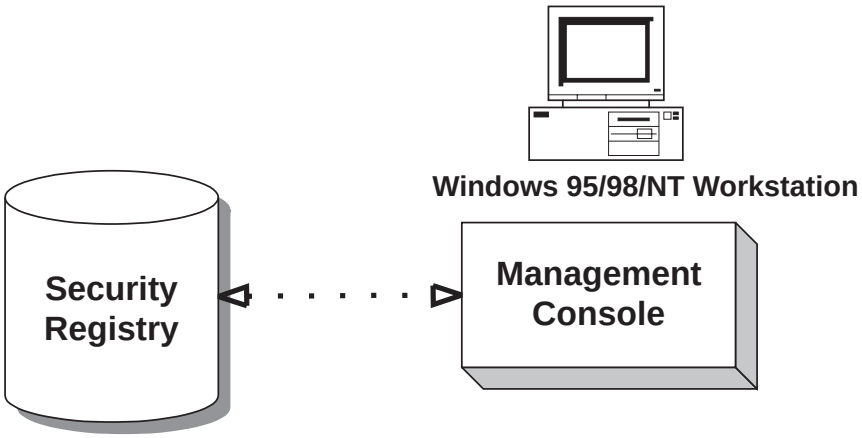
Who can participate in the secure domain?

The administrator maintains an official list of the users (or referred to as *principals* in the DCE environment) and groups who are members of the Policy Director secure domain. Therefore, these users and groups can participate in accessing resources. The registry database (LDAP or DCE) stores this user and group information.

You can authorize a user to participate in the secure domain as soon as you create an account for that user.

Administration Task:

- Create user and group accounts using the Management Console (or by using the **ivadmin** command.



What objects you must protect and by what rules?

Policy Director can protect the following types of resources:

- Web objects, such as HTML files, CGI programs, and dynamic HTML
- Network services that are secured by NetSEAL, such as Telnet, POP3, and custom applications
- Management functions

Policy Director represents actual resources as objects in a protected object namespace. You assign specific access permissions by attaching *policy templates* to these objects. Policy Director uses a type of policy template that is known as an *access control list* (ACL). An ACL defines:

- Who can access the object.
- What operations can be performed on the object.

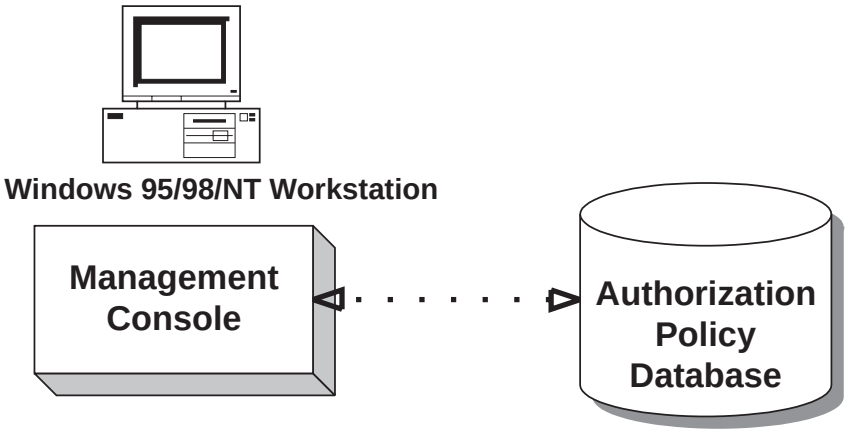
For example, you can grant view privileges on the object to all groups but allow only one group to change the object.

Policy Director employs a mechanism to set global permissions known as the *sparse (or inherited) ACL model*. The sparse ACL model refers to not having an ACL applied directly to every object in the hierarchy. Instead, the model uses ACL inheritance. If an object within the hierarchy does not have an ACL applied to it, the effective ACL is the next ACL that appears higher in the hierarchy. An ACL is required on the root object (/) so that every object is guaranteed to have an ACL to inherit.

When you use the global permissions mechanism, you do not have to set permissions for every file or directory.

Administration Task:

- Define security policy for the namespace by using the Management Console to apply policy templates (ACLs) on objects that require protection.

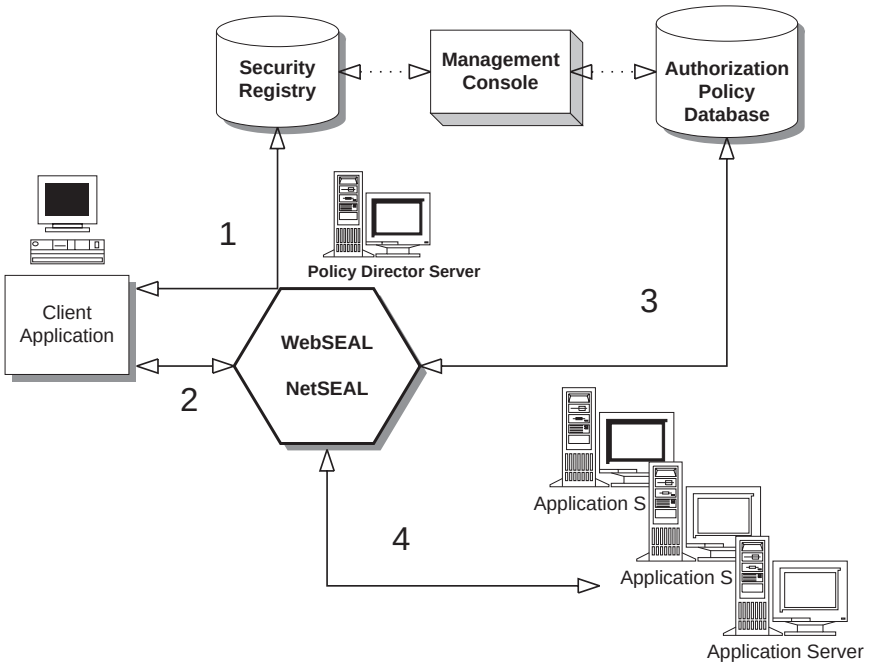


Applying security policy to a client request

When a user requests access to a protected object or application, Policy Director applies the appropriate authentication and authorization checks before approving the request.

1. The client user authenticates to the Security server to establish proof of identity. Policy Director supports authentication through both public and secret key methods.

   Based on this identity, the Security server returns the user's authorization credentials. Credentials define the groups and organizations to which the user belongs and defines the roles the user can assume.



2. A secure communication tunnel is established between the client user and the Policy Director server.
3. An authorization check takes place against a centralized and replicated authorization policy database. Policy Director enforces ACLs that are based on the user's credentials.
4. If the permission settings are appropriate for the user's credentials, Policy Director will pass the request to the application server to complete the transaction.

Chapter 2. Authentication and credentials acquisition

Authentication is the process of identifying an individual who is attempting to log in to a secure domain. The goals of authentication are to prove a client's identity and to obtain credentials that describe that client. Credentials can be used by Policy Director for authorization, auditing, and other services.

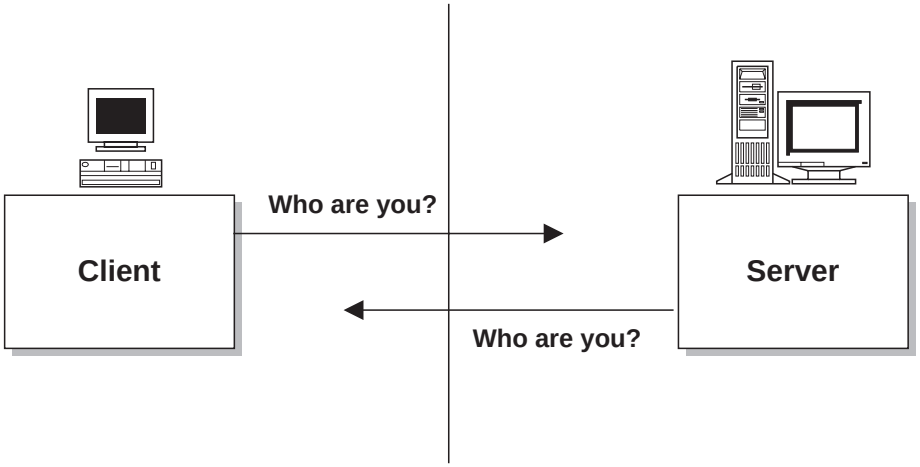
This chapter includes:

Basic concepts of authentication

When servers enforce security in a secure domain, each client must provide proof of its identity. When access to every resource in a secure domain is controlled by a server, the server's demands for authentication and authorization can provide comprehensive network security.

Authentication is the process of identifying an individual who is attempting to log in to a secure domain. In security systems, authentication is distinct from authorization.

Authorization determines whether an authenticated user has the right to perform an operation on a specific resource. Authentication ensures that the individuals are who they claim to be, but says nothing about their ability to perform operations on a resource.



Policy Director supports a flexible approach to authentication that allows security policy to be based on business needs and not physical network topology.

Policy Director authenticates the identity of users when they log in to a secure domain in order to access protected information. Users can assume a number of defined roles—each role have different access permissions.

Goals of authentication

The authentication process achieves two important goals:

1. Determine the client's identity.
2. Acquire credentials for that client.

The *authentication mechanism* and the *credentials acquisition mechanism* are actually two distinct processes. Policy Director supports a number of authentication mechanisms (refer to "Supported authentication mechanisms").

Policy Director also provides default and customizable services for credentials acquisition. A *credentials acquisition service* maps mechanism-specific identity information into a Policy Director credential. Policy Director credentials use the Extended Privilege Attribute Certificate (EPAC) format.

Credentials are used by any Policy Director service that requires information about the client. Credentials can be leveraged by Policy Director to perform a multitude of services, such as: authorization, auditing, and delegation.

Supported authentication mechanisms

Policy Director supports both secret and public/private key-based *authentication mechanisms*:

Secret Key:

- Kerberos Version 5
- Lightweight directory access protocol (LDAP)

Public/Private Key:

- Login over SSL by using client-side X.509 certificate—Policy Director supports the PKIX-compliant public key infrastructure (PKI) products (such as IBM SecureWay Trust Authority, Version 3.1) or Entrust-based PKI products (such as IBM Vault Registry, Version 2.2.2).

  Requires the aid of a credentials acquisition service to obtain credentials.

Types of authentication

The following types of authentication are supported by Policy Director:

- SSL authentication—for Internet and intranet authentication
- Username and password authentication—for user identity-based authentication
- Kerberos authentication—for network authentication

SSL authentication

The secure socket layer (SSL) protocol provides authentication, security, and privacy over the Internet. SSL uses:

- Public/private key pair cryptography for authentication.
- Secret key to encrypt data over the SSL connection.

As an authenticator, the SSL protocol supports server-only authentication and mutual authentication.

Policy Director supports SSL Versions 2 and 3.

Protocol details

The SSL protocol, built on top of TCP/IP, is application independent. The SSL protocol allows application protocols (such as HTTP, FTP, and Telnet) to be layered transparently on top. The HTTP Web communication protocol that operates over an encrypted SSL channel is known as *HTTPS*.

The SSL protocol is able to negotiate encryption keys as well as authenticate the server before data is exchanged by the higher-level communication application. The SSL protocol maintains the security and integrity of the transmission channel by using encryption, authentication, and message authentication codes.

Third-party trust and the certificate authority

SSL authentication depends on the fundamental trust of a third party that vouches for the trustworthiness of one or both authenticating parties. This trusted third party is known as a *certificate authority* (CA).

A CA is responsible for issuing *digital certificates* (electronic identities) that identify individuals, groups, or systems that use the network and prove to anyone else that the owner is trusted by the CA.

By digitally signing these certificates, the CA binds the identity of the certificate owner to the public key contained in the certificate. Anyone trusting the CA should also trust the user.

Network users can obtain the CA's own public key certificate, and use it to verify other users' certificates. With this verification, they have the assurance that the public keys in those certificates are the authentic keys of the named owners and know that the CA (whom they recognize and trust through the root certificate) vouches for this binding.

After two authenticated parties exchange public key certificates, they can proceed to encrypt and digitally sign session data. This encryption and digital signing removes the possibility that others might eavesdrop on the session or tamper with data.

A CA can be a company that sells certificates over the Internet, or it can be a department responsible for issuing certificates for your company's intranet. You must decide which CAs you trust enough to serve as verifiers of other people's identities.

One of the IBM SecureWay FirstSecure (FirstSecure) set of security products is IBM SecureWay Trust Authority (Trust Authority) . This product provides the capability of issuing your own certificates for your company's intranet. The most up-to-date Information about FirstSecure and its components can be found at this Web site:

`http://www.ibm.com/software/security/firstsecure/library`

X.509 digital certificates

Authentication over SSL is provided through digital certificates. The certificate is a file that contains certain identifying information. You purchase, or otherwise receive, a certificate from a trusted certificate authority (CA). The CA's main responsibility is certifying the authenticity of users.

Certificates are nontransferable and nonforgeable files that act as a type of electronic identity badge or passport. Certificates help ensure that users or computers are who they say that they are. The file is signed with the CA's private key to guarantee its authenticity and integrity.

An SSL-enabled browser uses the industry-standard certificate type, known as *X.509*. Version 3 of X.509 contains the following information:

- Version
- Serial number
- Signature algorithm ID
- Issuer name
- Validity period
- Subject (user) name
- Subject public key information
- Issuer unique identifier (the distinguished name of the issuing certificate authority)
- Subject unique identifier (the distinguished name of the individual that is identified by the certificate)
- Extensions (for Version 3 only)
- Signatures for all these fields, listed above

The X.509 Version 3 standard allows more detailed identifying information, such as: what business the certificate holder is in, and how long they have been in business. The certificate is signed by the issuer to authenticate the binding between the subject's (user) name and the user's public key.

Certificates do not prove conclusively that people or computers are who they claim to be, but they do signify that some CA has a degree of trust in the person or computer. When you trust the CA that issued the certificate, you have some degree of confidence when you exchange information with a certificate holder.

Basic concepts of SSL authentication mechanism

An SSL *handshake* protocol is the process of exchanging signals to set up communications. The SSL handshake protocol can consist of two phases:

- "Server authentication using server-side certificates"
- "Client authentication by using client-side certificates" on page 20 (optional)

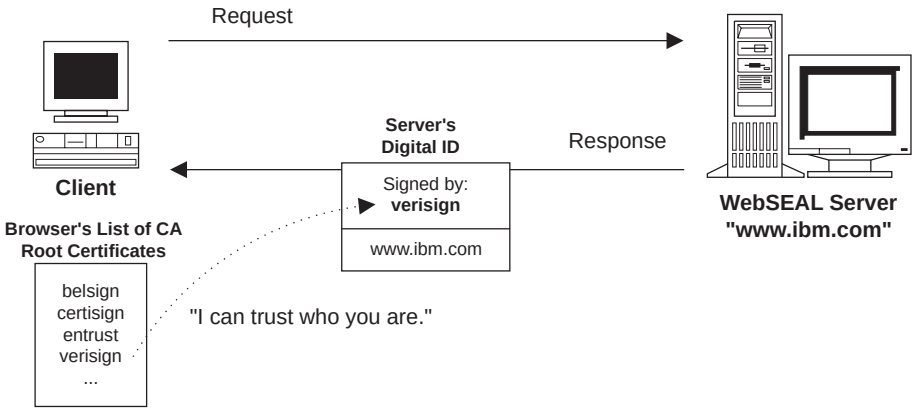
Both clients and servers can have certificates. The server must always have a certificate for authentication over SSL. Clients can access the secure domain over SSL with or without client-side certificates.

When a server sends its certificate to a client, the process is called *server authentication*. When a client sends its certificate to a server, the process is called *client authentication*. The combination of server and client authentication is known as *mutual authentication*.

Server authentication using server-side certificates

Server authentication is required for an SSL connection. Server authentication over SSL is accomplished in the following manner:

1. A client requests a connection with an SSL-enabled server.
2. In response, the server signs (but does not encrypt) its certificate. Then, the server sends the certificate that contains the server's public key to the client.
3. The client uses the server's public key that is included in the certificate file to verify that the owner of the certificate is the same one who signed it.
4. The client checks whether the certificate's issuer is one that it accepts by checking the browser's CA root certificate database for the listed CA. If it is one that it accepts, the client will proceed to the next step; otherwise, the browser informs its user that this certificate was issued by an unknown CA. Then, it is the user's responsibility to accept or reject the certificate.
5. The client then generates a master key, encrypts it with the server's public key, and transmits the encrypted master key to the server.
6. The server recovers the master key and authenticates itself to the client by returning a message that is encrypted with the master key. Subsequent data is encrypted with keys that are derived from this master key.



Client authentication by using client-side certificates

The server unlocks a client digital certificate with the client's public key. Client public key certificates follow the X.509 syntax.

Authentication using client-side certificates over SSL is accomplished in the following manner:

1. After server authentication is completed, the server sends a challenge to the client.
2. The client returns its digital signature on the challenge as well as its public key certificate. The digital signature is computed using the client's private key.
3. The server uses the client's public key, which is included in the certificate file, to verify that the owner of the certificate is the same one who signed it.
4. The server attempts to match the certificate to a trusted CA. If the client's CA is not listed as trusted, some servers will end the transaction, log an error, and return a message to the client. Other servers might choose to proceed without this type of action.
5. When the client's CA is trusted, the server fulfills the transaction.

Client-side certificates are not essential for authentication over an SSL connection. You can still exchange encrypted information. Client certificates do give extra assurance to both client and server that they are sending the encrypted information to the correct parties. True mutual authentication is possible with client certificates.

In either case, when your CAS server has been set up to require client certificates for access control, clients are rejected if they do not have a valid certificate.

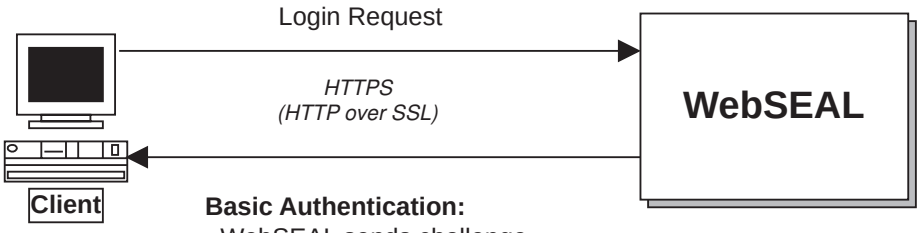
For more information about client-side X.509 certificates used by public and private keys, see "Using X.509 certificate authentication" on page 170.

Username and password authentication

The authentication process requires the client to submit some form of identity information during login. Policy Director WebSEAL supports username and password authentication.

There are two username and password authentication methods that provide this identity information:

- Basic authentication
- Forms-based login



Basic Authentication:
- WebSEAL sends challenge
- Browser produces login prompt

Forms-Based Login:
- WebSEAL sends login form

See “Using username and password authentication” on page 166 for complete information about the authentication mechanisms that require client identity information in the form of a username and password.

Kerberos authentication

Kerberos Version 5 is a network authentication protocol that enables two parties to mutually authenticate each other for the purpose of securely exchanging information across an otherwise open network.

Policy Director can use Kerberos authentication in the following exchanges:

- Management Console to Management server
- Management Console to Security server
- Authorization server to Management server
- WebSEAL to Management server
- WebSEAL to DCE registry (for client authentication)

Policy Director servers communicate with other servers in the Policy Director secure domain by using Kerberos and the NetSEAT client network module. NetSEAT communicates with the Policy Director security service and sets up the secure SSL tunnel during the exchange of information.

Kerberos authentication depends on the fundamental trust of a third party that vouches for the trustworthiness of both authenticating parties. This trusted third-party security management service is known as the *security server*.

The Policy Director Security server (secd) is a physically secured server that stores security-related information (for example, user names, groups, and passwords) in a database that is known as the *registry*.

Kerberos uses a shared, session-specific, secret key mechanism (LDAP Secret Key) to support mutual authentication between servers. Kerberos relies on the trusted Security server for key distribution. Exchange of information occurs using Remote Procedure Call (RPC).

The Kerberos authentication protocol is a complex exchange of a series of messages. This series of messages exchange contains secret keys and other information that is required for servers to identify each other. The goal of Kerberos is to keep all participants from knowing each other’s keys. In fact, the life span of many keys is limited to the duration of a simple exchange.

Credentials acquisition

One of the primary goals of the authentication process is to acquire credential information that describes the client user. Policy Director distinguishes the authentication of the user from the *acquisition of credentials*.

A user’s identity is always constant. However, credentials that define the groups or roles in which a user participates are variable. Context-specific credentials can change over time. For example, when a person is promoted, credentials must reflect the new responsibility level. A person’s credentials on the job are also different from the credentials at the user’s bank.

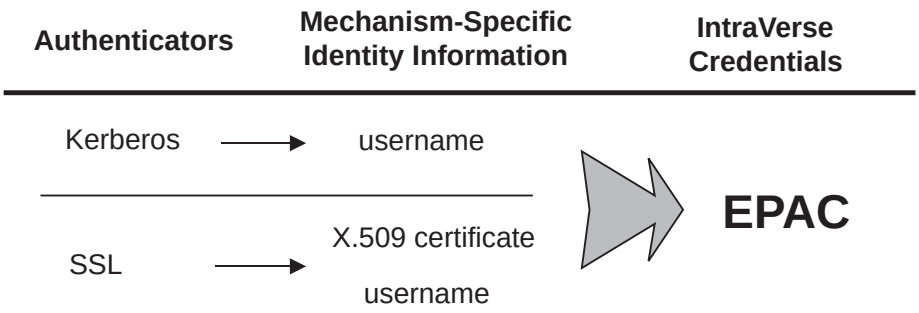
The authentication process results in mechanism-specific user identity information. This information must then be transformed (*mapped*) to a common domain-wide representation and format. Policy Director uses the EPAC format.

Mechanism-specific identity information

For a credentials acquisition service, different authentication mechanisms provide different user identity information:

- Kerberos is based on a username (and password).
- X.509 client-side digital certificates provide X.509 field information.
- Basic authentication and forms-based login methods (SSL) are based on a username (and password).

The following figure illustrates the kind of identify information available from the specified authentication mechanism and the use of the credentials acquisition service over SSL to transform the information into the EPAC format.



Mechanism-specific identity information (for example, passwords, key pairs, and certificates) represent physical identity properties of the user. This information is used to establish the secure session with the server.

The resulting credential, which represents a user's role in the secure domain, describes the user in a specific context and is only valid for the lifetime of that session.

EPAC certificate

Credentials are used by any Policy Director service that requires information about the client. For example, the Policy Director Authorization Service uses credentials to determine whether a user is authorized to perform specific operations on a protected resource in the secure domain.

One way to work with ACLs is for Policy Director to use EPACs that contain universally unique identifiers (UUIDs). Policy Director uses credentials for other services, such as:

- Auditing service
- Delegation capabilities in WebSEAL and NetSEAL junctions

The following EPAC fields are appropriate to Policy Director:

Attribute	Description
Secure Domain ID	User's home secure domain identifier
Principal UUID	UUID of the user (or "principal" if DCE)
Group UUIDs	UUID or UUIDs of groups to which the user belongs

Mechanism-specific authentication information must be transformed into EPAC fields:

- Policy Director clients are automatically mapped to credentials by WebSEAL.
- Non-Policy Director SSL clients from an external registry can have usernames mapped to Policy Director identities through an external credentials acquisition service.
- Clients accessing using client-side X.509 certificates can have certificate information mapped to Policy Director identities through an external credentials acquisition service.

Chains of trust

During the SSL protocol exchange between the browser client and the WebSEAL server, the server passes the browser a list of certificates of the CAs that the server trust. This causes the browser to display to the user a list of browser client certificates that are either:

- Signed by one of these CAs.
- Trusted by virtue of a chain-of-trust relationship with one of the server's trusted CAs—the client certificate is signed by a CA whose signing CA is one that the server trusts.

This process is known as *certificate chaining*.

The browser user selects one of these client certificates for transmission to the server. If the client certificate is directly signed by one of the CAs that the server trusts, then the browser transmits only the client certificate to the server (assuming that the server already has the certificate of the signing CA).

If the client certificate is not directly signed by one of the CAs that the server trusts, the browser constructs and transmits a CA certificate chain that illustrates the chain of trust between the client certificate and one of the server's trusted CAs.

Again, the browser does not actually transmit the certificate of the CA that the server trusts. It assumes that the server already has that certificate.

The Policy Director secmgrd.conf configuration file contains a list of root CA certificates that Policy Director trusts. So Policy Director trusts client certificates that are issued by these CAs.

The server takes each CA certificate that is transmitted by the browser and validates that:

- The certificate is a CA certificate.
- The signature of the signing CA.
- The certificate has not expired.

A *chain of trust* has been established when one CA trusts a second CA, who trusts a third CA, and so forth. If Policy Director trusts any CA in this chain of trust, it should be able to trust the client certificate.

Credentials acquisition service overview

Credential acquisition is the process of transforming or mapping specific identity information provided by an authentication mechanism into a common, domain-wide representation of the client's identity. This common representation is called the *client credentials*.

Credentials derived from the authentication process are used by any Policy Director service that requires information about the client. Examples of such services include authorization and auditing. The major duties of Policy Director depend on the availability of credentials for every client.

Policy Director uses the EPAC format to represent the credential information that is derived from the authentication process.

Policy Director automatically generates credentials for SSL clients who:

- Are members of the secure domain.
- Authenticate with a valid username and password.

In this case, the username and password you provide must match an existing account entry in the default registry (LDAP).

There are other possible scenarios where the client access does not fit the above model:

- The client does not belong to the default Policy Director registry.
- The client access is done by using a client-side certificate.

In all these scenarios, Policy Director must rely on a customized authentication and mapping service that is capable of:

- Performing authentication on these clients.
- Referring to an external (third-party) account registry.
- Mapping external identity information to a Policy Director identity.

This customized authentication and mapping service is known as an *external credentials acquisition service* (CAS).

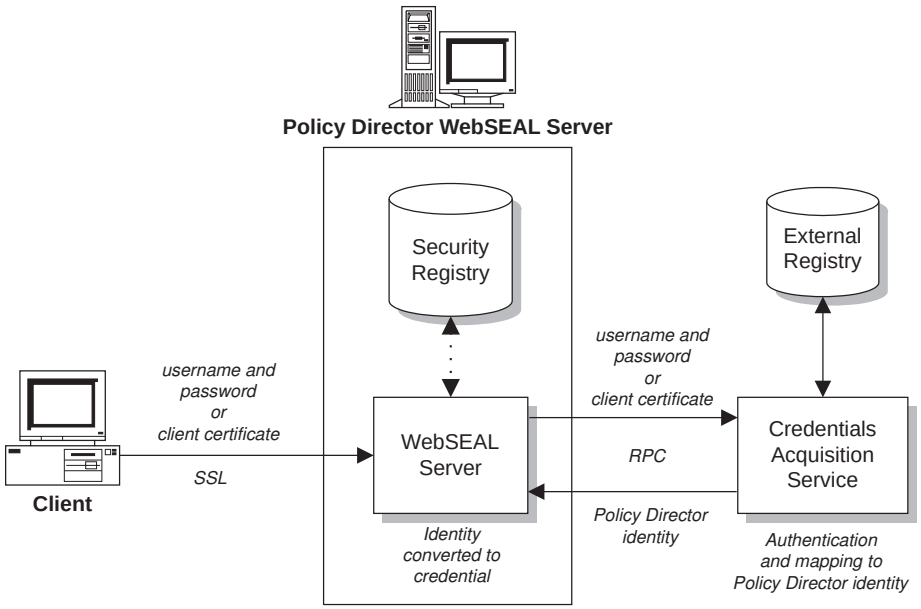
Introduction to the credentials acquisition service

A credentials acquisition service's (CAS) architecture allows you to substitute the default WebSEAL SSL authentication process (based on a username and password) with a custom external authentication process that can reference a user registry other than the Policy Director security registry. The customized credentials acquisition service also performs the appropriate mapping of any special identity information (certificates, tokens) to a Policy Director identity.

The credentials acquisition service must be specially written and customized by the administrator to provide a specific solution for the secure domain.

The credentials acquisition service must use RPCs to secure all communication between WebSEAL and the CAS server.

A credentials acquisition service allows a user who does not have an account in the default Policy Director registry to participate in the secure domain. The credentials acquisition service can authenticate that user (by using an external registry, if necessary). The credentials acquisition service then passes a Policy Director identity back to WebSEAL for conversion to credentials. Policy Director can use these credentials to allow the user to participate in the secure domain.



A credentials acquisition service can accommodate legacy user databases that would be difficult or impossible to migrate to the registry that is normally used by Policy Director. An example legacy system might involve a customer ID and PIN mechanism (token). Such a legacy authentication mechanism verifies the user information through its own registry database.

The Policy Director Authorization Application Developer's Kit (IVAuthADK) provides a demonstration CAS server. This server defines the interface (an IDL) between WebSEAL and the credentials acquisition service. The ADK also provides source if you are writing your own credentials acquisition service.

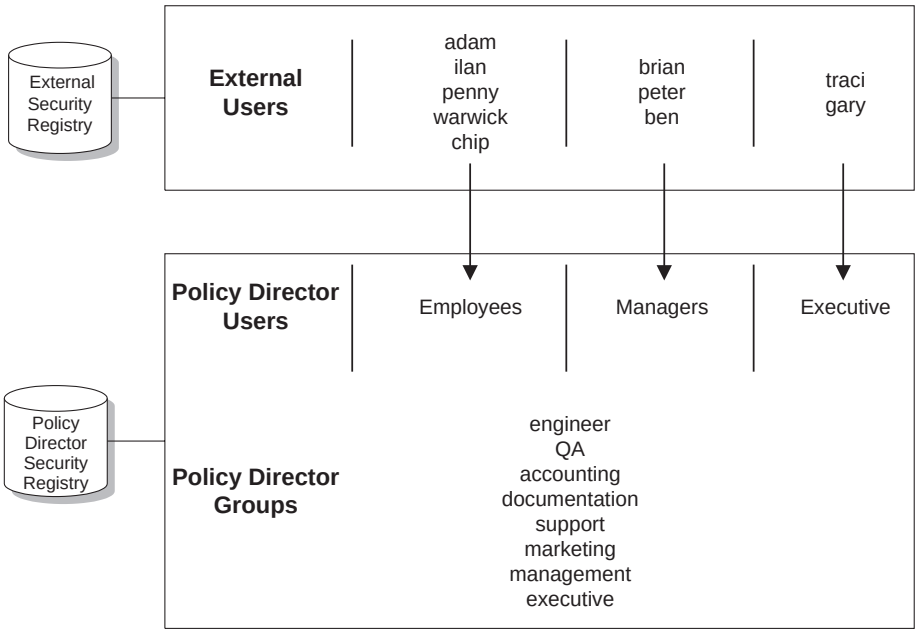
Many-to-one mapping solution

A credentials acquisition services is appropriate for a *many-to-one* solution; that is, the module allows you to map many legacy accounts to one Policy Director user.

In the many-to-one mapping scenario, a Policy Director user takes on the role of a group whose members are the collection of users from the legacy database. The many-to-one mapping solution results in identical access rights, visibility, and accountability for all users who are mapped to the same user. All users who are mapped to a particular user have exactly the same permissions. This fact must be taken into consideration when determining your security policy.

In the following figure, users from an external registry might be mapped to a single Policy Director user. For example, the Policy Director user (`Employees`) acts as a role for a collection of users from the external registry. Although the users are mapped to the same Policy Director account, they can also gain individual

distinction by being assigned membership to one or more Policy Director groups. Policy Director authorization decisions can be based on both user identity and group membership.



Note: The level of accountability in a many-to-one mapping is not fine-grained. Auditing services track the Policy Director user only—not the individual users mapped to this user.

Function modes

A credentials acquisition service can be written to process authentication information, such as client certificates, user names, and tokens. WebSEAL must be configured to accept non-registry SSL clients and to route the authentication information to the proper credentials acquisition service for authentication and mapping to a Policy Director identity.

A credentials acquisition service performs its authentication and identity mapping based on the specific identity information provided by the client. Therefore, the credentials acquisition service can be written to perform in one of the following possible modes:

- "X.509 certificate mapping mode"
- "Username mapping mode" on page 28

See "Custom credentials acquisition service" on page 32 for information on using a custom-written credentials acquisition services.

X.509 certificate mapping mode

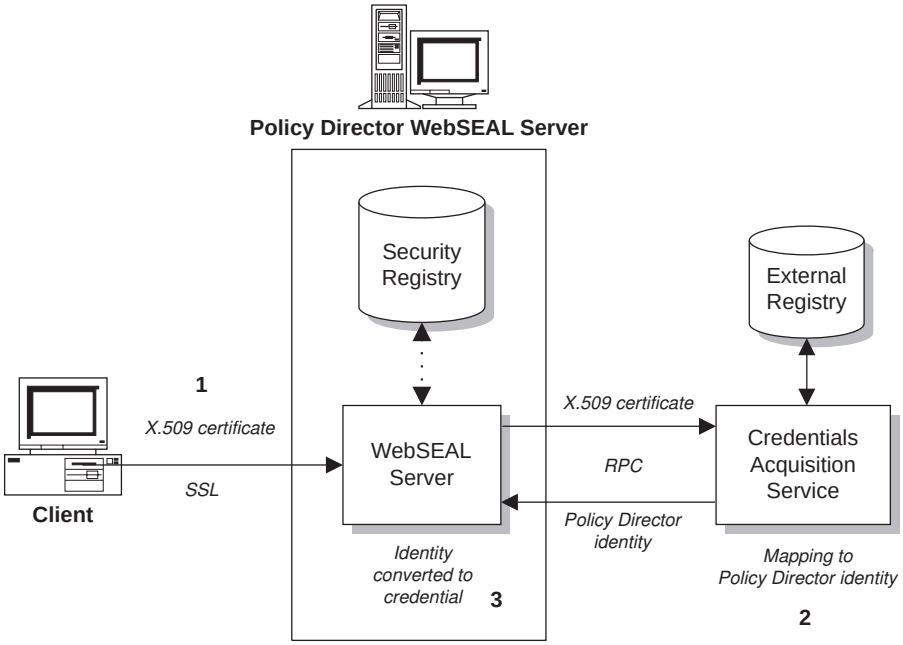
Through any credentials acquisition service, Policy Director can support authentication of clients that use digital X.509 certificates over SSL. A credentials acquisition service X.509 mode maps specific information that is included in a client-side X.509 digital certificate to a Policy Director identity. This Policy Director identity is returned to WebSEAL, which converts it to appropriate credentials.

The X.509 mode is appropriate under the following conditions:

1. Clients communicate over SSL.
2. Clients use X.509 digital certificates for authentication.
3. Clients need to access protected resources in the Policy Director secure domain.

A CAS server can map the certificate information to a Policy Director identity on a one-to-one basis or a many-to-one basis. The goal of the mapping service is to provide the Policy Director Authorization Service with useful credentials for making authorization decisions.

The following figure illustrates the sequence of events when WebSEAL is configured to use a credentials acquisition service for X.509 certificate mapping:

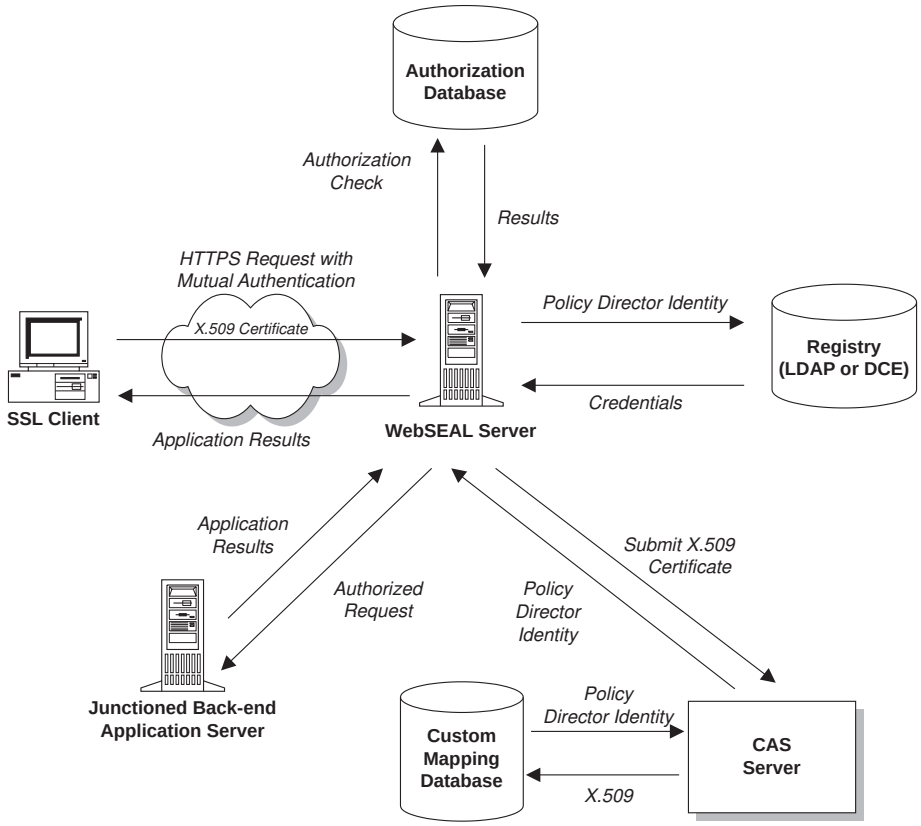


1. The client accesses WebSEAL over SSL and presents an X.509 certificate.

   Note that authentication is accomplished at this point through public and private key certificate exchange. The only duty remaining for the credentials acquisition service is to map user credentials.
2. The CAS server takes identity information (specific to the application) from the validated certificate and maps that information to a known Policy Director identity. The CAS server can make use of an external (third-party) registry.
3. The Policy Director identity is returned to WebSEAL which then uses its default registry to convert the identity to appropriate credentials.

The following figure illustrates the complete sequence of events that occurs when a client, accessing WebSEAL using an X.509 certificate, requests a resource from the secure domain.

1. Certificate information is mapped to a Policy Director identity by the credentials acquisition service, which returns this identity to WebSEAL.
2. WebSEAL creates a credential from this identity and uses the credential to make an authorization decision that involves a protected resource on a junctioned application server.



Username mapping mode

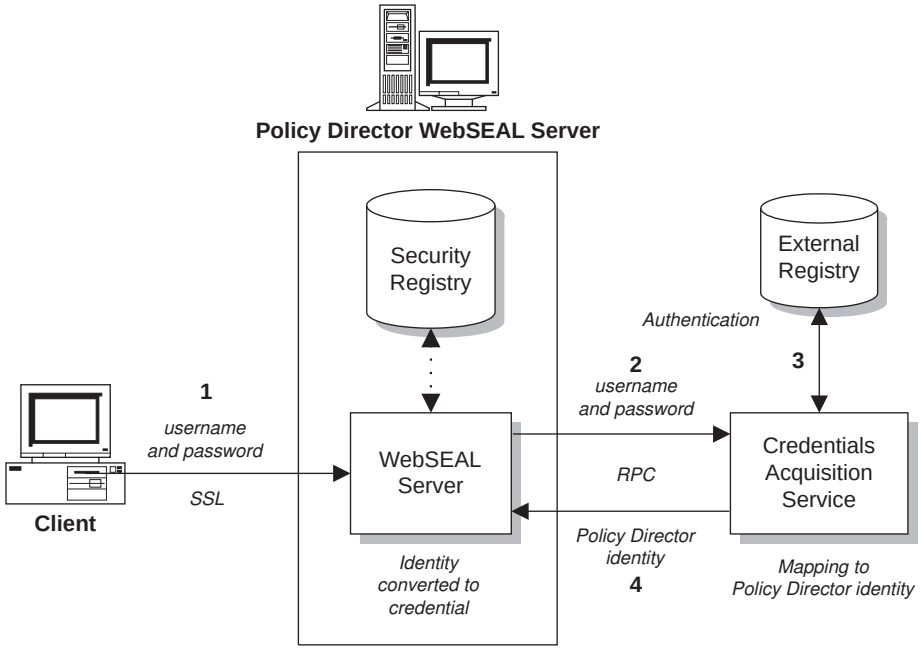
Username mapping mode is another type of authentication and identity mapping. Using the username mapping mode, you can substitute the default authentication process with a custom external process that references a user registry—other than the default Policy Director registry (LDAP).

Standard Policy Director authentication using SSL requires the user to log in with a username and password. Authentication and credentials acquisition for this identity are determined from the Policy Director registry.

The primary value of a credentials acquisition service of this type is to accommodate legacy user databases that would be difficult or impossible to migrate to the Policy Director registry.

The following figure illustrates the sequence of events when WebSEAL is configured to use a credentials acquisition service for username mapping:

1. The client accesses WebSEAL over SSL with a username and password.
2. WebSEAL is configured to pass user names and passwords to the credentials acquisition service for both authentication and credentials acquisition.



3. The credentials acquisition service uses an external (third-party) registry to authenticate the user and then maps the user to a Policy Director identity.
4. The Policy Director identity is returned to WebSEAL which then uses its default registry to convert the identity to a credential.

This mode is best used as a many-to-one solution; that is, the module allows you to map many legacy accounts to one Policy Director user. See “Many-to-one mapping solution” on page 25.

Authentication service choices

You can choose one of the following types of authentication service:

- The default Policy Director Credentials Acquisition Service (see “CAS provided with Policy Director”).
- A custom-written credentials acquisition service (see “Custom credentials acquisition service” on page 32).

CAS provided with Policy Director

Policy Director provides its own client authentication service component—the Policy Director Credentials Acquisition Service (Policy Director CAS). The Policy Director CAS is supported when using LDAP as the user registry.

You must configure WebSEAL to use the Policy Director CAS for authentication by checking and updating the iv.conf and the secmgrd.conf configuration files (see “Policy Director Credentials Acquisition Service” on page 172).

The Policy Director CAS maps *client digital certificates* from the SSL-enabled browser to a Policy Director user identity. When the user tries to access a protected Web page, the SSL-enabled browser contacts the WebSEAL server. If WebSEAL is configured for client certificate-based authentication, WebSEAL requests an X.509 certificate from the browser. When WebSEAL receives the certificate from the

browser, it passes the certificate to the CAS server. The Policy Director CAS attempts to map the received certificate to a user identity understood by Policy Director.

The Policy Director Credentials Acquisition Service has been written to provide support login over SSL using one or both of these client-side X.509 Version 3 certificates:

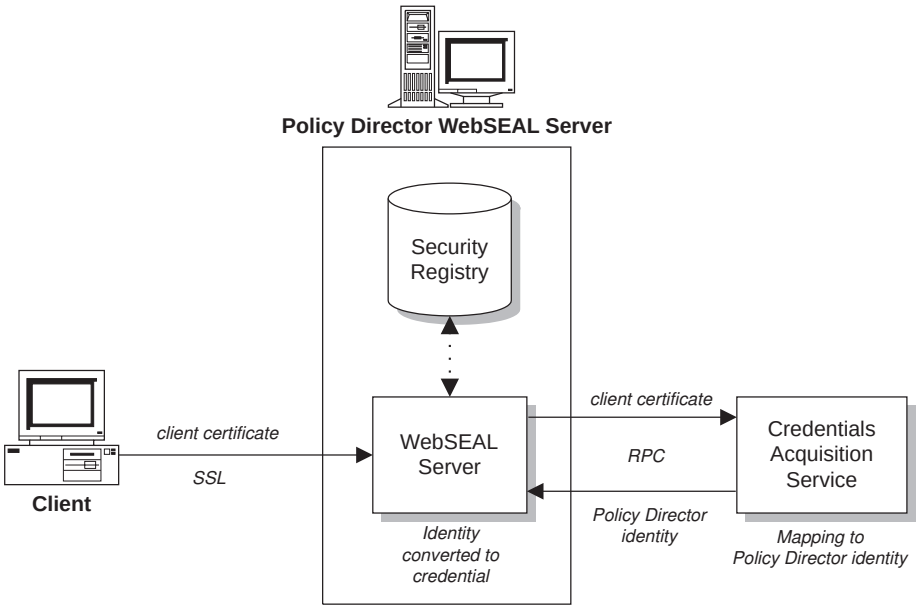
- PKIX-compliant product (for example, IBM SecureWay Trust Authority, Version 3.1)
- Entrust-compliant product (for example, IBM Vault Registry, Version 2.2.2)

All certificates are Distinguished Encoding Rules (DER)-encoded when transmitted.

An RPC interface is used between the Policy Director Credentials Acquisition Service and WebSEAL. The Policy Director Credentials Acquisition Service uses RPCs to secure all communication between WebSEAL and the CAS server. Because Policy Director Credentials Acquisition Service is a DCE-RPC application, a DCE client is needed to perform the relevant client identity mapping.

The default Policy Director Credentials Acquisition Service:

- Allows you to use client-side certificates or specify that they are optional.
- Does not perform any associated certificate revocation list (CRL) checking.
- Supports certificate chaining.
- Supports a one-to-one mapping solution



One-to-one mapping solution

The Policy Director Credentials Acquisition Service uses the one-to-one mapping mode. A *one-to-one* mapping of individual legacy accounts to single users might require a high degree of account maintenance. However, within the Policy Director CAS cdas.conf configuration file, the Policy Director administrator can create a table that is used to associate a certificate distinguished name (DN) with the DN of a Policy Director (LDAP) user.

When the Policy Director CAS is called by WebSEAL with a certificate, it first extracts the DN from the certificate and searches the table for a match. If a match is found, the Policy Director Credentials Acquisition Service returns the associated Policy Director user correctly formatted DN to WebSEAL. WebSEAL then uses this DN to identify the Policy Director (LDAP) user. If a match is not found, the CAS returns the DN from the certificate to WebSEAL. In this case the DN in the certificate is used to identify the Policy Director (LDAP) user. The WebSEAL server uses the returned DN to retrieve the user's credentials.

Required administration tasks

The administration tasks required for setting up the Policy Director Credentials Acquisition Service include:

1. Configure WebSEAL to use the CAS for authentication by checking and updating, if necessary, the iv.conf and the secmgrd.conf configuration files (see "Policy Director Credentials Acquisition Service" on page 172).
2. As required, update the DN mapping table section of the cdas.conf configuration file (see "Distinguished name mapping" on page 174).

Credentials Acquisition Service functionality

The Policy Director Credentials Acquisition Service interface can be used to provide the following functions:

- The Policy Director CAS is called by WebSEAL with a certificate.
- Policy Director CAS extracts the DN from the certificate and searches the DN mapping table for a match.
- If a match is found:
  - The Policy Director Credentials Acquisition Service returns the associated Policy Director user DN to WebSEAL.
  - WebSEAL then uses this DN to identify the Policy Director user.
- If a match is not found:
  - The CAS returns the DN from the certificate to WebSEAL.
  - The DN in the certificate is used to identify the Policy Director user.
  - The WebSEAL server uses the returned DN to retrieve the user's credentials.

Custom credentials acquisition service

Because of the diversity of each application server and its associated authentication framework, no one credentials acquisition service can be written to suit all requirements. For this reason, a demonstration CAS server source is included with Policy Director in the IVAuthADK package. This demonstration CAS server can be adopted as the basic framework for a production CAS server—with the addition of the application-specific username mapping and mapping management functions.

WebSEAL must be configured to accept non-registry SSL clients and to route the authentication information to the proper credentials acquisition service for authentication and mapping to a Policy Director identity.

The custom-written CAS must use RPCs to secure all communication between WebSEAL and the CAS server.

The requirements for mapping X.509 certificate information to a Policy Director identity vary widely from customer to customer. Although Policy Director does not include any generic rules for defining a mapping service, there are two provisions that assist the administrator who wants to set up a customized mapping service:

1. Policy Director defines an IDL interface that allows developers to write their own service for mapping X.509 certificate information to Policy Director identities. Details of this IDL interface are provided in the *Policy Director Programmer's Guide and Reference*.
2. Policy Director includes an example implementation of a mapping service that provides a CAS server framework that simply returns a failure for each request. This framework can be further developed into a production CAS server.

   The source code for the example service is included in the Policy Director IVAuthADK installation package.

Required administration tasks

The administration tasks that are required for setting up a custom CAS include:

1. Writing a custom CAS that uses the IDL interface provided by Policy Director.
2. Configuring WebSEAL to use the external credentials acquisition service for authentication.

Custom CAS functionality

The custom CAS interface can be used to provide the following functions:

- Performing username and password validation against a user registry other than the default Policy Director registry.
- Mapping many users to the same Policy Director identity.
- Managing external user passwords.
- Adding custom audit information to the credential. Policy Director writes the entire credential to its audit log.

Chapter 3. Understanding Authorization

Authorization is a process that determines whether an identified entity has the right or authority to either:

- Start a specific service.
- Perform an operation on a specific resource in a secure domain.

The Policy Director Authorization Service helps to enforce a network's security policy by controlling the authorization decision-making process.

This chapter includes:

Conceptual model of authorization

When servers enforce security in a secure domain, each client must provide proof of its identity. In turn, security policy determines whether that client has permission to perform an operation on a requested resource. The server controls access to every resource in a secure domain. For this reason, the server's demands for authentication and authorization can provide comprehensive network security.

Authentication is the process of identifying an individual who is attempting to log in to a secure domain.

In security systems, authentication is distinct from authorization:

- *Authentication* is the process of identifying an individual who is attempting to log in to a secure domain. Authentication ensures that the individuals are who they claim to be, but says nothing about the rights to perform operations on a protected resource.
- *Authorization* determines whether an authenticated client has the right to perform an operation on a specific resource in a secure domain. In the authorization model, Policy Director accomplishes the authorization policy independent of the mechanism that is used for user authentication. Users can authenticate their identity either by using public/private key or secret key, or by using customer-defined mechanisms.

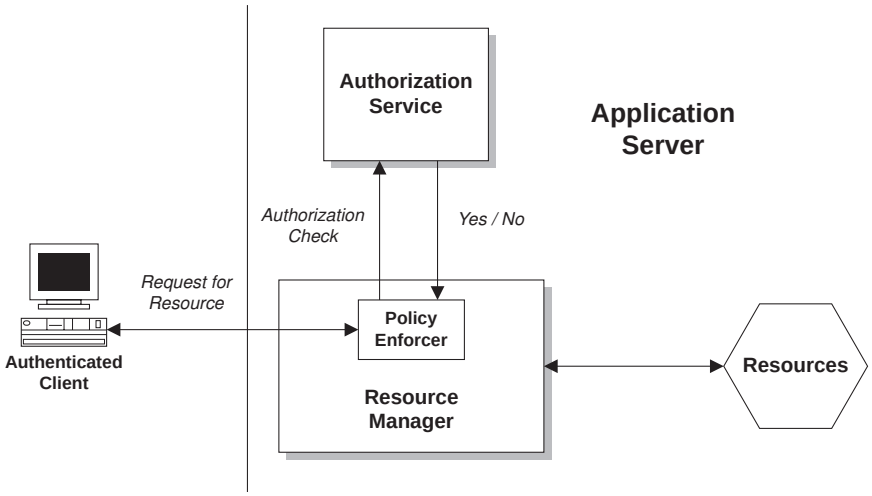
Part of the authentication process involves the acquisition of a credential that describes the identity of the client. An authorization service bases authorization decisions on user credentials.

The resources in a secure domain receive a level of protection as dictated by the security policy for the domain. The security policy defines the allowed secure-domain participants and the degree of protection by surrounding each resource that requires protection.

The basic components of the authorization process include:

Resource Manager
The resource manager is responsible for carrying out the requested operation when Policy Director grants authorization. A component of the resource manager is a Policy Enforcer that directs the request to the authorization service for processing.

Authorization Service
The authorization service performs the decision-making action on the request.



Traditional applications bundle the policy enforcer and the resource manager into one process. Examples of this structure include Policy Director WebSEAL and third-party applications. The independent functionality of these authorization components allows much flexibility in the design of the security enforcement strategy.

For example, such independence lets the security administrator control:

- The location of the processes.
- The person who writes the code for the processes.
- The way the processes perform their tasks.

Benefits of a standard authorization service

Authorization in most systems, both legacy 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.

A standard authorization service provides the following benefits:

- Reduces cost of developing and managing access to applications.
- Reduces total cost of ownership and management of separate authorization systems.
- Provides leverage of existing security infrastructure.
- Allows new businesses to open more securely.
- Enables more new and different kinds of applications.
- Allows shorter development cycles.
- Shares information securely.

Benefits of the Policy Director Authorization Service

Policy Director integrates into existing legacy and emerging infrastructures. Policy Director provides secure, centralized policy management capability. The Policy Director Authorization Service—together with WebSEAL and NetSEAL resource managers—provides a standard authorization mechanism for business network systems.

Existing applications can take advantage of the authorization service without modification of the application itself. Policy Director bases its authorization policy on user roles or group roles. You can apply authorization policies to:

- Network servers
- Individual transactions or database requests
- Specific Web-based information
- Management activities
- User-defined objects

The Policy Director Authorization API allow existing applications to make calls to the Policy Director Authorization Service. In turn, the authorization service makes decisions based on the corporate security policy. See “Policy Director Authorization API” on page 44.

The Policy Director Authorization Service is also extensible. You can configure the Policy Director Authorization Service to call on other authorization services for additional processing by using the external Policy Director Authorization API.

The Policy Director Authorization Service provides the following benefits:

- Is application independent.
- Uses a standard authorization coding style that is language independent (the Policy Director Authorization API).
- Is centrally managed and therefore easy to administer. The addition of a new employee, for example, requires changing the privilege database in one central location, rather than across multiple systems.
- Addresses the application of security services in a heterogeneous cross-platform environment.
- Integrates existing non-Policy Director authorization systems through an external authorization service capability.

- Has a scalable and flexible architecture that is easily integrated with existing infrastructure.
- Enables multitiered authorization—the service passes a credentials packet through the multiple layers of an application process or transaction.
- Uses a common and effective auditing model.
- Is independent of any authentication mechanism.

Policy Director Authorization Service

The Policy Director Authorization Service is responsible for the authorization decision-making process that helps to enforce a network security policy. Authorization decisions made by the authorization service result in the approval or denial of client requests to perform operations on protected resources in the secure domain.

Policy Director Authorization Service components

Three basic components make up the Policy Director Authorization Service:

- The primary (master) authorization-policy database
- The Management server
- The authorization decision-making evaluator

Master authorization-policy database

The primary authorization-policy database contains the security policy information for all resources in the secure domain. The database also contains all necessary credential information that is associated with the participants of the secure domain.

Use the Policy Director Management Console to enter and change the contents of this database.

Management server

The Management server (ivmgrd) performs these tasks:

- Maintains the primary authorization policy database.
- Replicates this policy information throughout the secure domain.
- Updates the database replicas whenever a change is made to the primary authorization policy database.

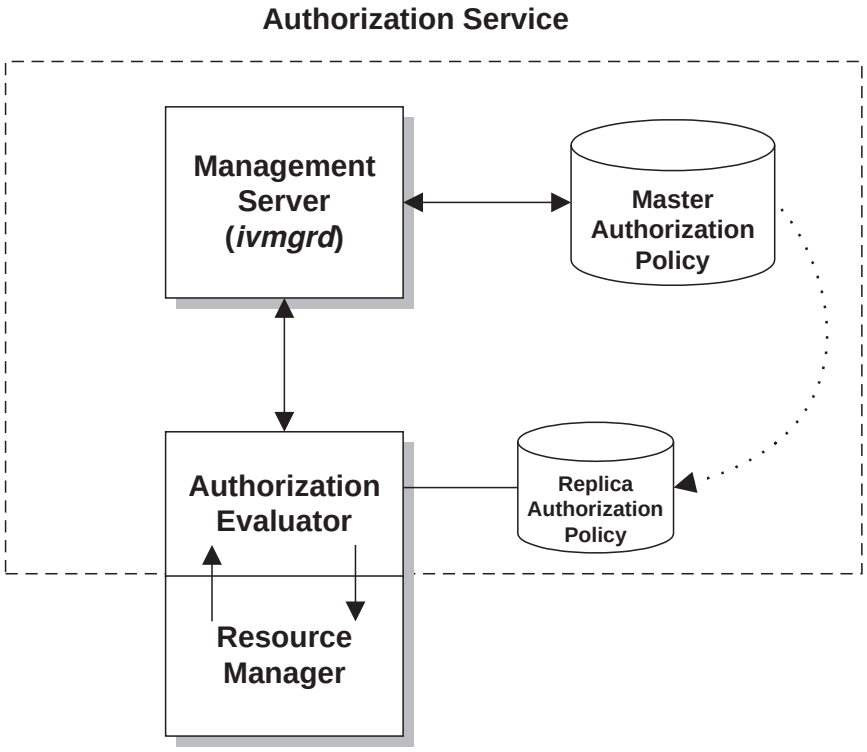
The Management server also maintains location information about the other Policy Director and non-Policy Director servers that are operating in the secure domain.

Note: There must be only one instance of the Management server in any secure domain.

Authorization evaluator

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

The following figure illustrates the main components of the Policy Director Authorization Service:



Policy Director Authorization Service interfaces

The Policy Director Authorization Service has two interfaces where interaction takes place:

Management interface

The security administrator manages the security policy of the network. The security administrator uses the Policy Director Management Console, or the **ivadmin** utility, to:

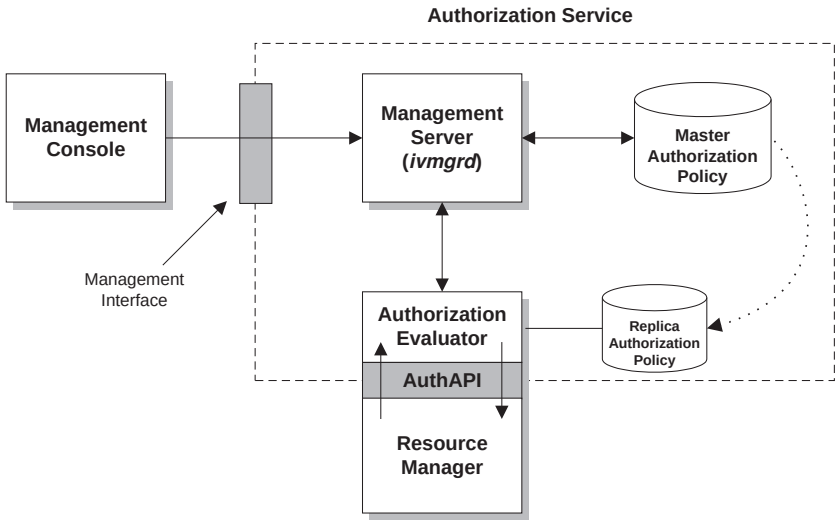
- Apply policy rules (templates) on network resources.
- Register the credentials of participants in the secure domain.

The Management Console applies this security policy data to the primary authorization policy database that uses the Management server.

This interface involves detailed knowledge of the namespace, policy templates, and credentials.

Authorization API

The Policy Director Authorization API passes requests for authorization decisions from the resource manager to the authorization evaluator which then passes back a recommendation. The *Policy Director Programmer's Guide and Reference* contains the details of this API.



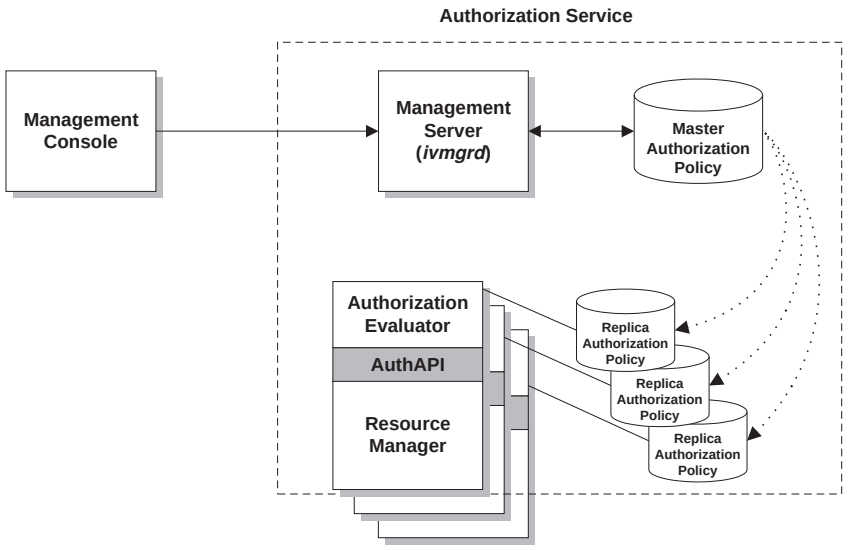
Replication for scalability and performance

You can replicate Policy Director Authorization Service components to increase availability in a heavy-demand environment.

Policy Director always automatically replicates the primary authorization policy database that contains policy rules and credential information. Applications that call the authorization service have two options for referring to this database information:

- The application uses a local cache of the database when configured to work seamlessly with the authorization evaluator. The replication of the database occurs for each application that uses the authorization service in local cache mode.
- The application uses a shared replica that is cached by the remote Policy Director Authorization server component. The replication of the database occurs for each instance of the Policy Director Authorization server. Many applications can access a single authorization server.

Update notification from the Management server triggers the caching process to update all replicas. These update notifications take place whenever changes to the primary authorization policy database occur.



Performance notes:

- The application servers are sent update notifications directly from the Management server. The application servers check the version of the primary authorization policy database every few minutes. Checking ensures that the application servers have not missed an update notification.

  If an update notification fails to reach a server, Policy Director will create a log entry. In both cases a retry mechanism also ensures 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 template in its own cached version of the database. WebSEAL does not have to access the network to obtain this information from the primary database. The result is fast response times (performance) for authorization checks.
- The calling application server does not cache individual authorization results.

Network security policy

How you control user participation and group participation in the domain determines the security policy for a secure domain . The security policy applies rules to resources that require protection. These rules are known as *policy templates*.

The Policy Director Authorization Service enforces this policy by matching a user's identity and credentials with the policy template that is assigned to the requested resource. Policy Director passes the resulting recommendation to the resource manager, which completes the response to the original request.

Network security-policy definition

The Policy Director Authorization Service uses a central database that lists all resources in the secure domain and the policy templates that are assigned to each resource. This primary authorization policy database and the security registry are the key components that help define a network security policy. The security registry contains user accounts and group accounts.

In summary, a network security policy controls:

- Users and groups that are allowed to participate in the secure domain. The security registry contains and maintains this information.
- The level of protection on all objects in the secure domain. The primary authorization policy database maintains this information.

Protected object namespace

The *protected object namespace* is a hierarchical portrayal of resources belonging to a secure domain. The objects that appear in the hierarchical namespace represent the actual network resources.

- **System resource**—the actual physical file, network service, or application.
- **Protected object**—the logical representation of an actual system resource that is used by the Policy Director Authorization Service, the Management Console, and other Policy Director management utilities.

You can attach policy templates to objects in the namespace in order to provide protection of the resource. The Policy Director Authorization Service makes authorization decisions based these templates.

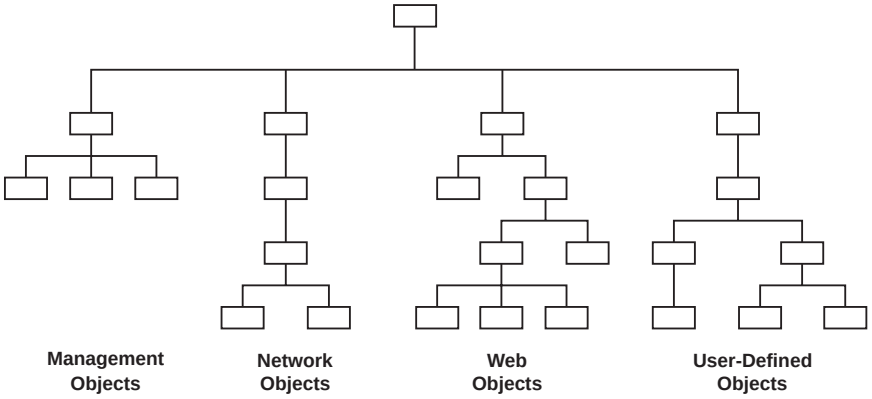
Policy Director uses the following namespace categories:

Web objects
: These objects represent anything that an HTTP URL can address, such as static Web pages and dynamic URLs. You can convert static Web pages and dynamic URLs to database queries or some other type of application.

Network objects
: These objects represent TCP-based applications (such as Telnet and FTP) that map to the TCP network addresses (ports) that are being used by the applications.

Management objects
: These objects represent the management activities you can perform using the Policy Director Management Console. The objects represent the tasks necessary to define users and set security policy. Policy Director supports delegation of management activities and can restrict an administrator's ability to set security policy to a subset of the namespace.

User-defined objects
: These objects represent tasks or network resources that are protected by applications that use the Policy Director Authorization Service, which in turn uses the Policy Director Authorization API.



Definition and application of policy templates

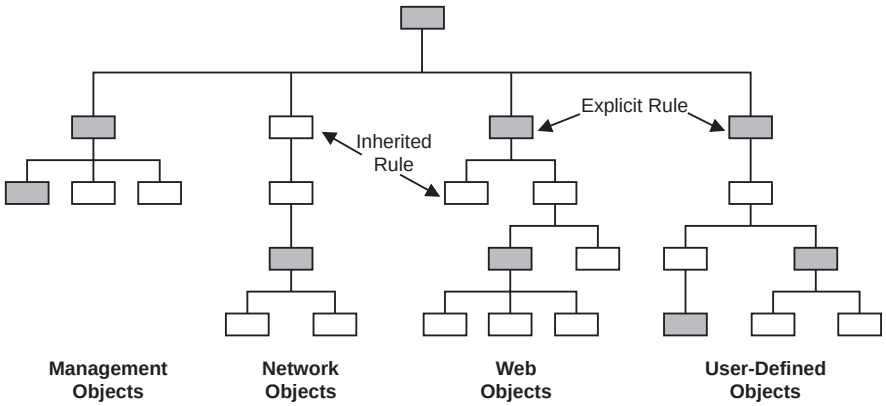
Security administrators protect system resources by defining rules (policy templates), and applying these templates to the object representations of those resources in the namespace.

The Policy Director Authorization Service performs authorization decisions that are based on the policy templates that are applied to these objects. When Policy Director permits a requested operation on a protected object, the application responsible for the resource carries out this operation.

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

Explicit and inherited policy

You can explicitly apply or inherit policy. The Policy Director protected object namespace supports inheritance of security policy attributes. This is an important consideration for the security administrator who manages the namespace. The administrator only needs to apply explicit policy templates at points in the hierarchy where the rules must change.



Examples of policy templates types include:

- Hard-coded rules
- External authorization capability
- Special secure labels
- Access control lists

Access control list

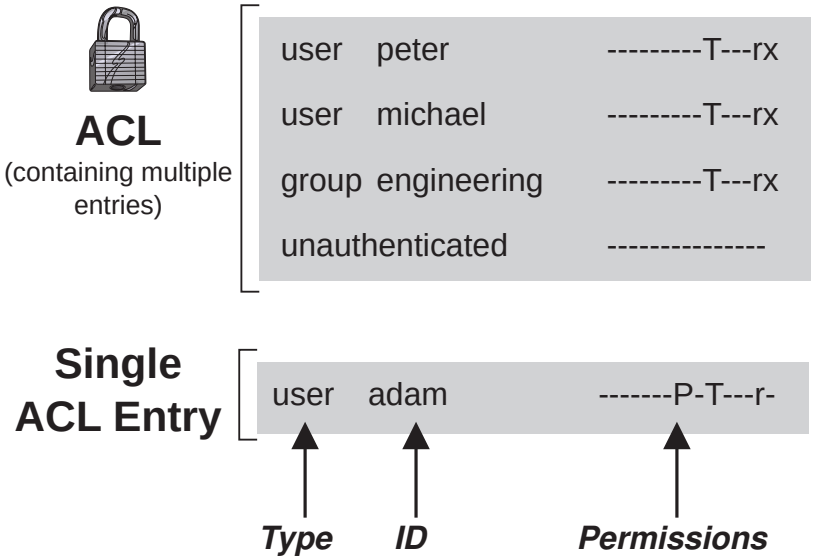
An access control list (ACL) is an example of a policy template. Policy Director uses ACLs as its primary policy template.

An ACL is the set of controls (permissions) that specifies the conditions necessary to perform certain operations on that resource. ACL definitions are important components of the security policy that is established for the secure domain. You use ACLs, like all policy templates, to stamp an organization's security policy onto the resources that are represented in the protected object namespace.

An ACL specifically controls:

- The operations performed on the resource.
- The people who can perform these operations.

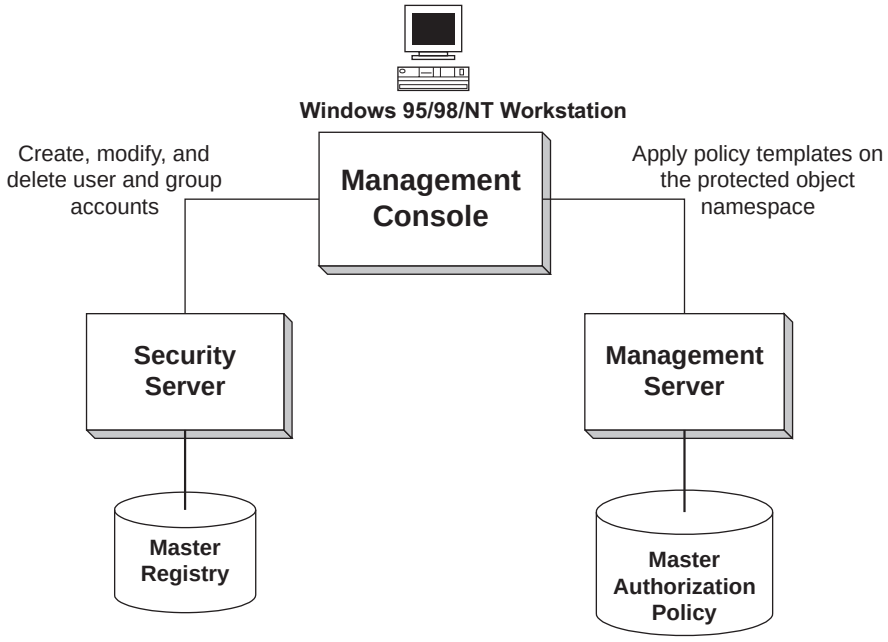
One or more entries, which include user designations and group designations, plus either their specific permissions or rights make up an ACL.



Policy administration

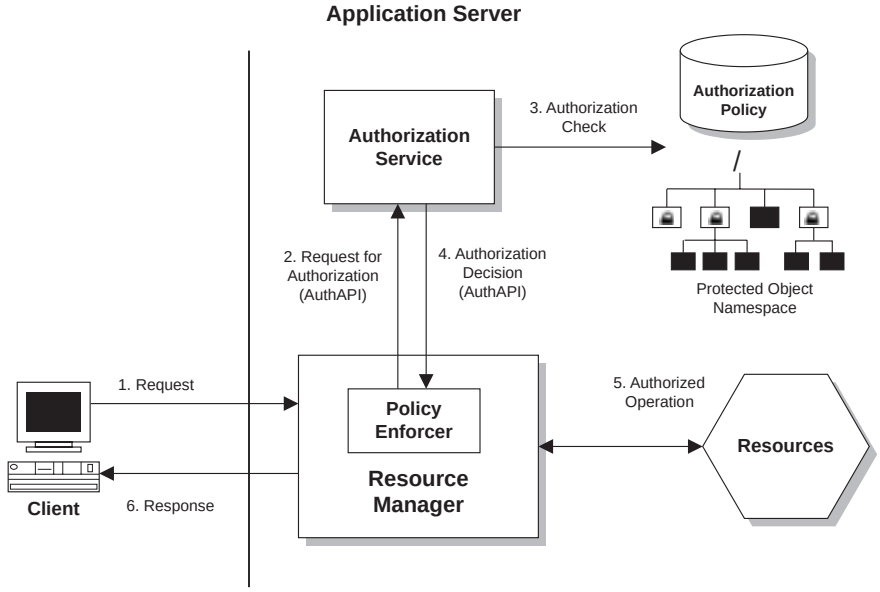
The Policy Director Management Console is a Java-based graphical application that is used to manage security policy in a Policy Director secure domain. The optional **ivadmin** command-line utility provides the same administrative capabilities as the Management Console.

From the Management Console, or by using the **ivadmin** utility, you can manage the Security server registry, the primary authorization policy database, and all Policy Director servers. You can also add or delete users and groups, and apply policy templates or ACLs to network objects.



Step-by-step authorization process

The following diagram illustrates the complete authorization process:



1. Policy Director directs an authenticated client request for a resource to the resource manager server. The policy enforcer process intercepts the request.

   The resource manager can be WebSEAL (for HTTP and HTTPS access), NetSEAL (for TCP/IP network access), or a third-party application.
2. The policy enforcer process uses the Policy Director Authorization API (see “Policy Director Authorization API” on page 44) to call the Policy Director Authorization Service for an authorization decision.

3. The authorization service performs an authorization check on the resource that is represented as an object in the Policy Director protected object namespace. The policy template applied to the object is checked against the client's credentials.
4. A recommendation to the resource manager (using the policy enforcer) is returned for the decision to accept or deny the request.
5. If the authorization check approves the request, the resource manager will pass the request on to the application responsible for the resource.
6. The client receives the results of the requested operation.

Policy Director Authorization API

The Policy Director Authorization *application program interface* (API) allows Policy Director applications and third-party applications to query the Policy Director Authorization Service to make authorization decisions.

The Policy Director Authorization API is the interface between the resource manager (that requests the authorization check) and the authorization service itself. The Policy Director Authorization API allows the policy-enforcing application to ask for an authorization decision. However, the Policy Director Authorization API shields the application from the complexities of the actual decision-making process.

The Policy Director Authorization API provides a standard programming model for coding authorization requests and decisions. The Policy Director Authorization API lets you make standardized calls to the centrally managed authorization service from any legacy or newly developed application.

You can use the Policy Director Authorization API in one of two modes:

Remote Cache Mode

In this mode, Policy Director initializes the API to call the remote Policy Director Authorization server (ivacld) to perform authorization decisions on behalf of the application. The Policy Director Authorization server maintains its own cache of the replica authorization policy database. Use this mode for handling authorization requests from application clients.

See "Remote cache mode" on page 46.

Local Cache Mode

In this mode, Policy Director initializes the API to download and maintain a local replica of the authorization database for the application. Local cache mode allows the application to perform all authorization decisions locally, which results in better performance and reliability.

However, the overhead of database replication and the security implications of using this mode make it best suited for use by trusted application servers. Trusted application servers include WebSEAL and NetSEAL.

See "Local cache mode" on page 47.

A primary value and benefit of the Policy Director Authorization API are its ability to shield the user from the complexities of the authorization service mechanism itself. Policy Director hides issues of management, storage, caching, replication, credential formats, and authentication methods behind the Policy Director Authorization API.

The Policy Director Authorization API also works independently from the underlying security infrastructure, the credential format, and the evaluating mechanism. The Policy Director 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.

The Policy Director Authorization API provide support for the following platforms:

- Microsoft Windows NT, Windows 98, and Windows 95
- IBM AIX Version 4.3
- Sun Solaris Version 2.6

Authorization API examples

The WebSEAL and NetSEAL authorization services perform access control on URLs and ports respectively. A third-party application can use the Policy Director Authorization API to perform access control on very specific and specialized processes.

Example 1: You can design a graphical user interface (GUI) to dynamically show task buttons as active or inactive, according to the results of the authorization check.

Example 2: The following figures demonstrate another use of the Policy Director Authorization API. This figure illustrates a request for a CGI transaction by a Web application.

The lowest level of authorization, illustrated in Figure A, involves an “all-or-nothing” access control on the URL. This coarse-grained level of authorization only determines whether the client can run the CGI program. By allowing access to the CGI application, no further control is available to resources that are manipulated by the CGI application.

In Figure B, access controls are a set of resources that the CGI program manipulates. The Web application is configured to use the Policy Director Authorization API. Now, the CGI program can call the Policy Director Authorization Service to make authorization decisions on the resources it manipulates. The authorization decisions can be based on the identity of the requesting client.

Figure A

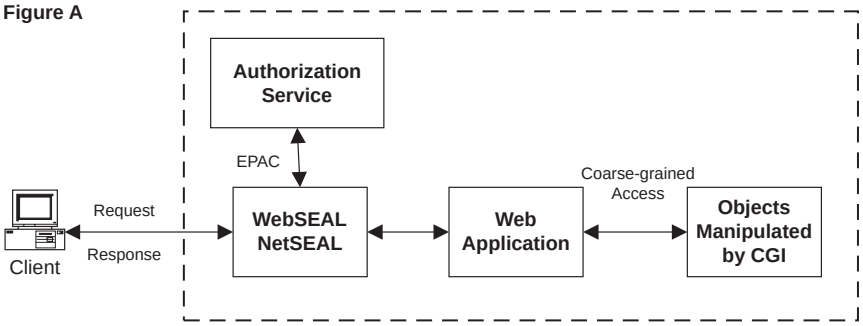
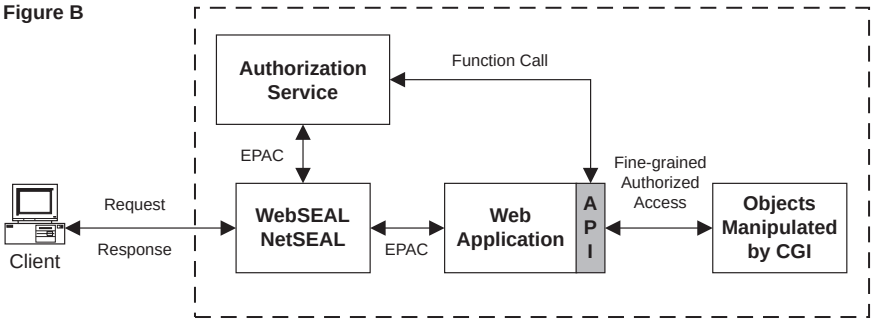


Figure B

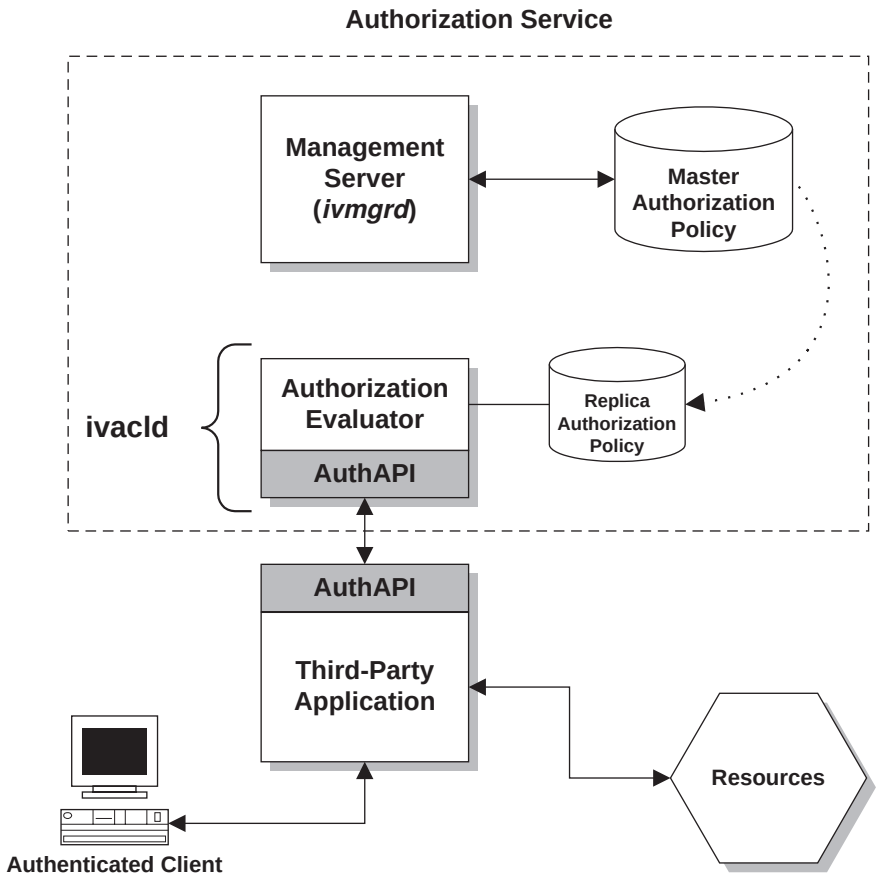


Remote cache mode

In *remote cache mode*, applications use the function calls that are provided by the Policy Director Authorization API to communicate to the remote Policy Director Authorization server (ivacld). The Policy Director Authorization server functions as the authorization decision-making evaluator and maintains its own replica authorization policy database.

The Policy Director Authorization server makes the decision and returns a recommendation to the application by using the API. The server can also write an audit record that contains the details of the authorization decision request.

There must be an Policy Director Authorization server that runs somewhere in the secure domain. The Policy Director Authorization server can reside on the same machine as the application or on another machine. You can also install the Policy Director Authorization server on more than one machine in a secure domain to allow for high availability. The Policy Director Authorization API transparently fails over when a particular Policy Director Authorization server fails.



Local cache mode

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

The local replica is persistent across calls of the application. The API starts in replica mode. When it starts, it checks for any updates to the primary Authorization Policy database. These updates are ones that might have occurred since the local replica was built.

Figure A

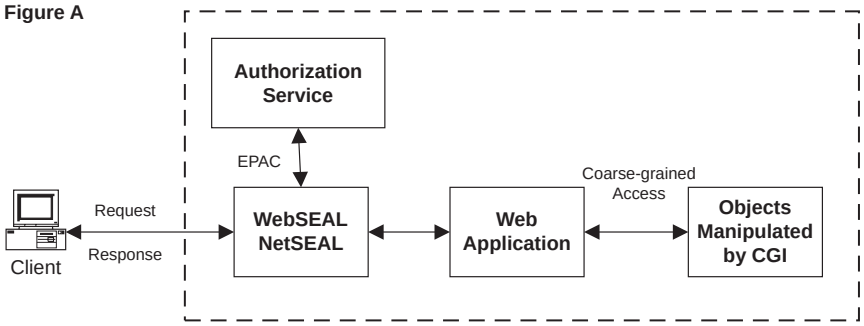
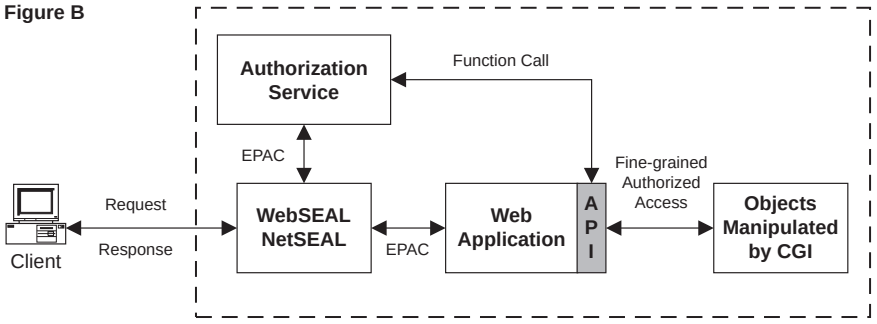


Figure B



External authorization capability

In some situations, the standard set of Policy Director permissions might not be able to express all the authorization rules required by an organization's security policy. Policy Director provides optional external authorization capability to accommodate any additional authorization requirements.

An external authorization service allows you to impose additional authorization controls and conditions that are dictated by a separate, external authorization server program.

Extension of the authorization service

The Policy Director Authorization Service builds in an external authorization capability automatically. If you configure an external authorization service, the Policy Director Authorization Service simply will incorporate the new controls and conditions into its evaluation process.

Applications that use the Policy Director Authorization Service include WebSEAL, NetSEAL, and any application that uses the Policy Director Authorization API. These applications benefit from the additional, but seamless, contribution of a configured external authorization service. Any addition to the security policy through the use of an external authorization service is transparent to these applications and requires no change to the applications.

The external authorization service architecture allows the full integration of an organization's existing security service. An external authorization service preserves a company's initial investment in security mechanisms. This external authorization services allow legacy servers to be incorporated into the Policy Director authorization decision-making process.

Setting up an external authorization service requires these general steps:

1. Write a server program that will be referred to during an authorization decision.
2. Register the external authorization service with Policy Director.

After registering the service, a new permission that represents this service appears in the Policy Director Management Console. You can now use this permission in any ACL entry.

When encountering the permission during an authorization check, the external authorization service is referred to for additional authorization decisions.

Complete details describing the setup of the external authorization service are in the *Policy Director Programmer's Guide and Reference.*

Conditions on resource requests

You can also use an external authorization service to impose a specific condition on a successful or unsuccessful access attempt.

Examples include conditions, such as:

- Causing an external auditing mechanism to record the successful or unsuccessful access attempt.
- Actively monitoring the access attempt and cause an alert or alarm for unacceptable behavior detection.
- Billing and micro-payment transactions.

Authorization evaluation process

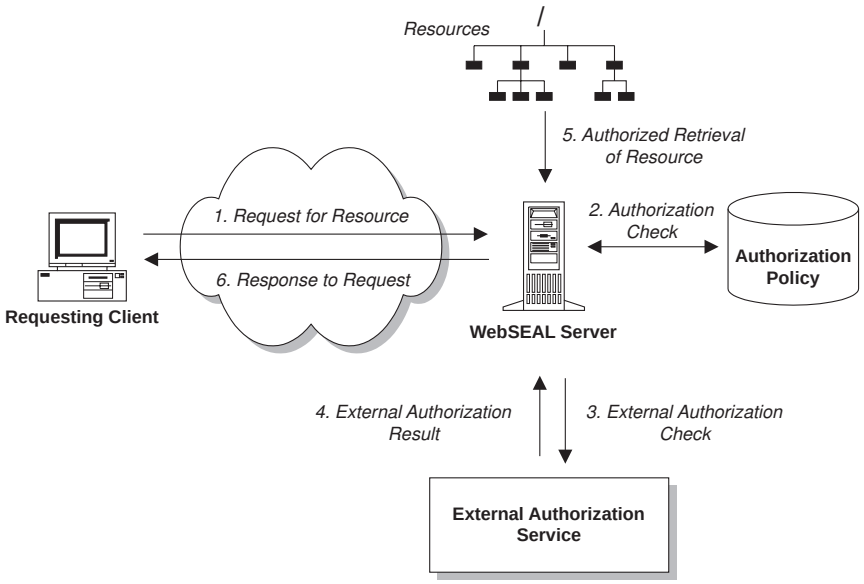
An authorization decision that incorporates an external authorization service takes place in the following manner:

1. Makes an ACL check to determine that the set of permissions granted to the requesting user.
2. Sends an authorization request that uses authenticated RPC to each external authorization service whose permission appears in the ACL.

   This external authorization check occurs regardless of whether or not the necessary permission is granted to the user.
3. Sums all authorization decision results.

   Denials can come from either the Policy Director ACL check or any external authorization check. If any denial occurs, the Policy Director Authorization Service will deny the authorization request.

Example: The following figure illustrates an authorization decision that involves a WebSEAL server and an external authorization service.



In this example, the purpose of the external authorization service is to impose time restrictions on accessing objects. The character assigned to represent the permission in the Management Console is k.

1. The Policy Director WebSEAL server receives a request from a client for access to a sensitive engineering document. The client is a member of the Engineering group.
2. The WebSEAL server first checks the replica authorization policy database to determine the permissions that are assigned to that document object.

   ```
   group engineering rk
   ```

   If the ACL entry does not contain the external authorization service permission, the final authorization decision will be based solely on this information.

   In the above example, the ACL entry contains a standard read permission. The entry also contains an additional (k) permission that refers to an external authorization server to supplement the authorization evaluation.
3. Policy Director uses an authenticated RPC to the external authorization server to send a request. The set of granted permissions are determined in step 2. Policy Director sends this set of permissions with this request so that the external authorization server can base its decision on this information.

   In this example, the design of the external authorization server allows you to set time boundaries on the ability to access this document. Access to the document happens only when the request occurs between the hours of 8AM and 6PM, Monday through Friday.
4. The client in this example has made the request at 10AM on a Tuesday. The server sends a successful response back to the WebSEAL server.

   The sum total of all the above authorization decisions results in a final recommendation to allow access to the document object.
5. The WebSEAL server retrieves the document resource.
6. The WebSEAL server allows the client to view the document.

Refer to the *Policy Director Programmer's Guide and Reference* for advanced details on putting into effect the external authorization service.

Implementation strategies

Policy Director allows you to accomplish an external authorization service in several ways:

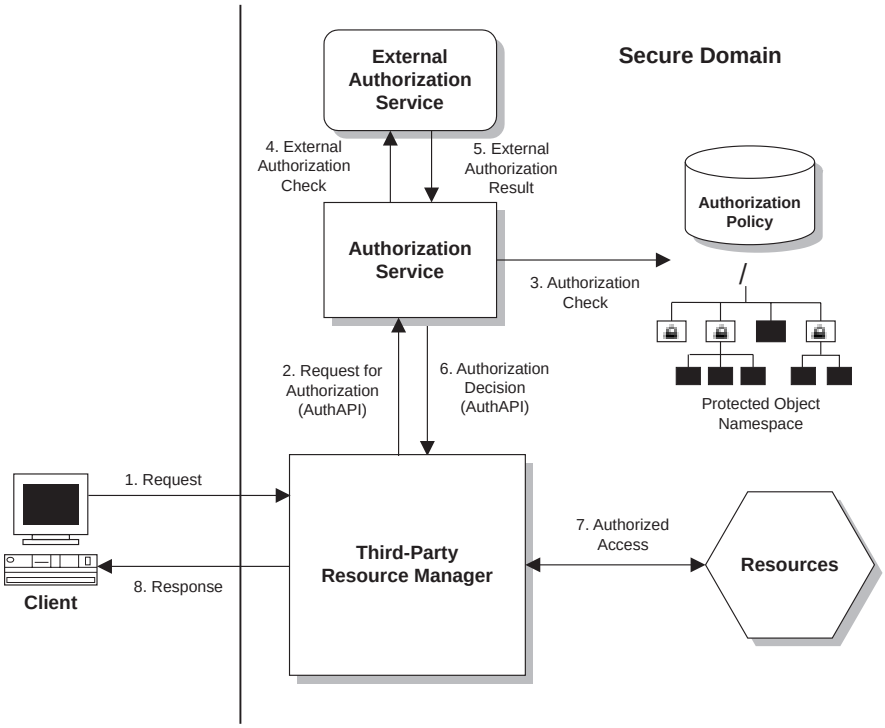
- You can add any number of external authorization services to your secure domain to fulfill a variety of authorization evaluations. Each different permission in an ACL represents a service.
- You can call more than one external server for a given permission by chaining multiple external authorization servers. Each one passes the authenticated identity to the next server in the chain and collects the results from the servers downstream.
- You can replicate an external authorization service throughout the secure domain.

Each of these implementations are independent of one another and can be used in any combination.

Extensibility and flexibility

The combination of the Policy Director Authorization API and an external authorization service provides a highly extensible and flexible solution for carrying out a security policy.

The following figure illustrates the extensible architecture possible using the combined features of the Policy Director Authorization API for a third-party application and an external authorization service:



Chapter 4. Introducing the Management Console

The Policy Director Management Console is a Java-based graphical application that is used in a distributed network to securely manage all Policy Director components. From the Management Console, you can manage the Security server registry, the primary authorization policy database, and all Policy Director servers. The Management Console also allows you to add and delete users or groups and apply ACLs.

This chapter includes:

Management Console overview

The Policy Director Management Console is a Java-based graphical application that is used to manage security policy in a Policy Director secure domain. The optional **ivadmin** command-line utility provides the same administration capabilities as the Management Console.

From the Management Console, or by using the **ivadmin** utility, you have the ability to:

- Change the registry database (accounts).
- Change the primary authorization policy (ACL) database.
- Add and delete users.
- Add and delete groups.
- Apply policy templates or ACLs to objects.
- Add, delete, or change Global Sign-on (GSO) resources, resource groups, and resource credentials
- Add, delete, or change proxy user (optional)

Management Console features

The Management Console provides tools for performing tasks and displays information in distinct regions of the Management Console window. The primary tools and display regions include:

- Task tabs
- Management task panels (top and bottom panels)
- Action buttons
- Toolbar
- Bulletin Board (default bottom panel)
- Status Bar
- Title Bar

Management task panel tools

These tools are provided for the management task panels:

- Task tabs
- Management task panels (top and bottom panels)
- Action buttons

Task tabs

The Management Console provides these main *task tabs* for performing management operations:

- Login
- Users
- Groups
- GSO Resources
- GSO Resource Groups
- ACLs
- Object Space
- Proxy User (optional)

Management task panels

Each task tab produces a management task panel that includes a collection of information views that are presented in the top panel of the Management Console.

The management tasks include:

Login	Is the entry point for logging in and logging out of the Management Console.
Users	Allow you to create and maintain user participants in the secure domain.
Groups	Allow you to create and maintain groups in the secure domain.
GSO Resources	Allow you to create and maintain GSO resource information.
GSO Resource Groups	Allow you to create and maintain GSO resource group information.
ACLs	Allow you to create and maintain policy templates or ACLs.
Object Space	Allows you to attach policy templates to and remove policy templates from objects in the namespace.
Proxy User	Allows you to create and maintain proxy user information.

Action buttons

A specific set of action buttons accompanies each task tab. Use these action buttons to perform administration operations. The buttons appear on the left side of the panel. Action buttons change and update the appropriate database.

Panel view types

You can display information in the management task panels in one of three views. Each view type has specific characteristics:

Detail view

A detail view contains dynamic fields for data entry.

List view

You can sort a list view in ascending and descending order by clicking in the title bar for the appropriate column. Some lists have query capability.

Tree view

You can expand and contract a tree view.

When an item is selected and is active in any view, the highlight appears blue.

When an item is selected but is not currently active, a gray color is used.

Toolbar

The toolbar appears at the top of the Management Console window and includes buttons that activate specific Management Console functions.

The **Move Task Down** button repositions the current management task in the top panel to the bottom panel.

The **Move Task Up** button repositions the current management task in the bottom panel to the top panel.

The **Pin View** button takes the currently active information in the top panel and places a copy of this information into a bottom panel. Currently active information in the top panel includes information such as a list of groups or users. A new tab also appears for the panel. Information in this panel is only a static copy and is not dynamic. However, you can still expand and contract tree views, such as the object space tree and group list.

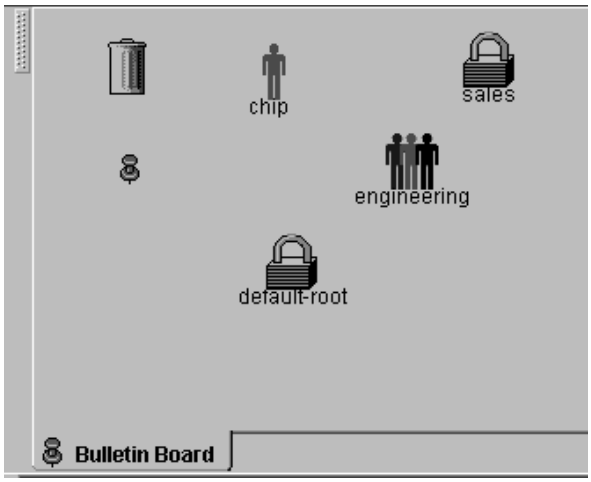
The **Erase** button clears the currently selected view from the Management Console. This action erases the view only; the actual database information is not affected.

The **Stop** button discontinues the action currently in progress and returns control of the Management Console back to the administrator. In actuality, it ignores the result of the current action, and the Management Console behaves as if the operation failed.

Bulletin Board

The Bulletin Board is the default lower panel of the Management Console. You use the Bulletin Board as a temporary storage location for any object you might want to use multiple times during an administration session. Objects can include files, ACLs, lists of users and groups, and attributes. You drag and drop object icons onto the Bulletin Board and use them in administration tasks as needed.

Standard icons on the Bulletin Board include the **Trash** icon and the **Pin View** icon.



Trash icon

You can remove icons that are stored on the Bulletin Board by dragging them to the **Trash** icon.

Pin view panel

You can select (highlight) specific objects and information from any other management task panels and drop them on the Bulletin Board's **Pin View** icon.

This action results in a new lower panel that displays a copy of the selected information. A new tab also appears.

Information in this panel is only a static copy and is not dynamic. However, you can still expand and contract tree views, such as the Object Space tree and Group list.

If the active information is a list of groups, only those groups that are selected (highlighted) will be placed in the Pin View panel.

To close a Pin View panel, click the close box button in the upper right-hand corner of the window.

Note: The same action occurs when you select the information and click the **Pin View** button in the toolbar.

Status bar

The status bar at the bottom of the Management Console displays status and error messages.

- General status information appears in **black**.
- Warnings appear in **blue**.
- Errors appear in **red**.

A status indicator icon appears to the left the message area:


Success status icon


Warning status icon


Error status icon

If you double-click the status indicator icon, the status bar will display a:

- Management Console version
- Secure domain that is affected by Management Console activity
- Java version

Title bar

The Management Console's title bar displays two important pieces of information:

- The secure domain that is currently affected by Management Console activity.
- The logged-in status of the user (either UPDATE or READ-ONLY).

When the primary Security server registry is not available, you cannot make modifications to account data through the Management Console. In this rare situation, the Management Console obtains current account information from replicas of the registry. An error message appears when you attempt to change information that affects the registry.

Login management task

The Login task panel is the entry point to the Management Console for an authenticated user. If this event is successful, the full set of management task tabs will appear.

When the user logs out, all context is lost and task tabs are removed.

Task tab

Task tab: **Login**

Management task

The Login management task panel contains fields for **User Name** and **Password** entries. The **Secure Domain** field initially displays the default secure domain that is configured for NetSEAT.

The Login menu allows you to select a different secure domain in which to perform Management Console tasks.

Action buttons

Action buttons for **Login** are:

- Login
- Logout

Users management task

The Users management task panel allows you to create and maintain user participants in the secure domain. When you select a user from the list view, or tree view, the fields in the detail view populate with the current data.

Task tab

Task tab: **Users**

Management task

The Users management task panel contains a Users list view, a User Detail view, and a Groups list view.

The Groups view contains additional tabs for GSO Resources and GSO Resource Groups.

Action buttons

Action buttons for **Users** are:

- New
- Get
- Save
- Delete

Groups management task

The Groups management task panel allows you to create and maintain groups in the secure domain. When you select a group from the list view, or tree view, the fields in the detail view populate with the current data.

Task tab

Task tab: **Groups**

Management task

The Groups management task panel contains a Groups list view, a Group detail view, and a User ID list view.

Action buttons

Action buttons for **Groups** are:

- New
- Get
- Save
- Delete

GSO Resources management task

The GSO Resources management task panel allows you to create and maintain GSO resources in the secure domain. GSO resources are always Web resources. When you select a GSO resource from the list or tree view, the fields in the detailed view populate with the current data.

Task tab

Task tab: **GSO Resources**

Management task

The GSO Resources management task panel contains a Resources list view and a Resources Detail view.

Action buttons

Action buttons for **GSO Resources** are:

- New
- Get
- Save
- Delete

GSO Resource Groups management task

The GSO Resource Groups management task panel allows you to create and maintain GSO resource groups in the secure domain. A *resource group* refers to a group of Web servers, where all the servers in the group have the same set of user IDs (userids) and password. When you select a GSO resource group from the list or tree view, the fields in the detailed view populate with the current data.

You can create a single resource credential for all the resources in the resource group. Policy Director uses a single resource credential for a resource group instead of a resource credential for each resource in the resource group.

Task tab

Task tab: **GSO Resource Groups**

Management task

The GSO Resource Group management task panel contains a Resource Groups list view, a Resource Group Detail view, and a GSO Resources list view.

Action buttons

Action buttons for **GSO Resource Groups** are:

- New
- Get
- Save
- Delete

ACLs management task

ACLs management task panel allows you to create and maintain ACL policy templates. When you select an ACL from the **ACL List** view, the fields in the ACL Definition view populate with the current data.

Task tab

Task tab: **ACLs**

Management task

The ACLs management task panel contains an ACL List view, an ACL Definition detail view, and an ACL Entry detail view with a permissions tree view.

Action buttons

Action buttons for **ACLs** are:

- New ACL
- New Entry
- Save
- Delete
- Get
- List
- Where Used

Object Space management task

The Object Space management task panel allows you to attach ACLs to and remove ACLs from objects in the namespace.

When you select an object in the Object Space tree view, the sequence of inherited ACLs appears in the Inherited ACLs tree view. This list represents all objects with explicitly set ACLs that affect the permissions of the selected object because of inheritance.

Task tab

Task tab: **Object Space**

Management task

The Object Space management task panel provides three distinct information views that are selected from the panel's sub-tabs:

- **Inherited ACLs** view (default)

  This view is the default view. It displays the chain of ACLs that affects the selected object. An arrow always points to the ACL that immediately affects the selected object in the Object Space tree view.
- **Edit ACL** view

  This view provides the portion of the ACLs management panel that allows you to directly change ACL attributes.
- **Inheritance** tree view

  This view displays a tree view of the ACL inheritance chain that is directly affecting the selected object.

Action buttons

Action buttons for **Object Space** are:

- Attach ACL
- Remove ACL
- Find ACL
- Save ACL
- List

Proxy User management task

Policy Director, in combination with a corporate firewall system, can fully protect the enterprise intranet from unauthorized access and intrusion. The Proxy User management task panel allows you to create and maintain proxy users in the secure domain. When you select a proxy user from the tree view, the fields in the detail view populate with the current data.

Task tab

Task tab: **Proxy User**

Management task

The Proxy User management task panel contains a Proxy Users list view and a Proxy User Detail view.

The Proxy User Detail management task panel contains fields that display the default proxy user information.

Action buttons

Action buttons for **Proxy User** are:

- Save
- Delete

Management Console properties and controls

The Management Console requires the Policy Director NetSEAT client on Microsoft Window NT or Windows 95 or 98 to perform management tasks through secure communication channels. For AIX and Solaris, the Management Console uses the system's DCE client for performing management tasks.

Dragging and dropping

You can perform many Management Console operations by using the mouse to drag objects from one location and drop them on another location. For example, you can add a user to a group by dragging the **User** icon from the User List. Then, you just drop it onto a group in the Groups tree view.

The cursor shape changes to a hand when it is over an icon that you are permitted to drag.

You can drag and drop these objects:

- User
- Group
- ACL
- ACL entry
- Objects in the namespace
- GSO Resources and GSO Resource Groups
- Proxy User

All drops that cause database updates produce a confirmation-alert dialog box.

You can perform data queries with the drag and drop method. For example, when you drag an **ACL** icon from the Bulletin Board to the ACL Definition view, the fields populate with the current data.

If you drop an object (icon) on a location that does not accept this operation, the object (icon) will animate back to its origin.

Drag and drop operations work only within the context of the Management Console.

Performing top and bottom panel activity

Management Console update operations take place in both the top and bottom panels. You can drag objects from the bottom panel and drop them on the top panel. Save any modifications to either panel to the database.

Policy Director always synchronizes ACL information that is displayed in both the Object Space panel and the ACL panel.

Selecting multiple items in a list

You can select items in lists and tables by using these standard Windows® selection techniques:

- A single click on an item selects the item.
- Hold the Ctrl key to select additional items at the same time.
- A click followed by a Shift + click selects the block of text between the two clicks.
- All items in a list can be selected using **Ctrl + a**.

Editing a data entry field

When editing a data entry field, remember that:

- You can use the Enter key to toggle a text edit field on and off.
- You can use the Esc key to restore the previous data entry in a field when editing.
- On Windows client machines, you can use standard Windows keystrokes to perform Management Console-specific copy, cut, and paste.
- After exiting a field where data was changed, a red indicator appears in the top left corner of the view. Click the **Save** button to commit all changes to the database.
- You can use drag and drop to populate some data fields.

Drag and drop operations work only within the context of the Management Console.

Querying from lists

Many list views have query capabilities. The query icon appears in the top left corner of the list view window.

Navigating

To navigate between fields:

- In a detail view, the Tab key moves the cursor from one data entry field to the next.
- When the cursor is in the last field of a detail view, the Tab key moves the cursor to the next view. The cursor moves from left to right.
- When the cursor is in a list or tree view, the Tab key moves the cursor to the next view. The cursor moves from left to right.
- Shift + Tab moves the cursor to the next view from right to left.
- The Home key moves the cursor to the top of a view. The End key moves the cursor to the bottom of the view.

In a detail view, the Home + End keys move the cursor to the top and bottom of the view when all fields are inactive. If any field is active, Home + End will move the cursor to the beginning and end of the active field.

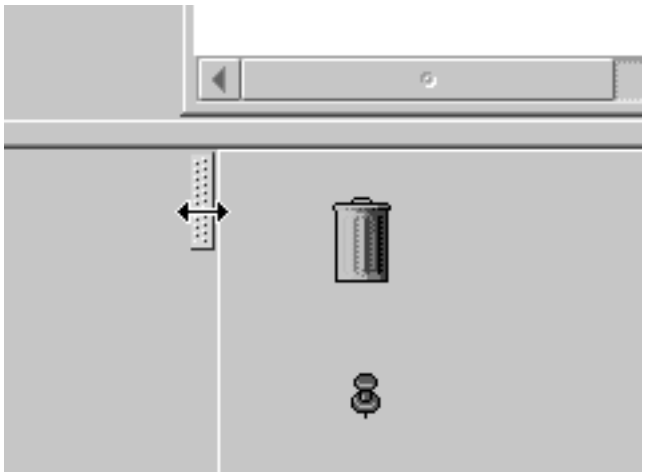
Using object icons

When using object icons, remember that:

- A unique icon represents each object type in the namespace graphically.
- Each task tab displays the icon of the object that is affected by that management task.
- The detail views display the icon of the object that is being edited in the upper left-hand corner of the view.
- The dragging and dropping operation displays the icon of the selected object.
- An object must be dragged using its icon.
- The cursor shape changes to a hand when it is over an icon you can drag.

Resizing views by using the splitter icon

Every management task panel contains one or more views. Use the splitter icon located at the top of the view's left border to resize these views. Also, you can also resize columns within list views by moving the cursor to the splitter icon between two column headings.



When you move the cursor across the splitter, the cursor shape changes to a double arrow, indicating that you can resize the view.

Sorting lists

You can toggle information in list views between ascending and descending sort order by clicking in the column's title bar. An icon on the right side of the title bar indicates the current sort order.

Selected items in a list remain selected after sorting the list.

Expanding and contracting tree views

A tree view is similar to the Windows Explorer display of files and directories. To be able to expand or contract a node, you must see an icon—a box with a plus or minus. Double-click this icon to toggle between expanded and contracted tree views. The keyboard equivalent is **Ctrl + e**.

If a node does not have objects or nodes below, then the expand/contract indicator will disappear after clicking on it.

You can refresh the entire tree by expanding and contracting the root node.

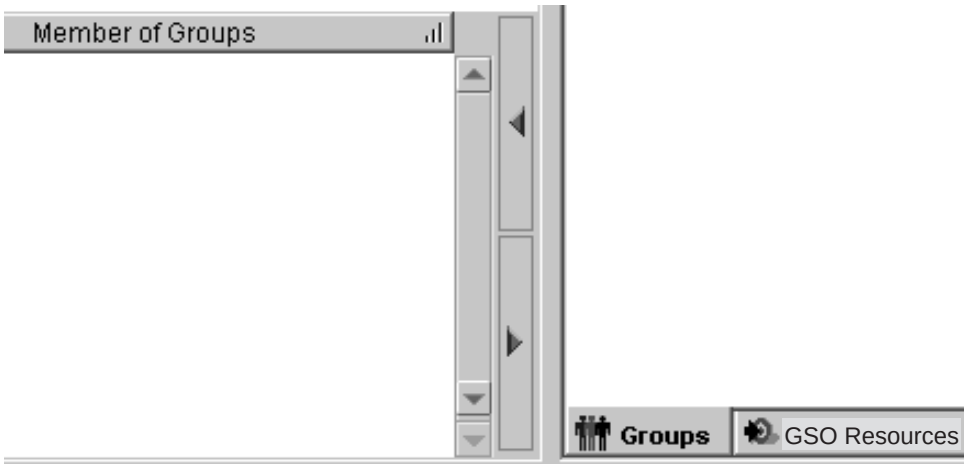
Using object space shift node arrows

The title bar of the **Object Space** view contains two blue arrows. These two blue arrows allow you to restrict the focus of the tree view to a branch of the tree.

- **Left arrow**—this arrow shifts the selected node or object completely flush left so that it appears in the root position.
- **Right arrow**—this arrow shifts the tree back one node at a time with each click. After using the left arrow, use the right arrow (one or more clicks) to restore the tree to its normal orientation.

Using selection arrows

Selection arrows move selected information from one window view to another window view. The effect is to populate the field of the second window with new information.



Chapter 5. Managing user accounts and groups

The Policy Director security model requires the authentication of a user identity before authorizing access to an object. You can accomplish authentication of user identities by matching user identities and encrypted passwords with account information in a primary registry database. By default, Policy Director uses the LDAP registry. A security administrator can use the Management Console to create and manage the users and groups who participate in the secure domain.

This chapter includes:

Understanding users, groups, and accounts

The Policy Director security service relies on a primary account registry—a database that contains information about users, groups, and accounts for the secure domain. By default, Policy Director uses the LDAP registry.

Users

Users are the principals, or participants, of the secure domain. Users can include human users, server processes, machines, or other secure domains.

A user is an entity that can participate in an authentication exchange with another user. *Authentication* is the process of verifying that a user is who it claims to be. Every user has a password, or secret key, that is used for authentication.

You can associate users with access control permissions. Each user has its own Universal Unique Identifier (UUID) that always remains constant, even when the user's name changes. Policy Director passes the UUID to the security service as a security credential.

Groups

Groups are collections of users that are identified by a group name. Groups can represent different levels of security roles, or responsibility. Policy Director treats users who are members of the same group identically for security purposes. A user can belong to more than one group, depending on the user's roles and duties.

Groups make it easier to administer and manage security policy. Use ACLs to define a group's security policy. When a user's role, responsibility, or existence has changed, all ACL entries for this user must also change. Without groups, the changing all the ACLs that contain entries for that individual user a user would be a nearly impossible task.

A group ACL entry can represent the user's role or responsibility. If so represented, then the only management required will be to add or remove the user's membership in the group.

Groups, like users, have UUIDs in addition to the group name. This group UUID represents a component of the user's security credentials.

Accounts

An *account* is a relationship that involves a user, associated groups, and relevant security information. An account in the registry defines a user's network identity. Accounts define a network identity by associating a user with one or more groups and any related security information, such as the password that is used for authentication.

You must create an account for any user that engages in communication across the secure domain, regardless of whether the communication is authenticated.

You can associate user accounts with the user's password and any information that is used when the user logs in to the secure domain. Account information can include the user's home directory, login shell, and authentication policy (passwords). This information helps control a user's access to the secure domain.

Note: You must add a group to the registry before using it in a user account.

Managing groups

A group is collection of one or more users. You usually create groups to represent common departments within an organization (for example, sales, training, and engineering). You can also create task-based group categories (for example, system administrators or a group of users that perform routine backups).

Group categories can simplify access control management within the Policy Director secure domain. You allow new users access to information by assigning them membership in the appropriate group category. This approach eliminates the need to create new ACL entries for every new user.

For example, when a new person joins the engineering department, you would create a new user account that includes membership in the group category `engineering`. The new user can now read all engineering documents that are permitted for the `engineering` group . You do not need to create a new ACL entry for this user in the engineering ACL template.

You use the Management Console to add, change, or remove group entries in the Policy Director security registry.

After you have created one or more group entries, you can assign users to groups. Users can belong to more than one group, reflecting the different roles and responsibilities of the user.

Using the Groups management panel

The security administrator for the secure domain uses the Management Console to create groups:

1. Log in to the Management Console as an administrator, such as `cell_admin`.
2. Click the **Groups** task tab.

   The Groups management task panel appears.

Drag and drop operations work only within the context of the Management Console.

Using action buttons for groups management tasks

You use the **Groups** action buttons to perform Groups management operations. The following table describes the tasks that each action button performs:

Action Button	Description
New	Creates a new group entry for the secure domain.
Get	Retrieves information about a specifically designated group and populates the detailed view windows.
Save	Saves this group entry. The new entry appears in the appropriate list window.
Delete	Removes the selected group from the registry database.

Using group detail fields

The following table describes the fields in the **Group Detail** view of the Management Console:

Field	Description
Group Name	The primary name assigned to a group of the secure domain. The Group Name is a key field that is used when a query is made to the registry.
Description	The informational text string that describes the group. The Description is an optional data field and is not used by the registry.
LDAP	For LDAP: • For cn =, type in a common name, such as `credit` • For dn =, type in a distinguished name, such as `cn=credit,o=IBM,ou=Austin,c=US`

Creating a new group

To create a new group:

1. Click the **New** action button.

   Activate entry fields for **Group Name** and **Description** in the Group Detail area.
2. Type the new group name, and optionally type a description of the group in the **Description** field.
3. Add the required LDAP registry information if LDAP is your default registry.
4. You can add users to the new group by dragging user icons from the User ID list view to the User ID window of the Group Detail view.

   Also, you can use selection arrows to move users in and out of the User ID panel.
5. Click the **Save** action button.

   A new group entry appears in the Groups list view.

Changing group details

To change the user membership or name of an existing group:

1. Select a group from the Groups list view.

   The Group Detail view populates with current information about the group.
2. In the Group Detail area, select the field to change (refer to “Using group detail fields” on page 69) and type the new values.
3. You can also add new users to the group or delete users from the group.

   From the User ID column, double-click the icon of a user. Then, drag and drop the new user to the User ID window in the Group Detail view. You can also use the selection arrows.
4. Click **Save**.

Removing a group

To remove a group:

1. From the Groups list view, select the name of the group you want to delete.

   The list of group members appears in the Group Detail view.
2. Select each group member, one-at-a-time, and remove each user from the group.
3. From the Groups list view, select the group.
4. Click **Delete** to completely remove the group entry.

Managing user accounts

Users requesting access to services and objects in the secure domain must authenticate themselves to Policy Director. Every user who wants to participate in the Policy Director secure domain must have an account registered with LDAP.

Using the users management panel

The security administrator for the secure domain uses the Management Console to create user accounts:

1. Log in to the Management Console as an administrator (for example, `cell_admin`).
2. Click the **Users** task tab.

   The Users management task panel appears.

Using action buttons for users management tasks

You use the **Users** action buttons to perform Users management operations. The following table describes the tasks that each action button performs:

Action Button	Description
New	Creates a new user account for the secure domain.
Get	Retrieves information about a specifically designated user and populates the detailed view windows.
Save	Saves this user account. The new entry appears in the appropriate list window.
Delete	Removes the selected user from the registry database.

Using user detail fields

The following table describes the fields in the **User Detail** view of the Management Console:

Field	Description
User ID	The primary name assigned to a user of the secure domain. The User ID is a key field that is used when a query is made to the registry.
Description	The informational text string that describes the user. The Description is only an optional data field and is not used by the registry.
Account Valid	This check box allows you to control the ability of a user to participate (or not participate) in the secure domain. When the box is cleared, the account is no longer valid. The account information still remains in the registry, however.
Password Valid	This check box allows you to force a password change the next time the user logs in to the secure domain. When the box is cleared, the user is informed that his password has expired.
GSO User	This check box indicates the user has GSO capabilities.
LDAP	For LDAP: • For cn=, type in a common name, such as `Diana Lucas` • For sn= , type in a surname, such as `Lucas` • For dn=, type in a distinguished name, such as `cn=Diana Lucas,o=IBM,ou=Austin,c=US`

Adding a new user account

To add a new user account:

1. Click the **New** action button.

   Blank entry fields appear in the User Detail view.
2. Type the appropriate data in the fields. See “Using user detail fields” for complete information about these fields.

   You can drag and drop a **group** icon from the Groups tree view to populate the Member of Groups window—or use the selection arrows.
3. Add the required LDAP registry information if LDAP is your default registry.
4. Click **Save**.

Changing account properties

To change the properties of an existing account:

1. Select the user from the User ID list view.

   The User Detail area populates with current data.
2. In the User Detail view, click the field you want to change.
3. Enter the new data.
4. Click **Save**.

Removing a user account

To remove a user account:

1. Select the user from the User ID list view.
2. Click **Delete**.

Creating multiple administrative accounts

An administrative user must be in the following groups. When in these groups, the administrative user has the authority to add, change, or remove users, groups, and organizations in the secure domain:

- acct-admin
- subsys/dce/sec-admin
- subsys/dce/cds-admin

When you initially create a secure domain, the only account that contains this group combination is cell_admin.

After a secure domain reaches a certain size, it becomes very problematic for one administrator to manage the growing number of tasks. Managing a large secure domain requires the delegation of administrative responsibility.

You can create additional administrative accounts with the same ability to operate the Management Console by assigning them membership in the three groups that are listed. Planning and organizing this delegation of authority must coincide with the creation of these accounts.

Importing information from other sources

You can populate the registry with user data and group data from other sources.

Chapter 6. Managing GSO resources, resource groups, and resource credentials

Policy Director supports a more flexible single sign-on solution by integrating with IBM Global Sign-On (GSO) technology. You achieve integration by creating Policy Director smart junctions. For complete information about smart junctions, see “Chapter 15. WebSEAL: smart junction administration” on page 185.

When WebSEAL receives a request for a resource that is located on the junctioned server, WebSEAL uses GSO to retrieve the appropriate authentication information. GSO contains a database of mappings for each registered user that provides user names and passwords for specific resources and applications. Policy Director stores the GSO data directly in the IBM SecureWay Directory (LDAP).

The combination of WebSEAL and GSO provides a complete single sign-on Web solution with the extra benefits of encrypted data, high availability, and scalability. Refer to “Integrating GSO and WebSEAL single sign-on” on page 207 for further information.

This chapter includes:

Understanding GSO resources and GSO resource groups

GSO contains a specific *resource credential* for the user that maps a GSO resource to a specific user identification (user name) and password combination. The resource credential provides this username and password for a GSO user-specific resource, such as a Web server or a group of Web servers.

GSO provides authentication information to WebSEAL. When a user wants to run an application resource, WebSEAL asks GSO for the user’s authentication information. GSO maintains a complete database of authentication information in the form of resources mapped to specific authentication information. The mapping of application resources to username and password combinations is known as *GSO resource credentials*. You can create GSO resource credentials for registered users only.

Note: The GSO resource, or GSO resource group, must exist before you can apply the GSO resource credential to it.

Managing GSO resources

A *GSO resource* is a Web server. You can name the Web resource to identify it.

Using the GSO resource management panel

GSO resources requesting access to services and objects in the secure domain must authenticate themselves to Policy Director. Every GSO resource that wants to participate in the Policy Director secure domain must have an account registered with LDAP.

Using action buttons for GSO resource management tasks

You use the **GSO Resources** action buttons to perform GSO resource management operations. The following table describes the tasks that each action button performs:

Action Button	Description
New	Creates a new GSO resource account for the secure domain.
Get	Retrieves information about a specifically designated GSO resource and populates the detailed view windows.
Save	Saves this GSO resource account. The new entry appears in the appropriate list window.
Delete	Removes the selected GSO resource from the registry database.

Using GSO resource detail fields

The following table describes the fields in the **Resource Detail** view of the Management Console:

Field	Description
Resource Name	The name assigned to a GSO resource of the secure domain. Resource Name is a key field that is used when a query is made to the registry.
Description	The informational text string that describes the resource. The Description is only an optional data field and is not used by the registry.

Adding a new GSO resource

To add a new GSO resource:

1. Click the **New** action button.

   Blank entry fields appear in the Resource Detail view.

2. Type the appropriate data in the fields. See “Using GSO resource detail fields” for complete information about these fields.

   You can drag and drop a **Resource Group** icon from the Resource Groups tree view to populate the Member of GSO Resource Groups window—or use the selection arrows.

3. Click the **Save** action button.

Creating a resource credential for the GSO resource

After you have created the resource definition, you can create a resource credential for a user:

To create a resource credential:

1. Select the **Users** tab.
2. Select the user for which to create the resource credential by highlighting the user's name.
3. In the User Detail GSO Resources view, select the **Resources** tab to list the resources that are currently available.
4. Select the resource for which the credential applies.
5. Drag the selected resource to the resource pane of the User Detail view.

   The sign-on ID and password values for the new resource credential default to the same values as the user's account.
6. The sign-on ID and password can be changed to the proper values for the resource credential for this user by clicking on either the **Sign-on ID** or **Password** button and filling in the proper value.
7. Click the **Save** action button.

Changing GSO resource information

To change the properties of an existing account:

1. Select the GSO resource from the Resources list view.

   The Resource Detail area populates with current data.
2. In the Resource Detail view, click the field you want to change.
3. Change the existing data or enter the new data.
4. Click the **Save** action button.

Removing a GSO resource

To remove a user account:

1. Select the GSO resource from the Resources list view.
2. Click **Delete**.

Managing GSO resource groups

A *resource group* refers to a group of Web servers, where all the servers in the group have the same sets of user IDs (userids) and passwords.

Using the GSO resource group management panel

The security administrator for the secure domain uses the Management Console to create GSO resource groups:

1. Log in to the Management Console as an administrator, such as `cell_admin`.
2. Click the **GSO Resource Groups** task tab.

   The GSO Resource Groups management task panel appears.

Using action buttons for GSO resource group management tasks

You use the GSO Resource Groups action buttons to perform resource groups management operations. The following table describes the tasks that each action button performs:

Action Button	Description
New	Creates a new GSO resource group entry for the secure domain.
Get	Retrieves information about a specifically designated GSO resource group and populates the detailed view windows.
Save	Saves this GSO resource group entry. The new entry appears in the appropriate list window.
Delete	Removes the selected GSO resource group from the registry database.

Using GSO resource group detail fields

The following table describes the fields in the **Resource Group Detail** view of the Management Console:

Field	Description
Resource Group Name	The primary name assigned to a GSO resource group of the secure domain. The GSO resource group name is a key field that is used when a query is made to the registry.
Description	The informational text string that describes the GSO resource group. The description is only an optional data field and is not used by the registry.

Adding a new GSO resource group

To create a new GSO resource group:

1. Click the **New** action button.

   Activate entry fields for **Resource Group Name** and **Description** in the Resource Group Detail area.
2. Type the new resource group name, and optionally type a description of the resource group in the **Description** field.
3. You can add GSO resources to the new GSO resource group by dragging GSO resource icons from the GSO Resources list view to the GSO resource window of the Resource Group Detail view.

   Also, you can use selection arrows to move resources in and out of the GSO Resources panel.
4. Click the **Save** action button.

   A new GSO group entry appears in the Resource Groups list view.

Creating a GSO resource credential

You can create a single resource credential for all the resources in the resource group. Policy Director uses a single resource credential for a resource group instead of a resource credential for each resource in the resource group. First, you must create the resource group before you can create the GSO resource credential.

To create a GSO resource credential for a user after you have created the resource group definition:

1. Select the **Users** tab and select the user for which to create the resource credential.
2. In the User Detail GSO Resource Groups view, select the **Resource Groups** tab to list the resource groups currently available.
3. Select the resource group for which the credential applies.
4. Drag the selected resource group to the Resource Group pane of the User Detail view.

   The sign-on ID and Password values for the new resource credential default to the same values as the user's account.
5. The sign-on ID and Password can be changed to the proper values for the resource credential for this user by clicking on either the **Sign-on ID** or **Password** button and filling in the proper value.
6. Click the **Save** action button.

Changing GSO resource group information

To change the resource membership or name of an existing resource group:

1. Select a resource group from the Resource Groups list view.

   The Resource Group Detail view populates with current information about the resource group.
2. In the Resource Group Detail area, select the field to change (either **Resource Group Name** or **Description**). Type the new values.
3. You can also add new resources to the resource group or delete resources from the group.

   From the Resource Groups column, double-click the icon of a resource group. Then, drag and drop the new resource to the GSO Resources window in the Resource Group Detail view. You can also use the selection arrows.
4. Click the **Save** action button.

Removing a GSO resource group

To remove a GSO resource group:

1. From the Resource Groups list view, select the name of the GSO resource group you want to delete.

   The list of GSO resource group members appears in the Resource Group Detail view.
2. Select each GSO resource, one-at-a-time, and remove it from the GSO resource group.
3. From the Resource Groups list view, select the GSO resource group.
4. Click **Delete** to completely remove the GSO resource group entry.

Migrating GSO data

If you have GSO data from IBM SecureWay Global Sign-on Version 2.0.200 or from previous releases of IBM Global Sign-On, you will need to migrate your GSO data so that it can be used by this version of Policy Director.

The most up-to-date information and tools (for example, migration tools) can be found at the IBM SecureWay Policy Director Web site:

`http://www.ibm.com/software/security/policy/library`

Changing the GSO resource credential password

A user can update the stored GSO password for a GSO resource or a GSO resource group using the Policy Director Web-based password tool, chpwd.exe. Before you can use this tool, a resource credential must have been created. Use this tool after first changing the password on the resource.

This file can be found at:

UNIX: `/opt/intraverse/www/docs/cgi-bin/chpwd`

Windows: `c:\Program Files\www\docs\cgi-bin\chpwd.exe`

To change the GSO resource credential password using the Web tool:

1. Open an instance of your secure browser.
2. Enter the following URL location:

   `https: //`*`webseal server`*`/cgi-bin/chpwd.exe`

   Where *webseal server* is the name assigned to your WebSEAL server. For Windows, you must type the .exe extension as part of the URL designation.
3. Click on the name of the resource in the Resource Name column to select it.
4. Type your user name in the **User ID** field.
5. Type the password you want to change it to in the **New Password** field. Then, confirm it by typing it again in the **Confirm New Password** field.
6. Click **Update**.

Chapter 7. Understanding access control

You can protect resources in a secure domain. You protect resources by defining special rules and attaching these templates to object representations of these resources. These special rules are known as *policy templates*. Policy Director recognizes and uses a type of policy template that is known as the *access control list* (ACL). Use ACLs to stamp an organization's security policy onto the resources that belong to the secure domain.

This chapter includes:

Protected object namespace

The Policy Director security model depends on rules, or permissions, to protect resources in the secure domain. A specific set of permissions is known as a *policy template*.

When attached to a resource, a policy template effectively applies the corporate security policy to the resource. To accomplish this security model, Policy Director uses a logical object representation of the secure domain's physical resources inventory.

To provide protection of the actual physical resource, attach policy templates to the logical objects in the namespace. The Policy Director Authorization Service makes authorization decisions by comparing user credentials that are obtained during authentication with the permissions that are defined in these templates.

The Policy Director *protected object namespace* is the logical and hierarchical portrayal of resources that belong to a secure domain. Objects that appear in the hierarchical namespace represent actual physical network resources.

System Resource
: The actual physical file, network service, or application.

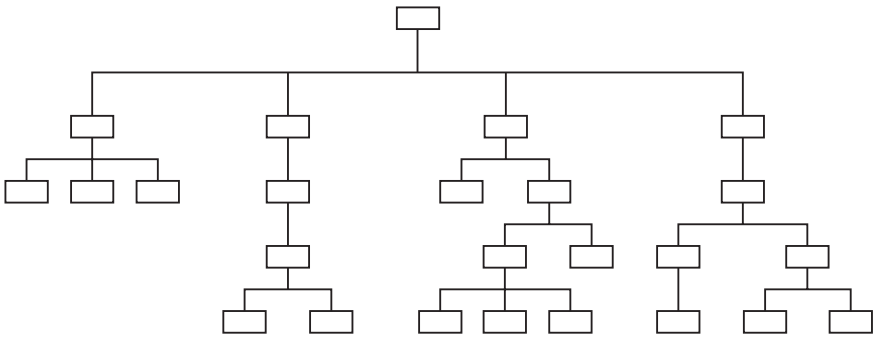
Protected Object
: The logical representation of an actual system resource that is used by the Policy Director Authorization Service, the Management Console, and other Policy Director management utilities.

The protected object namespace uses two types of objects:

Container objects
: Container objects are structural designations that allow you to organize the namespace hierarchically into distinct functional regions. Container objects contain resource objects.

Resource objects

Resource objects are the representations of actual network resources (such as services, files, and programs) in your secure domain.



Protected object namespace hierarchy

The structural top of the protected object namespace is the **root** container object. On the Policy Director Management Console, the symbol for root is the forward slash (/).

The following namespace categories follow the root object:

Web objects (/WebSEAL container)

The WebSEAL container object is the logical Web space root of the secure domain. Policy Director authorizes all HTTP operations against some object in this subtree.

Web objects represent anything that a URL can address, including static Web pages and dynamic URLs. Web-to-application gateways convert these static Web pages or dynamic URLs to database queries or some other type of application call.

Network application objects (/NetSEAL container)

The NetSEAL container object is the root of the logical space that contains NetSEAL protected services in the secure domain. These objects represent TCP-based applications (such as Telnet and FTP) that map to the TCP network addresses (ports). The application uses these ports.

Policy Director management objects (/Management container)

The Management container object is the root of the logical space that controls all Policy Director management operations. Management objects represent the services that are required to define users and set security policy. Perform these tasks by using the Policy Director Management Console, or the **ivadmin** utility.

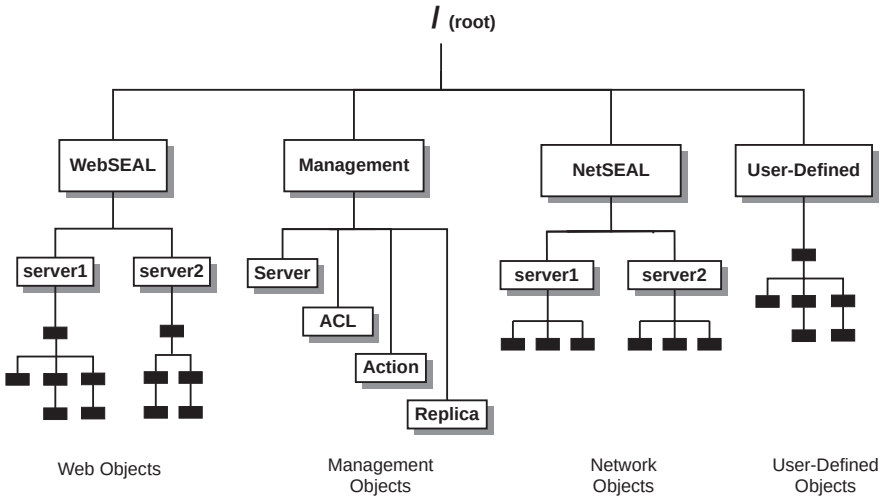
Subdivisions of this region include:

- Server management tasks (/Server)
- Security policy tasks (/ACL)
- Third-party authorization control (/Action)
- Authorization database replication control (/Replica)

Policy Director supports delegation of management activities and can restrict an administrator's ability to set security policy to a subset of the namespace.

User-defined objects

These objects represent tasks or network resources that are protected by third-party applications that use the Policy Director Authorization Service, which in turn use the Policy Director Authorization API.



Third-party application namespaces

Policy Director can provide authorization services to any application object that is defined by the protected object namespace. Applications that are part of the Policy Director family include WebSEAL (for Web applications) and NetSEAL (for TCP-based applications).

Policy Director and third-party applications make calls to the Policy Director Authorization Service through the Policy Director Authorization API. To integrate a third-party application with the Policy Director Authorization Service, use these two steps:

1. Describe the third-party application's namespace.
2. Apply permissions on any namespace objects that require protection.

Optional user-defined object containers are regions of the protected object namespace where you can build third-party application namespaces.

You must define the root (*junction point*) at which this third-party namespace starts in the protected object namespace. See "Defining third-party application namespaces" on page 131 for more detailed information.

You then use the Management Console or the **ivadmin** utility to create, attach, and delete ACLs in the objects in this new namespace.

Access control lists

An *access control list* (ACL) is the policy template type that is used by Policy Director to provide protection for resources in the secure domain.

An ACL is a set of rules, or permissions, that specify the conditions necessary to perform an operation on a protected resource. ACLs identify the operations that are permitted on protected resources and list the identities (users, groups, or both) who can perform those operations, such as:

- Definitions of user identities and group identities in the security registry.
- Definitions of the protected object namespace and the policy templates in the authorization policy database.

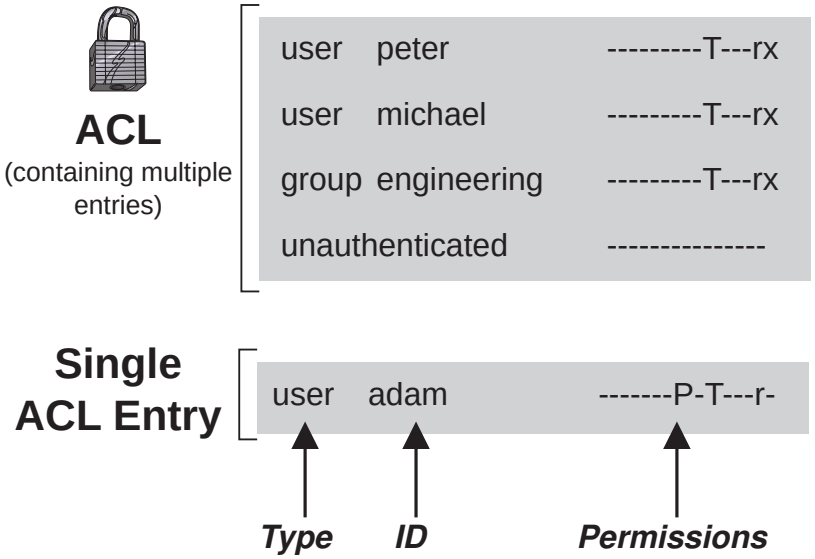
As a policy template, each Policy Director ACL has a unique name, or *label*, that reflects the security policy it represents. Then, you apply ACLs to the object representations of resources in the protected object namespace.

An ACL consists of one or more entries that include user designations and group designations and their specific permissions.

ACL entries

An ACL consists of one or more entries describing:

- The UUIDs of users and groups whose access to the object is explicitly controlled
- The specific operations that are permitted to each user, group, or role
- The specific operations that are permitted to the special “any” and “unauthenticated” user categories



A *user*, or principal, represents any identity that is authenticated by the Policy Director Security server. Typically, users represent network users or application servers.

A *group* is a collection of one or more users. A network administrator can use group ACL entries to assign the same permissions to multiple users. New users gain access to objects by becoming members of appropriate groups. This eliminates the need to set new ACL entries for every new user. Groups can represent organizational divisions or departments within a secure domain. Groups are also useful in defining roles or functional associations.

Collectively, users and groups are *entities*.

You use the Policy Director Management Console (**ACLs** management tab) to create, change, and delete ACL entries.

ACLs as policy templates

The Management Console allows you to:

- Create a specific ACL.
- Save it with a label.
- Apply it as a security policy template to objects in the namespace.

The ACL becomes a single source template, like a formula or recipe. The ACL contains the specific entries that provide the correct level of protection for any objects that are associated with it.

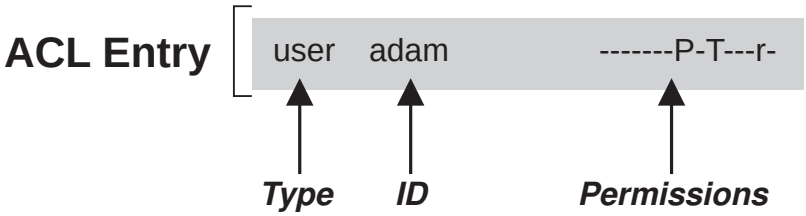
An example ACL list might include:

```
ACL Name
    default-management
    default-netseal
    default-replica
    default-root
    default-webseal
```

An ACL template provides the same single source-referred qualities as does a paragraph formatting style in a word-processing document. If the security policy requirements change, you edit only the single ACL. Policy Director instantly puts into effect the new security definition for all objects to which that ACL is attached.

ACL entry syntax

An ACL entry contains either two or three attributes, depending on the ACL entry type and appears in the following format:



Type The entity category (user or group) for which the ACL was created.

ID (Identity) The unique identifier (name) of the entity.

Any-authenticated ACL entry types, or unauthenticated ACL entry types, do not require the ID attribute.

Permissions The set of operations that are permitted on the object by this user or group.

Most permissions dictate the client's ability to perform a specific operation on the resource. Several permissions impose certain conditions whenever an action on a resource is allowed. Examples of the latter can include forcing data encryption, protecting data integrity, writing a report record to the auditing service, or requiring some external authorization condition.

In this example, Adam (type=user, ID=adam) has permission to read (view) the object that is associated with the ACL that contains this entry. The read (r) permission allows the read operation. The encryption (P) permission mandates that the communication channels use data encryption. The traverse (T) permission enforces the traverse attribute.

Type attribute

An ACL entry type designates the entity for the ACL entry. There are four ACL entry types:

Type	Description
user	Sets permissions for a specific user in the secure domain. The user entry type requires an account name (ID). The entry format is: `user ID permissions`. For example: `user greg ----------r-`
group	Sets permissions for members of a specific group in the secure domain. The group entry type requires a group name (ID). The entry format is: `group ID permissions`. For example: `group engineering ----------r-`
any-authenticated	Sets permissions for all authenticated users. No ID designation is required. The entry format is: `any-authenticated ----------r-`

unauthenticated	Sets permissions for those users who have not been authenticated by the Security server. No ID designation is required. The entry format is: `unauthenticated permissions`. For example:
	`unauthenticated ---------T---r-`
	
	This ACL entry is a mask (a bit-wise "and" operation) against the any-authenticated ACL entry to determine the permission set. For example, the following unauthenticated ACL entry:
	`unauthenticated -------------r-`
	
	masked against this any-authenticated ACL entry:
	`any-authenticated ---------T---r-`
	
	results in these permissions:
	`-------------r- (read only).`

ID attribute

The ACL ID is the *unique identifier*, or name, for user entry or group entry types. IDs must represent valid users and groups that are created for the secure domain and stored in the registry database.

Following are examples of unique identifiers:

```
user michael
user greg
group engineering
group documentation
group accounting
```

Note: Do not use the ID attribute for the any-authenticated and unauthenticated ACL entry types.

Permission attributes

Each ACL entry contains a set of *permissions* describing:

- The specific operations that are permitted on the object by the user or group
- Any restrictions on the type of access to the object, such as:
  - Mandatory use of data privacy or integrity on the communications channel
  - Audited access
  - External (third-party) authorization requirements

ACLs control protected resources by controlling:

- A user's ability to perform operations on protected objects
- An administrator's ability to change access control rules on the object and any subobjects
- The Policy Director server's ability to delegate user's credentials

Permission sequence

ACL permissions are *context-sensitive* permissions. Context-sensitive permissions mean that the behavior of certain permissions varies. The behavior varies according to the region of the protected object space in which they are applied. For example, the m permission has a different meaning on a WebSEAL object than on a Management object.

The seventeen standard permissions are grouped into four categories and appear in the following order when used in an ACL entry:

Base	Generic	NetSEAL	WebSEAL
a A b c g I P T	d m s v	C p	l r x

The ACL Definition/ACL Entry window displays the list of permission that you can select from. Select the check boxes next to these permissions to choose them:

Base
  (a) Attach
  (A) Audit
  (b) Browse
  (c) Control
  (g) Delegation
  (I) Integrity
  (P) Privacy
  (T) Traverse
Generic
  (d) Delete
  (m) Modify
  (s) Server Admin
  (v) View
NetSEAL
  (C) Connect
  (p) Proxy
WebSEAL
  (l) List Directory
  (r) Read
  (x) Execute

Regions of the namespace

Container objects represent specific regions of the protected object namespace and serve these important security functions:

1. You can use the container object's ACL to define high level policy for all subobjects within the region when no other explicit ACLs are applied.
2. You can use the container object's ACL to define high-level policy for all subobjects within the region. You can define a high-level policy when no other explicit ACLs are applied.
3. You can quickly deny access to all objects in a region by removing the traverse permission from the container object's ACL.

Traverse permission

The traverse permission is a generic permission that applies throughout the protected object namespace:

Traverse Permission:	Access	Description
T	traverse	Allows the requester to hierarchically pass through the object on the way to the requested object. It does not allow any other type of access to the object. Traverse is also required on the requested object itself.

Access conditions

Access conditions are generic permissions that apply throughout the protected object namespace:

Access Conditions for All Protected Object:	Access	Description
A	audit	Causes the Policy Director server to write an audit record to the audit service whenever the object is accessed. Audits all access attempts, including authorization failure
I	integrity	To access this object, data integrity protection is required between the client and the Policy Director server
P	privacy	To access this object, data encryption is required between the client and the Policy Director server

Control permission

The control permission is a powerful permission that gives you ownership of the ACL. Control allows you to change the entries in the ACL. Control means that you have the power to create entries, delete entries, grant permissions, and take away permissions.

Control Permission:	Access	Description
c	control	Ownership of the ACL template; allowed to create, delete and modify entries for this ACL

The administrator can delete this ACL from the list of ACL templates. Before deletion, the administrator must have an entry in that ACL. Also, the administrator must have the control permission set in that entry.

The control permission allows you to grant administration powers to another user. For example, you can grant the ability to attach an ACL to objects. You attach an ACL to objects using the attach (a) permission.

You must use the control (c) permission with great care because of its powerful ownership properties.

Root container object

The following security considerations apply for the *root* (/) container object:

- The root object begins the chain of ACL inheritance for the entire protected object namespace.
- If you do not apply any other explicit ACLs, the root object defines (through inheritance) the security policy for the entire namespace.
- You must use the traverse (T) permission for access to any object below root.

WebSEAL namespace

The following security considerations apply for the /WebSEAL container object:

- The WebSEAL object begins the chain of ACL inheritance for the WebSEAL region of the namespace.
- If you do not apply any other explicit ACLs, this object defines (through inheritance) the security policy for the entire Web space.
- You must use the traverse (T) permission for access to this object and any object below this point.

/WebSEAL/host

This subtree contains the Web space of a particular Policy Director WebSEAL server. The following security considerations apply for this object:

- You must use the traverse (T) permission for access to any object below this point.
- If you do not apply any other explicit ACLs, this object defines (through inheritance) the security policy for the entire namespace on this machine.

/WebSEAL/host/file

This subtree is the resource object that is checked for HTTP access. The permissions checked depend on the requested operation.

WebSEAL Permissions

The following table describes the permissions applicable for the WebSEAL region of the namespace:

WebSEAL Namespace Permissions:	Access	Description
r	read	View the HTTP object
x	execute	Run the CGI program
d	delete	Remove the HTTP object
m	modify	PUT an HTTP object (Place - publish - an HTTP object in the WebSEAL namespace)
l	list	Generate an HTTP directory auto-list
g	delegation	Assigns trust to a WebSEAL server to act on behalf of a client, and pass that request to a junctioned WebSEAL server

NetSEAL namespace

The following security considerations apply for the /NetSEAL object:

- The NetSEAL object begins the chain of ACL inheritance for the NetSEAL region of the namespace.
- If you do not apply any other explicit ACLs, this object defines through inheritance the security policy for all NetSEAL protected services in the namespace.
- You must use the traverse (T) permission for access to this object and any object below this point.

/NetSEAL/host

This subtree contains all NetSEAL protected services on the particular server machine. The following security considerations apply for this object:

- You must use the traverse (T) permission for access to any resource below this point.
- If you do not apply any other explicit ACLs, this object defines (through inheritance) the security policy for all NetSEAL protected services on this machine.

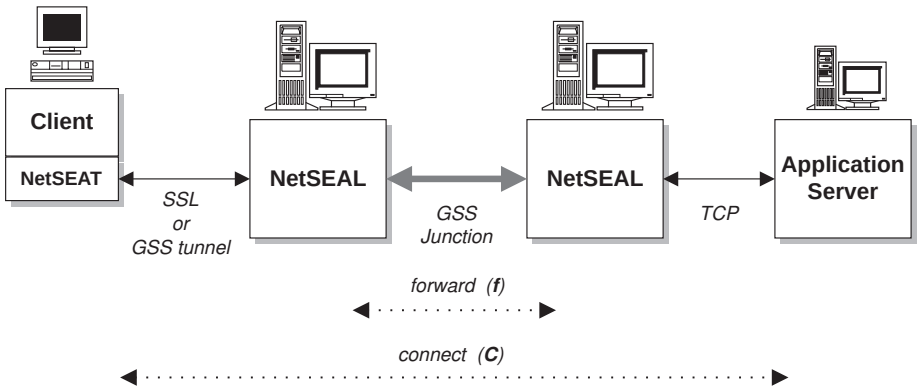
/NetSEAL/host/service

This subtree is the object that is checked for access to the protected service it represents. The permissions checked depend on the requested operation.

NetSEAL permissions

The following table describes the permissions applicable for the NetSEAL region of the namespace:

NetSEAL Protected Object Permissions:	Access	Description
C	connect	Connect to a NetSEAL server
f	forward	Permit outgoing connection across a NetSEAL junction; traverse the junction.



Management namespace

The following security considerations apply for the /Management object:

- The Management object begins the chain of ACL inheritance for the Management region of the namespace.
- If you do not apply any other explicit ACLs, this object defines (through inheritance) the security policy for the entire Management namespace.
- You must have the traverse (T) permission access for this object.

/Management/server

The /Management/server container object of the Policy Director protected object namespace allows administrators to perform server management tasks (when appropriate permissions are set).

Use server management controls to determine if a user has permission to create, change, or delete a server definition. Server definitions contain information that allows other Policy Director servers, particularly the Management server (ivmgrd), to locate and communicate with that server.

You can create a server definition for a particular Security Manager (secmgrd) or Authorization Server (ivacld) as part of the installation process. Policy Director also deletes the definition for a server when you uninstall the server.

The creation and deletion of the definition happen automatically; the installation administrator does not have to perform any special steps to create the definition. However, the administrator must have modify (m) permission on the /Management/Server object in order to create the definition during installation.

In addition, the administrator must have delete (d) permission on the /Management/Server object in order to delete the definition during uninstallation.

Other operations that can be performed on a server definition include allowing the user to:

- View the definitions through the **ivadmin** utility. The user must be granted view (v) permission on the server object.
- Perform server management operations, such as start, stop, suspend, resume servers, or delete logs. The user must be granted server administration (s) permission on the server object.
- Change parts of the definition by using the **ivadmin server modify** command. The user must be granted modify (m) permission on the server object.

Server Management Permissions:	Access	Description
s	server	Performs server administration tasks (such as start, stop, suspend, resume)
v	view	Lists servers
m	modify	Creates a new server definition
d	delete	Deletes a server definition

/Management/ACL

This object allows administration users to perform high-level ACL management tasks that can affect the security policy for the secure domain.

ACL Management Permissions:	Access	Description
b	browse	Views the contents of the namespace below the object
a	attach	Attaches ACL templates to objects; remove ACL templates from objects
m	modify	Creates a new ACL template
d	delete	Deletes an existing ACL template. The ACL must contain an entry, with the control (c) permission, for this same user.
v	view	Lists or views an ACL

You must define ACL administrators in the default ACL management object. The administrator's ACL entry can contain any of the above permissions. These permissions give the administrator powers to create new ACL templates, attach ACLs to objects, and delete ACL templates.

An ACL administrator cannot change 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 change its entries.

Note that the creator of a new ACL template becomes the first entry in that ACL, with the abcT permissions that are set by default.

For example, if `cell_admin` is an entry in the default-management ACL, with (m) permission, cell_admin can create a new ACL template. User cell_admin becomes the first entry in the new ACL, with abcT permissions. The control (c) permission gives cell_admin ownership of the ACL and allows cell_admin to change the ACL. User cell_admin could then grant administration permissions to other user entries in that ACL.

Ownership of the default-management ACL itself is given to user `cell-admin` and group `iv-admin` by default.

/Management/action

This object allows administration users to perform ACL management tasks in a third-party namespace. Action tasks and associated permissions include:

Action Management Permissions (Third-Party Authorization):	Access	Description
m	modify	Creates a new action
d	delete	Deletes an existing action

Policy Director provides authorization services to applications. Applications that are part of the Policy Director family include WebSEAL (for Web applications) and NetSEAL (for TCP-based applications).

Third-party applications can make calls to the Policy Director Authorization Service through the Policy Director Authorization API. Two necessary steps required to integrate a third-party application with the Policy Director Authorization Service include:

- Define the application's namespace.
- Apply permissions on objects (resources) that need protection.

The administrator of a third-party application namespace can use the **ivadmin** utility to define new permissions and actions. The administrator must have Management and Action permissions in order to create and delete these permissions and actions.

/Management/replica

The /Management/Replica container object of the Policy Director protected object namespace controls the replication of the authorization database. High-level controls on this object affect the operation of the Management server and the Security Managers in the secure domain.

Replica management controls are used to determine what processes are allowed to read or update the primary authorization policy database in order for replication to take place properly.

Controls and associated permissions include:

Replica Management Permissions:	Access	Description
v	view	Read the primary authorization database
m	modify	Authorize modification of the replica database or databases

All Policy Director servers maintain a local replica of the authorization database. Policy Director servers include all Security Managers (secmgrd) and Policy Director Authorization servers (ivacld). All Policy Director servers have view (v) permission on the /Management/Replica object.

The replication process allows these processes to view and access entries out of the primary authorization policy database. The Policy Director installation automatically grants read (r) permission to any server that requires access to the authorization policy database.

Policy Director currently does not use the modify (m) permission. Currently, you change the primary policy authorization database through the Management Console or the **ivadmin** utility. These tools are subject to other finer-grained checks.

Guidelines for a secure namespace

These guidelines provide information that helps to make the namespace secure:

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

  Inherited ACLs simplify the maintenance of your tree because they reduce the number of ACLs you must maintain. This lower maintenance reduces the risk of an error which could compromise your network.
- 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 an explicit ACL at the root of the subtree.
- Create a core set of ACL templates and re-use these ACLs wherever necessary.

  Because an ACL template is a single source definition, any modifications to the template affect all objects that are associated with this ACL.
- Control user access through the use of groups.

  It is possible for an ACL to consist of only group entries. Adding users to or removing users from these groups can control access to an object by individual users efficiently.

Standard administration ACL templates

The following default administration ACL templates are suggested starting points for securing specific regions of the secure domain.

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

Note the users and groups in each ACL that contain the control (c) permission. Users, groups, or both with the control permission *own* the ACL and have the power to change the ACL entries.

Root

Core entries for the default root ACL (default-root) include:

```
user cell_admin                 abcT
group iv-admin                  abcT
any-authenticated               T
unauthenticated                 T
```

The root ACL is very basic. Everyone can traverse the namespace, but cannot perform any other actions. Typically, you would not need to change this. However, one useful function of the root ACL is to quickly deny access to the entire namespace for an individual user or group.

Consider the following entry in the root ACL:

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

The consequence of this entry (no permissions) is that `user john` cannot even traverse the root container object. This user cannot gain access at all to the protected object space, regardless of any permissions that are granted lower down in the tree.

You can apply this same approach to the WebSEAL and NetSEAL object spaces. For example, you can take away the traverse (T) permission from a particular user at the /WebSEAL container objects. When taken away, that user cannot gain entry to the WebSEAL namespace at all. The user cannot gain entry, regardless of any permissions that are granted on objects within those regions.

WebSEAL object space

Core entries for the WebSEAL ACL (default-webseal) include:

```
user cell_admin                 abcTdmlrx
group iv-admin                  abcTdmlrx
group webseal-servers           gTdmlrx
group ivmgrd-servers            Tl
```

At installation, this default ACL is attached to the /WebSEAL container object in the namespace.

The group, `webseal-servers`, contains an entry for each WebSEAL server in the secure domain. The default permissions allow the servers to respond to browser requests.

The group, `ivmgrd-servers`, contains only one entry that represents the Management server. Most management requests made by the Management Console are started using the Management server to the target WebSEAL server. Therefore, the Management server must have permission to perform the request at the target server.

The traverse permission allows expansion of the Web space as represented in the Management Console. The list permission allows the Management Console to display the contents of the Web space.

NetSEAL object space

Core entries for the NetSEAL ACL (default-netseal) include:

```
user cell_admin                 abcTC
group iv-admin                  abcTC
```

At installation, this ACL is attached to the /NetSEAL container object in the namespace. You must grant control (c) permission for access to a protected service.

Management object space

Core entries for the Management ACL (default-management) include:

```
user cell_admin                 abcTdmsv
group iv-admin                  abcTdmsv
group ivmgrd-servers            Ts
any-authenticated               Tv
unauthenticated                 Tv
```

At installation, this ACL is attached to the /Management container object in the namespace.

Replica management object

Core entries for the Replica management ACL (default-replica) include:

```
group secmgrd-servers           dmv
group ivacld-servers            dmv
group ivmgrd-servers            m
group iv-admin                  abcTv
user cell_admin                 abcTv
```

Evaluation of an ACL

Policy Director follows a specific evaluation process to determine the permissions that are granted to a particular user by an ACL.

Evaluation of authenticated requests

Policy Director evaluates an authenticated user in the following order:

1. Match the user ID with the ACL's user entries. The permissions granted are those in the matching entry.

   ```
   Successful: evaluation stops here. Unsuccessful: continue to the next
   step.
   ```

2. Determine the group or groups to which the user belongs and match with the ACL's group entries:

   If more than one group entry is matched, the resulting permissions are a logical "or" (most permissive) of the permissions granted by each matching entry.

   ```
   Successful: evaluation stops here.
   Unsuccessful: continue to the next step.
   ```

3. Grant the permissions of the any-authenticated entry (if it exists).

   ```
   Successful: evaluation stops here.
   Unsuccessful: continue to the next step.
   ```

4. An implicit any-authenticated entity exists when there is no any-authenticated ACL entry. This implicit entry grants no permissions.

   ```
   Successful: no permissions granted.
   End of evaluation process.
   ```

Evaluation of unauthenticated requests

Policy Director evaluates an unauthenticated user by granting the permissions from the ACL's unauthenticated entry.

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

Because unauthenticated depends on any-authenticated, it makes little sense for an ACL to contain unauthenticated without any-authenticated. If an ACL does contain unauthenticated without any-authenticated, the default response is to grant no permissions to unauthenticated.

Example ACL entries

You set permissions for specific users, groups, or both by specifying the appropriate ACL entry type. In the following example, the group `documentation` has full access privileges:

```
group documentation --bcg--TdmsvC-lrx
```

You can restrict access to other authenticated users in the secure domain (not belonging to the documentation group) by using the any-authenticated entry type:

```
any-authenticated -------T-------rx
```

You can further restrict access to the unauthenticated entry type for users who are not members of the secure domain:

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

Note: Without an unauthenticated ACL entry, unauthenticated users cannot access any secure documents within the Policy Director secure domain.

Sparse ACL model for ACL inheritance

To secure network resources in a protected object space, you must attach an ACL to each object.

You can assign an ACL to an object in one of two ways:

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

Adopting an inherited ACL scheme can greatly reduce the administration tasks for a secure domain. This section discusses the concepts of inherited, or sparse, ACLs.

Sparse ACL model overview

This principle is the base for the power of ACL inheritance: Any object without an explicitly attached ACL inherits the ACL of its nearest container object with an explicitly set ACL. In other words, all objects *without* explicitly attached ACLs inherit ACLs from container objects *with* explicitly attached ACLs. A particular chain of inheritance is broken when you attach an explicit ACL on an object.

ACL inheritance simplifies the task of setting and maintaining access controls on a large, protected, object space. In a typical object space, you only need to attach a few ACLs at key locations to secure the entire object space. A few ACLs at key locations mean that this is a *sparse* ACL model.

A typical Policy Director namespace begins with a single explicit ACL that is attached to the root container object. The root ACL must always exist and can never be removed. Normally, this is an ACL with very little restriction. All objects located in the namespace below inherit this ACL.

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

Default root ACL template

Policy Director checks inheritance by beginning with the root of the protected object space. If you do not explicitly set ACLs on any other objects in the tree, the entire tree inherits this root ACL.

There is always an explicit ACL template that is set at the root. An administrator can replace this ACL with another ACL that contains different entries and permission settings. But the root ACL can never be completely removed.

Policy Director explicitly sets the root ACL template during the initial Policy Director installation and configuration at the ACL Definition/ACL Entry window:

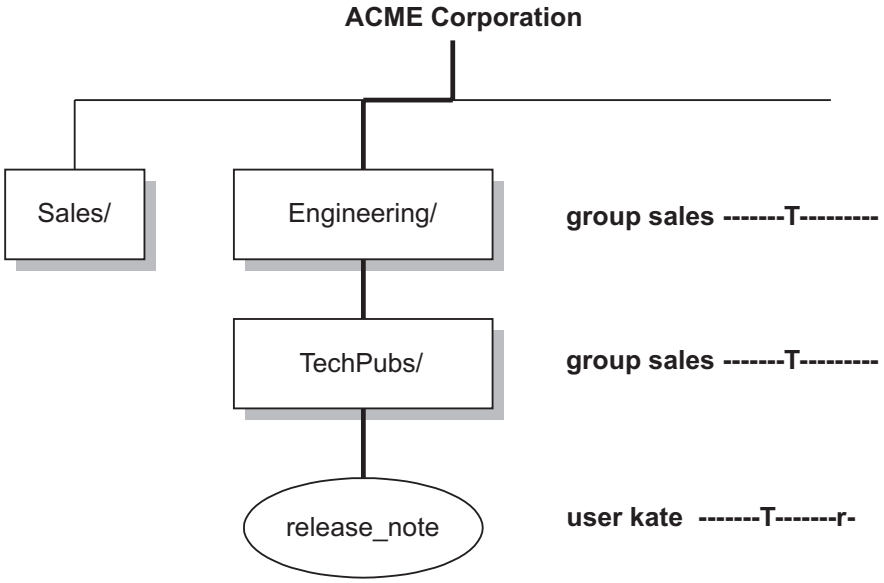
```
ACL Name     default-root
Description  Default Root ACL
```

Traverse permission

The traverse (T) permission specifies that the entity identified in the ACL entry has permission to pass through that object. The entry needs permission to pass through the object in order to gain access to an object below in the hierarchy. The traverse permission grants no other permissions for that object. You must provide the traverse permission on the requested object itself.

The following figure illustrates how the traverse permission works. Within the ACME Corporation, there is an Engineering directory, which also contains a TechPubs subdirectory. Kate (user kate) is a member of the Sales department and requires access to the /Engineering/TechPubs directory to review release note files.

The administrator places a group sales ACL entry, with the traverse (T) permission, on both the /Engineering and /TechPubs directories. Although user kate has no other permissions in these two directories, she can pass through these directories in order to access the release_note file. Because this file has both traverse (T) and read (r) permissions for user kate, she can view the file.



You can easily restrict access to the hierarchy below a given container object—without resetting individual permissions on these objects. Simply remove the traverse permission from the appropriate ACL entry. Removing traverse permission on a directory object protects all objects lower in the hierarchy, even if those objects contain other less restrictive ACLs.

For example, group sales must have the traverse (T) permission on the Engineering directory. If group sales did not, Kate could not access the release_note file, even though she has read (r) permission for the file.

Resolution of an access request

Inheritance begins with the root ACL and affects all objects in the namespace until it reaches an object with an explicit ACL. At this point, a new chain of inheritance begins.

Objects below an explicitly set ACL inherit the new access controls. If you delete an explicit ACL, access control for all objects reverts back to the nearest directory or container object with an explicitly set ACL.

When a user tries to access a secure object, Policy Director checks whether the user has the permissions to access the object. For example, a secure object might be a Web document. It does this by checking every object along the object hierarchy for the proper inherited or explicitly set permissions.

You can deny a user access to an object. Policy Director denies access when any directory object or container object in the hierarchy above does not include the traverse (T) permission for that user. Also, Policy Director denies access when the target object does not contain sufficient permissions to perform the requested operation.

In order to succeed an access check, the requester must have both:

1. Permission to traverse the path to the requested object.
2. Appropriate permissions on the requested object.

The following example illustrates the process of resolving whether a user can read (view) an object:

```
/acme/engineering/project_Y/current/report.html
```

Policy Director checks the:

1. Traverse permission on the explicitly set root ACL (/).
2. Traverse permission on any explicit ACLs attached to the directories: `acme`, `engineering`, `project_Y`, and `current`.
3. Read permission on the file itself (report.html).

Policy Director denies the user access when the user fails the access check at any of these points along the object hierarchy.

ACL templates applied to different object types

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

The reason for this behavior relates to the two features of Policy Director that are designed to make administration easier:

- ACL templates
- ACL inheritance

ACL templates allow you to attach the same ACL definition to multiple objects in the protected object namespace. The ACL definition consists of enough entries to meet the requirements of all objects to which the ACL will be applied. However, only a few of the entries affect each individual object.

In the ACL inheritance model, any object without an attached explicit ACL inherits the policy definitions. These inherited policy definitions come from the nearest ACL applied to an object above it in the hierarchy.

In summary, an ACL template has to describe all the necessary permissions for all object types to which it will be applied. The ACL templates do not describe just the object to which it is attached.

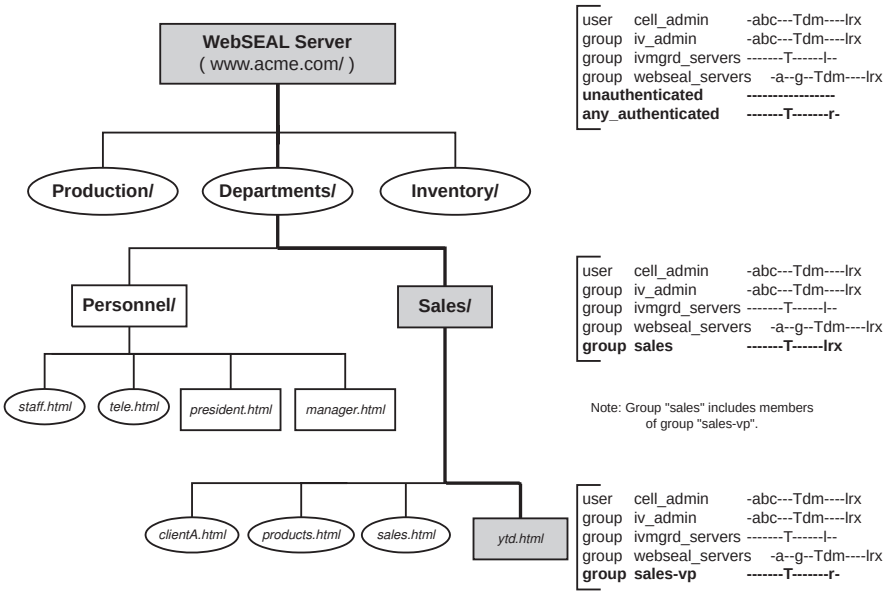
Example of ACL inheritance

The following figure illustrates the impact of an inherited and explicit ACL mixture in a corporate namespace.

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 their departmental subtree. Note that the ACL on this subtree no longer acknowledges the unauthenticated or any-authenticated entry types. The Year-to-Date sales file (ytd.html) has an explicit ACL. This explicit ACL grants read (r) permission to members of the `sales-vp` group; These `sales-vp` group members are also members of the `sales` group).

Note: This ACL scheme need not be changed with the addition or subtraction of users within the secure domain. New users are simply added to the appropriate group or groups. Likewise, users can be removed from those groups.



ACL management delegation

The distribution of administration responsibilities within a secure domain is called management delegation. The need for management delegation generally arises from the growing demands of a large site containing many distinct departmental or resource divisions.

Typically, a large object space can be organized into regions that represent these departments or divisions. Each distinct region of the domain can be better organized. Also, each distinct region can be maintained by a manager who is more familiar with the issues and needs of that branch.

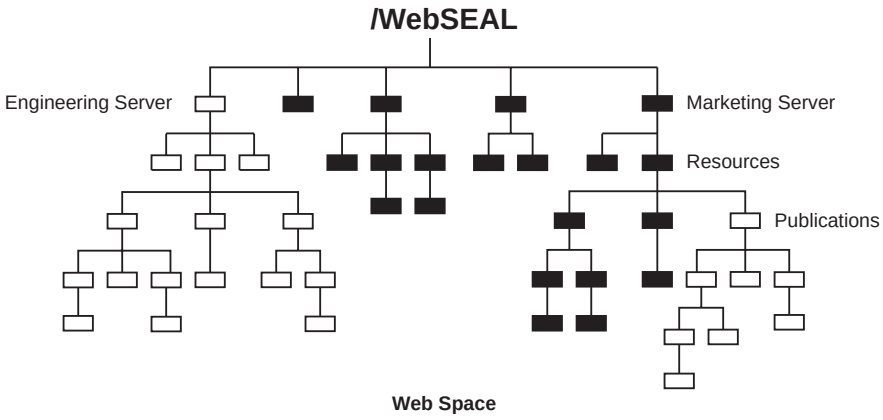
Each distinct region of the domain is usually better organized. The manager, who is more familiar with the issues and needs of that branch, maintains each distinct region.

In a Policy Director secure domain, the cell_admin account is initially the only account with administration permission. As cell_admin, you can create management accounts and assign to these accounts appropriate controls for specific regions of the object space.

Structuring the namespace for management delegation

Structure your object space to contain distinct regions, or branches, where submanagement responsibilities, specific to that branch, can be carried out.

In the example below, the Engineering and Publications regions of the object space require separate management control. Control of these regions begins with the root of each region and extends to all objects below.



Using default administration users and groups

Policy Director creates several important administration groups. By default, these users, groups, or both are given special permissions to control and manage all operations in the secure domain. This default security policy is defined by the ACLs that are created during installation.

The following sections detail the specific roles that are assigned to each of these users and groups at installation time. The administrator can customize these privileges at a later time to accommodate changing management policies.

user cell_admin

This user represents the administrator of the secure domain who is granted complete rights for all operations within the secure domain.

This policy can be changed as the object space grows. The policy can be changed by delegating management permissions to other users. Or, the policy can be changed possibly by revoking certain, or all, permissions from cell_admin.

group iv-admin

This group represents the administrator group. Like cell_admin, all members of this group are considered administrators of the secure domain by the default policy. All default ACLs grant user cell_admin and group iv-admin exactly the same permissions.

You can easily place users into an administration role by adding them to the iv-admin group. Note that there can be some danger with this procedure. When a user becomes a member of this group, this user now has the default ACLs. With the default ACLs, that user now has full rights to do everything on any object in the entire namespace.

The default policy for this group can be changed. For example, you can change the default policy by delegating management permissions to other users. Or, you can change the default policy by revoking some or revoking all management permissions from iv-admin.

group ivmgrd-servers

This group contains the Management server. Policy Director currently requires that exactly one Management server exist in the secure domain. Therefore, this group contains only that one entry.

Most management requests made by the console are run using the Management server to the target Policy Director server. Because of this processing, the Management server must have permission to perform the request at the target server. For this reason, this group is granted server administration (s) permission in the default management ACL, and list (l) permission throughout the Web space.

group webseal-servers

This group contains all the WebSEAL servers in the secure domain. The default WebSEAL ACL grants these servers the complete set of HTTP-specific permissions and the delegation permission. This policy allows all WebSEAL servers to junction to all other WebSEAL servers. A modification of this policy could grant these permissions on a server-by-server basis.

Creating administration users

With Policy Director, you can create administration accounts with varying degrees of responsibility. Responsibility is delegated to administrators through strategically place administration ACLs. The following list illustrates possible administration roles:

ACL administration responsibilities

The ACL administrator can control all or part of a protected object namespace region, depending on where the administration ACL is placed. The administrator's ACL entry could contain the b, a, and T permissions, plus any other permissions appropriate for operations on objects in that region.

The administrator can use the Management Console to attach ACLs to objects in the designated namespace. The administrator can use the existing set of ACL templates. ACLs are attached using the attach (a) permission. This administrator does not have permissions to create, change, or delete ACL templates.

ACL policy responsibilities

The ACL policy administrator should be responsible for controlling the creation and modification of all ACL templates that are used in the secure

domain. The ACL policy administrator should be granted d, b, m, and v permission on the /Management or /Management/ACL object.

This ACL policy administrator can create new ACL templates, using the (m) permission. As the creator of a new template, the administrator becomes, by default, the first entry in the new ACL template, with abcT permissions. The control (c) permission effectively gives the administrator ownership of the ACL, and therefore the ability to change the ACL.

As owner of the ACL, the administrator is able to use the delete (d) permission that is granted in the management ACL. The administrator uses this permission to remove the ACL from the list of templates. You cannot delete an ACL template unless you are the owner of that ACL.

Server management responsibilities
This administrator is granted d, m, s, and v permissions on the /Management/Server object. This administrator can perform operations that involve the Policy Director servers.

Authorization Action responsibilities
This administrator is granted (d) and (m) permissions on the /Management/Action object. This administrator can create or delete all permissions that are created for third-party applications.

See “Management namespace” on page 89 for more information on the management namespace.

Example administration ACL template

The following example illustrates how a user gains administration rights.

- The following ACL on /WebSEAL gives administration rights to `user adam`:

```
user cell_admin                 abcTdmlrx
group iv-admin                  abcTdmlrx
group webseal-servers           gTdmlrx
group ivmgrd-servers            Tl
user adam                       abcTdmlrx
any-authenticated               Trx
unauthenticated                 Trx
```

- The following ACL on /NetSEAL gives administration rights to `user adam`:

```
user cell_admin                         abcTC
group iv-admin                          abcTC
user adam                               abcTC
any-authenticated                       TC
unauthenticated                         TC
```

Example management delegation

A large object space might require many administration users to manage a variety of subbranches. In this scenario, the ACLs for the directories on the path to each of these branches must contain entries for each account, with traverse permission. For a site with many administration users, these ACLs could contain a long list of entries that represent all these administration accounts.

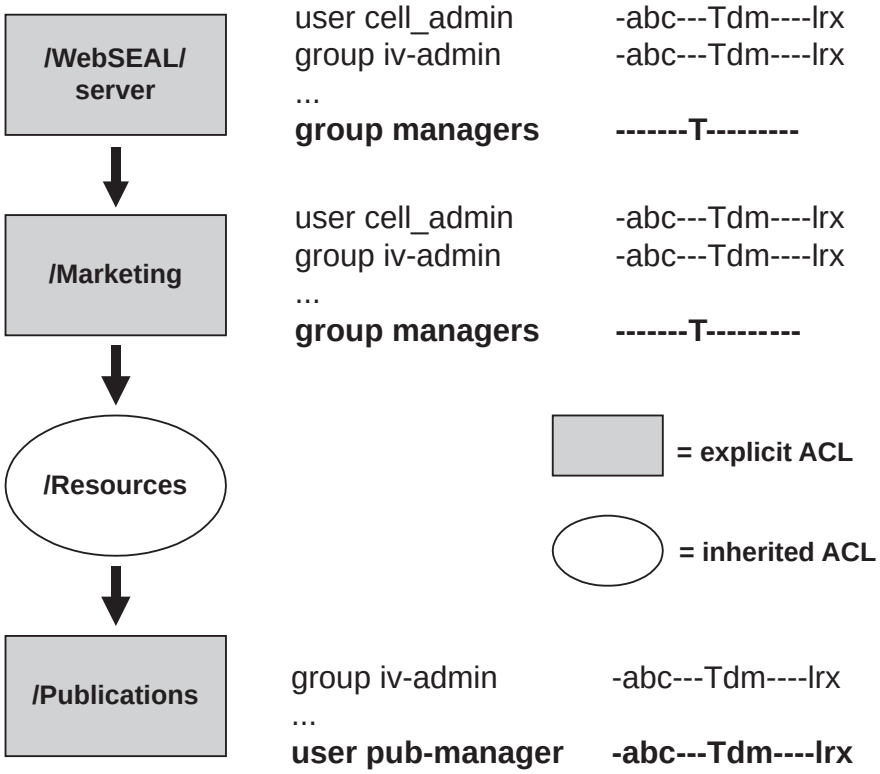
The following technique resolves the problem of numerous ACL entries for administrators:

1. Create an administration group account.
2. Add all new administration users to this group.
3. Add this group as an ACL entry (with traverse permission) to the directories that lead to each subbranch and that require management delegation.
4. At each branch root ACL, add the appropriate administration user entry (with b, c, T, plus other appropriate permissions).
5. The administrator can now remove the administration group ACL entry (and any other entry) from the root.

   Now only that user has control over the root and all objects below.

In the example below, the `managers` group has been created to contain all administration users. User `pub-manager` is a member of this group and therefore, has the necessary traverse permission required to navigate to the Publications directory.

The Publications directory includes the user `pub-manager` entry in its ACL. `pub-manager` is the delegated administrator of this branch and has the appropriate permissions. As the delegated administrator, `pub-manager` can remove the `manager` group account (and any other ACL entries) from the Publications ACL. By removing the group account and any other ACL entries, the delegated administrator gains total control over that branch of the Web space.



Chapter 8. Applying access control

In a secure domain, you can protect resources through the use of policy templates. Policy templates contain permissions that control the use of a resource. You must attach policy templates to the namespace object representation of the resource that require protection. Policy Director recognizes and uses a type of policy template that is known as the ACL. ACLs are used to stamp an organization's security policy onto the resources that belong to the secure domain.

This chapter discusses the common tasks that are required to manage the object space and apply access control.

This chapter includes:

ACL management overview

You use the Management Console's ACL management task panel to create, change, and delete ACL templates.

1. Log in to the Management Console as an ACL management administrator, such as `cell_admin`.
2. Click the **ACLs** task tab.

   The ACL management task panel appears.

Action buttons for ACL management tasks

You use the **ACL** action button to perform ACL management operations. The following table describes the tasks that each action button performs:

Action Button	Description
New ACL	Creates a new ACL template.
New Entry	Adds a new entry to the selected ACL template.
Save	Saves this ACL template. The ACL appears in the ACL Name list view.
Delete	Deletes the selected ACL template.
Get	Retrieves information about a specifically designated ACL template and populates the ACL detail view. Specify the ACL in the ACL Name field of the ACL Definition section.
List	Refreshes the list view.
Where Used	Displays the complete list of protected objects where the selected ACL template is attached. This display appears in the Bulletin Board section of the Console.

ACL management tasks

You can perform these ACL management operations:

- Create a new ACL template.
- Add an ACL entry.
- Edit permissions for an ACL entry.
- Delete an ACL template.

Creating a new ACL template

To create a new ACL template, you can start with one of the existing default ACL templates and change that ACL to your specifications.

1. From the **ACL Name** list, drag the icon of the default ACL template you want to use. The template can then be used as a foundation for your new ACL into the Bulletin Board.
2. From the ACL action buttons, click **New ACL**.

   Previous information in the ACL Definition area clears, and the fields are prepared for new entries.

   By default, your logged-in identity becomes the first ACL entry, with abcT permissions. The control (c) permission gives you ownership powers for this ACL.
3. Type the name of the ACL in the **ACL Name** field.
4. Tab to the **Description** field and type a phrase that describes the purpose for this ACL (how or why you would apply it).
5. Drag the default **ACL** icon in the Bulletin Board to the ACL Entry area of the new ACL Definition.

   The new ACL now contains the entries from the default ACL.
6. Make appropriate modifications to the entries.
7. Click **Save**.

Adding an ACL entry

To add an ACL entry:

1. From the **ACL Name** list, select an ACL template.
2. Click the **New Entry** action button.

   The ACL Entry area clears and resets itself for a new entry.
3. Click and hold the mouse button in the **Type** field.

   A drop-down menu appears.
4. Select a user, group, any-authenticated, or unauthenticated type.
5. Click in the **ID** field and type the appropriate ID.

   You can also drag and drop user and group icons from the **Accounts management** view. Click the **Accounts** task tab and move the **Accounts management** view to the lower panel area of the Console.
6. Use the permissions check boxes to apply appropriate permissions to this entry.
7. Click the **Save** button to commit the entry to the ACL.

Editing permissions for an ACL entry

To edit permissions for an ACL entry:

1. In the ACL Definition area, select the entry.
2. In the ACL Entry area, select appropriate permissions by selecting or not selecting the permission check boxes.
3. Click the **Save** button to commit the changes.

Deleting an ACL template

To delete an ACL template:

1. From the **ACL Name** list, select the ACL template you want to delete.
2. Click the **Delete** button.

   A warning box appears.
3. Click **Continue**.

   The Management Console does not delete an ACL that is still attached to an object. A message appears in the status bar warning you of this situation.

Example: You have attached an ACL that is designed for members of the `webtest` group. The `webtest` group is the group that are developing and testing new HTML pages. After testing, you can remove this explicit ACL to make the page available to other members of the secure domain.

Sample procedure for creating a new ACL template

To create a new ACL template:

1. Click the **New ACL** action button.

   Previous information in the ACL Definition area clears, and the fields are prepared for new entries.

   By default, your logged-in identity becomes the first ACL entry, with abcT permissions.
2. Type the name of the ACL in the **ACL Name** field.
3. Tab to the **Description** field and type a phrase that describes the purpose for this ACL (how or why you would apply it).
4. Click the **Save** button to commit this new ACL to the ACL List.
5. Click the **New Entry** button.

   The ACL Entry area clears and resets itself for a new entry.
6. Click and hold the mouse button in the **Type** field.

   A drop-down menu appears.
7. Click **unauthenticated** and grant no permissions.
8. Click the **Save** button.

   The entry appears in the ACL Definition area.
9. Follow the same procedures to add an any-authenticated entry with no permissions.

10. Click the **Accounts** task tab.

    The Accounts management panel appears as the top panel.
11. Click the **Move Task Down** button to place the Accounts management panel in the bottom portion of the Console.
12. To create a new group entry, click **New Entry** (ACLs panel).
13. From the Accounts panel **Groups** list, drag and drop a **group** icon to the **ID** field of the ACL Entry area.

    The **Type** and **ID** fields fill with the appropriate information.
14. Check the appropriate permissions.
15. Click the **Save** button to commit this entry to the ACL.

Object space management overview

You use the Management Console's Object Space management task panel to attach ACLs to objects and remove ACLs from objects.

1. Log in to the Management Console as a user with ACL management permissions, such as `cell_admin`.
2. Click the **Object Space** task tab.

   The Object Space management task panel appears.

Action buttons for Object Space management tasks

You use the **Object Space** action buttons to perform object space management operations. The following table describes the tasks that each action button performs:

Action Button	Description
Attach ACL	Assigns an ACL to an object.
Remove ACL	Removes an ACL from an object.
Find ACL	Finds all objects explicitly marked with a specific ACL. Objects are listed in the bottom panel of the Console.
Save ACL	Provides the ability to save a change made to an ACL when using the Edit ACL view.
List	Refreshes the Object Space tree.

Object Space management tasks

You can perform these object space management operations:

- Attach an ACL to an object.
- Remove an explicit ACL from an object.

Attaching an ACL to an object

You must have the appropriate management permissions to attach and remove ACLs. In particular, you must have the attach (a) permission to apply an ACL to objects and remove the ACL from objects.

1. Place the ACL management task panel in the bottom section of the Console.
2. Click the Object Space panel in the top panel of the Console.
3. Expand the appropriate region of the Object Space tree and select the target object where an explicit ACL is to be attached.
4. Drag the icon of the appropriate ACL template from the **ACL Name** list and drop it on the selected object in the Object Space tree.

Removing an explicit ACL from an object

You must have the appropriate management permissions to attach and remove ACLs. In particular, you must have the attach (a) permission to apply an ACL to objects and remove the ACL from objects.

1. Click the **Object Space** task tab.
2. Expand the appropriate region of the Object Space tree and select the target object with the attached explicit ACL.
3. Click the **Remove ACL** button.

Chapter 9. Managing proxy users

One of the IBM SecureWay FirstSecure (FirstSecure) set of security products is IBM SecureWay Boundary Server for Windows NT and AIX (Boundary Server). If LDAP is your default user registry, you can use Policy Director together with Boundary Server for an integrated IBM Firewall and Policy Director proxy user solution.

The most up-to-date Information about FirstSecure and its components can be found at this Web site:

`http://www.ibm.com/software/security/firstsecure/library`

This chapter includes:

Introducing boundary security

Like Policy Director, Boundary Server is one of the components provided with the IBM SecureWay FirstSecure product security package. Or, you can purchase Boundary Server or Policy Director individually and then run them as standalone products—without having to purchase the complete FirstSecure package.

Boundary security not only protects your network, applications and information, it also extends their reach. Proper boundary security requires that you control both who can access your network and what information enters and leaves your network. Boundary Server provides firewall protection, content security, and VPN. Boundary Server creates a boundary to the Internet that you can use to block potentially harmful viruses, Java script, applets, ActiveX controls, and even junk e-mail (SPAM).

See the *IBM SecureWay Boundary Server Up and Running* book that is provided with IBM SecureWay Boundary Server product for information on how to plan for, install, configure, use, and troubleshoot Boundary Server.

Boundary Server is a package of products. Boundary Server brings together best-of-breed technology from the security industry into an integrated solution. The solution includes IBM support and services, which you can optionally purchase.

One component of Boundary Server is IBM SecureWay Firewall Version 4.1 (Firewall).

Integrating with IBM Firewall

The purpose of a *firewall* is to prevent unwanted or unauthorized communication into or out of the secure network. A firewall serves as a blockade between one or more secure internal private networks, and other (nonsecure) networks or the public Internet.

IBM Firewall is a network security program. IBM SecureWay Firewall Version 4.1 includes these new features:

- Secure mail proxy enhancements
- Socks Protocol, Version 5, enhancements
- Remote Access Service (RAS)
- HTTP proxy

The HTTP proxy handles browser requests through the SecureWay Firewall, which eliminates the need for a socks server for Web browsing. Users can access useful information on the Internet without compromising the security of their internal networks and without requiring their client environments to implement HTTP proxy.

Before you can install SecureWay Firewall, you must ensure that you have the required prerequisites installed and configured. You must also define secure interfaces, determine and set up your security policy, and define network objects. The following key network objects must be defined:

- Secure interfaces of the firewall.
- Nonsecure interfaces of the firewall
- A secure network
- Each subnet on your secure network.
- A host network object for your Security Dynamics servers and your Windows NT domain servers, if appropriate.

Complete installation and configuration information can be found in the *IBM SecureWay Boundary Server Up and Running* book that is provided with Boundary Server.

Describing types of users

The administrators for IBM Firewall are responsible for configuring creating and changing definitions for proxy users, but they cannot create or modify the definitions of other firewall administrators.

Firewall administrators perform these administrative tasks:

- Adding users to the IBM Firewall so that they can access hosts outside their protected network.
- Changing the attributes of the users who access the firewall.
- Deleting users who no longer need access outside their network.

For an integrated IBM Firewall and Policy Director solution, it is the Policy Director administrator who takes over proxy user management.

Firewall users

Users in the secure network can access a nonsecure network by using a networking mechanism (such as Socks or proxy). If you want to allow your secure users to use a nonsecure network (proxy), you need to configure and set up the proper connections to allow this type of traffic.

Which services get carried out depend on decisions you make at the planning stage. Putting a service into effect often requires setting up some connection configurations to allow certain types of traffic. For example, if you want to allow your secure users to surf the web on the Internet by using HTTP Proxy, not only do you need to configure the HTTP Proxy daemon on the Firewall, but you also need to set up connections to allow HTTP traffic.

If you are going to require authentication for functions like outbound Web access, define these users to IBM Firewall.

Proxy users

Policy Director administration can be configured to manage proxy users as an extension of Policy Director users. For an integrated Policy Director and IBM Firewall solution, the Policy Director administrator must set up users as Policy Director proxy users.

A proxy user is someone who uses firewall services, such as the HTTP proxy service, to access Web sites on the Internet from within a corporate network. Proxy users are able to use services through the firewall but do not have access to the firewall machine and cannot perform local logins to the firewall machine.

Enabling Proxy User management

Before you can proceed with proxy user management, you must enable this feature on the Management Console. To enable Proxy User capability, you must edit the console.properties file.

You can find the console.properties file at:

Windows: `C:\Program Files\IBM\IVConsole\console.properties`

UNIX: `/opt/intraverse/ivconsole/console.properties`

To set up proxy user management:

1. Using a text editor, open the console.properties file.
2. Remove the comment symbol (#) from the beginning of this line:

   ```
   #6, ProxyUsersTaskView            = IV.ProxyUserTask.ProxyUsersTaskView
   ```
3. Restart the Management Console to enable the Proxy User management feature.

Introducing proxy user management

For Policy Director, firewall users are known as *proxy users*. The Policy Director Management Console allows you to manage proxy users.

Policy Director administrators perform these administrative tasks:

- Adding users as proxy users so that they can use firewall services.
- Changing the attributes of the proxy users who use firewall services.
- Deleting proxy users who no longer need to use firewall services.

Firewall administrators should refer to the IBM Firewall documentation for more information on how to perform these administrative tasks.

Using the proxy user management panel

The Users management task panel contains a **Users** tree view and a **Proxy User** detail view.

Using action buttons for proxy user management tasks

You use the **Proxy User** action buttons to perform proxy user management operations. The following table describes the tasks that each action button performs:

Action Button	Description
Save	Creates a new proxy user if a Policy Director user is selected in the user tree view. Or, changes an existing proxy user if an existing proxy user is selected in the user tree view.
Delete	Removes the selected proxy user.

Using proxy user detail fields

The following table describes the fields in the **Proxy User Detail** view of the Management Console:

Field	Description
Proxy User	The name being specified for a user being defined for proxy access. This is the user name that with which this user will log into the Telnet or FTP server on the IBM Firewall. This user has no administration authority.
Proxy Domain	Specifies the name of the firewall proxy domain.
Password	Specifies the password to be used to log in to the firewall proxy domain.
Description	Specifies the informational text string that describes the proxy user. The description is only an optional data field and is not used by the registry.
Remote Shell	Specifies the remote login shell to be used for the proxy user. The list of choices include: **/bin/restrict.sh**, **/bin/csh**, **/bin/ksh /bin/bsh**, **/bin/oneact.sh**, and an empty string.
Local Shell	Specifies the local login shell to be used for the proxy user. The list of choices include: **/bin/restrict.sh**, **/bin/csh**, **/bin/ksh /bin/bsh**, **/bin/oneact.sh**, and an empty string.
Default Group	Specifies the default group to which the proxy user belongs. The administrator can select the group from a list of groups presented of which the proxy user is a member.

Secure FTP Authentication	Specifies the level of authentication this user needs to use FTP to access the firewall from the secure network. The list of choices include: **Firewall Password**, **Permit All**, **Deny All**, **SecurID Card**, **NT Logon Password**, **User-Supplied 1**, **User-Supplied 2**, **User-Supplied 3**, **AIX Logon Password**, and an empty string.
Remote FTP Authentication	Specifies the level of authentication this user needs to use FTP to access the firewall from the nonsecure network. The list of choices include: **Firewall Password**, **Permit All**, **Deny All**, **SecurID Card**, **NT Logon Password**, **User-Supplied 1**, **User-Supplied 2**, **User-Supplied 3**, **AIX Logon Password**, and an empty string.
Secure Telnet Authentication	Indicates whether this user's identity, when logging in from the secure network, must be authenticated by some means. The list of choices include: **Firewall Password**, **Permit All**, **Deny All**, **SecurID Card**, **NT Logon Password**, **User-Supplied 1**, **User-Supplied 2**, **User-Supplied 3**, **AIX Logon Password**, and an empty string.
Remote Telnet Authentication	Indicates whether this user's identity, when logging in from the nonsecure network, must be authenticated by some means. The list of choices include: **Firewall Password**, **Permit All**, **Deny All**, **SecurID Card**, **NT Logon Password**, **User-Supplied 1**, **User-Supplied 2**, **User-Supplied 3**, **AIX Logon Password**, and an empty string.
Secure SOCK Authentication	Specifies the Socks, Version 5, authentication method for Socks client connections coming from the secure side of the firewall. The list of choices include: **Firewall Password**, **Permit All**, **Deny All**, **SecurID Card**, **NT Logon Password**, **User-Supplied 1**, **User-Supplied 2**, **User-Supplied 3**, **AIX Logon Password**, and an empty string.
Remote SOCK Authentication	Specifies the Socks, Version 5, authentication method for Socks client connections coming from the nonsecure side of the firewall. The list of choices include: **Firewall Password**, **Permit All**, **Deny All**, **SecurID Card**, **NT Logon Password**, **User-Supplied 1**, **User-Supplied 2**, **User-Supplied 3**, **AIX Logon Password**, and an empty string.
Secure HTTP Authentication	Specifies a user ID and password pair type of authentication on outbound HTTP proxy requests. The list of choices include: **Firewall Password**, **Permit All**, **Deny All**, **SecurID Card**, **NT Logon Password**, **User-Supplied 1**, **User-Supplied 2**, **User-Supplied 3**, **AIX Logon Password**, and an empty string.
Local Authentication	Specifies the local authentication method. The list of choices include: **Firewall Password**, **Permit All**, **Deny All**, **SecurID Card**, **NT Logon Password**, **User-Supplied 1**, **User-Supplied 2**, **User-Supplied 3**, **AIX Logon Password**, and an empty string.
Idle Disconnect Time	Specifies the idle time allowed before a used is disconnected.
Warning Disconnect Time	Specifies the amount of warning time the user is allowed before being disconnected.
Password Valid	Specifies whether the user should be prompted for, and asked to enter, a valid password. The IBM Firewall will prompt for a password for this user.
Password Locked	Specifies whether the password is locked. The administrator can set this field to `yes` to prevent a user from using password authentication.

Adding a proxy user

To create a proxy user:

1. Click the **Proxy User** task tab.
2. Expand the appropriate region of the **Users** tree, and select the Policy Director user that you want to make a proxy user.
3. Fill in the fields in the **Proxy User Detail** view.
4. Click the **Save** button.

Changing proxy user information

To change the proxy user information:

1. Click the **Proxy User** task tab.
2. Expand the appropriate region of the **Users** tree view, and select an existing proxy user from the list.

   The Proxy User Detail area populates with current data.
3. Enter the new data.
4. Click the **Save** button.

Removing a proxy user

To delete a proxy user:

1. Click the **Proxy User** task tab.
2. Expand the appropriate region of the **Users** tree view and select an existing proxy user.
3. Click the **Delete** button.

Using the ivadmin policy commands for proxy user management

There are specific **ivadmin policy** commands that are used only with Policy Director proxy users. These **ivadmin policy** commands are a set of management commands that control general policy information for Policy Director users and proxy users. The administrator can manage the following policy attributes:

- “Managing login policies” on page 117.
- “Managing password policies” on page 118

A *policy* defines the set of constraints placed on user accounts and passwords in order to improve the overall security of the system. These constraints can be imposed generally (globally across every user in the system) or specifically (only to a specified user). If user has a specific policy applied, this specific policy takes precedence over any general policy that also might be defined. The precedence applies, regardless of whether the specific policy is more or less restrictive than the general policy.

Managing login policies

The following **ivadmin policy** commands allow the IBM SecureWay Boundary Server administrator to manage login-related policies.

Use the login-related **policy** management commands to create new login policies. These policies apply to all users.

For login-related policies, Policy Director defines relative time as DDD-hh:mm:ss, and defines absolute time as YYYY-MM-DD-hh:mm:ss when referring to **policy** management commands.

Command	Description
policy set disable-time-interval [*number*]	
policy get disable-time-interval	

	Specifies how long (in seconds) an account should be disabled after the maximum number of failed login attempts is reached.
	
	The *number* argument is the number of seconds an account should be disabled.
	
	Examples:
	`ivadmin> policy set disable-time-interval 3`
	
	Or:
	`ivadmin> policy get disable-time-interval`

policy set max-login-failures [*number*]	
policy get max-login-failures	

	Creates a new policy or displays an existing policy for the maximum number of failed login attempts allowed. The *number* argument is the maximum number of failed login attempts to be allowed.
	
	Examples:
	`ivadmin> policy set max-login-failures 5`
	
	Or:
	`ivadmin> policy get max-login-failures`

Managing password policies

The following **ivadmin policy** commands allow the IBM SecureWay Boundary Server administrator to manage password policies.

For password-related policies, Policy Director defines relative time as DDD-hh:mm:ss when referring to **policy** management commands.

Command	Description
policy set max-password-age [*relative-time*]	
policy get max-password-age	

	Manages the policy controlling the maximum time before a password must be changed. The *relative-time* argument is the time specified—expressed in days, hours, and minutes following this format: `DDD-hh:mm:ss`
	
	Examples:
	`ivadmin> policy set max-password-age 031-08:30:00`
	
	Or:
	`ivadmin> policy get max-password-age`

policy set max-password-repeated-chars [*number*]	
policy get max-password-repeated-chars	

	Manages the policy for the maximum number of characters that can be repeated sequentially in a user password.
	
	Examples:
	`ivadmin> policy set max-password-repeated-chars 3`
	
	Using this example of three-maximum repeated characters, the password `deptfff` could be set as a password because it does not exceed three “f” repeated characters. The password `deptffff` could not be set as a password because four “f” characters exceed the limit of three repeated characters.
	
	Or:
	`ivadmin> policy get max-password-repeated-chars`

policy set min-password-alphas [*number*]	
policy get min-password-alphas	

	Manages the policy for the minimum number of alpha characters that must be used for a user password.
	
	Examples:
	`ivadmin> policy set min-password-alphas 5`
	
	Using this example, a password must contain at least five alpha characters.
	
	Or:
	`ivadmin> policy get min-password-alphas`

policy set min-password-non-alphas [*number*]	
policy get min-password-non-alphas	

	Manages the policy for the minimum number of non-alpha characters that must be used for a user password.
	
	Examples:
	`ivadmin> policy set min-password-non-alphas 1`
	
	Using the example, a password must contain at least one non-alpha character to be valid.
	
	Or:
	`ivadmin> policy get min-password-non-alphas`

policy set min-password-different-chars [*number*]	
policy get min-password-different-chars	

	Manages the policy for the minimum number of different characters that must be used for a user password.
	
	Examples:
	`ivadmin> policy set min-password-different-chars 3`
	
	Using the example, a password must have at least three characters that are different to be valid. If the password specified is `ddddyyyy`, the password would not be valid because it only contains two different characters (d and y).
	
	Or:
	`ivadmin> policy get min-password-different-chars`

policy set min-password-length [*number*]	
policy get min-password-length	

	Manages the policy for the minimum length, in characters, for a password. The *number* argument is the minimal length allowed for a password.
	
	Examples:
	`ivadmin> policy set min-password-length 8`
	
	Or:
	`ivadmin> policy get min-password-length`

policy set min-password-reuse-num [*number*]	
policy get min-password-reuse-num	

	Specifies the number of times a password must be changed before a previously used password can be used again.
	
	Examples:
	`ivadmin> policy set min-password-reuse-num 3`
	
	Or:
	`ivadmin> policy get min-password-reuse-num`

policy set min-password-reuse-time [*relative-time*]	
policy get min-password-reuse-time	

	Manages the policy for the minimum time that must pass before a password can be reused.
	
	The *relative-time* argument is the minimum time, expressed in days, hours, and minutes in this format (`DDD-hh:mm:ss`). A user cannot use the same password again within a specified time limit (for example, 60 days or `060-00:00:00`).
	
	Examples:
	`ivadmin> policy set min-password-reuse-time 060-00:00:00`
	
	Or:
	`ivadmin> policy get min-password-reuse-time`

policy set password-expiry-date [*relative-time*]	
policy get password-expiry-date	

	Manages the policy for the date and time the password is to expire.
	
	Examples:
	`ivadmin> policy set password-expiry-date 031-08:30:00`
	
	Or:
	`ivadmin> policy get password-expiry-date`

policy set password-expiry-warn [*number*]	
policy get password-expiry-warn	

	Manages the policy that warns the user that the password is about to expire. The *number* argument is the number of days the warnings begin before the expiry date (for example, four days before the password is about to expire).
	
	Examples:
	`ivadmin> policy set password-expiry-warn 4`
	
	Or:
	`ivadmin> policy get password-expiry-warn`

Chapter 10. Managing the Policy Director servers

This chapter includes general tasks for administering and configuring the set of Policy Director servers. The configuration files that support each server are also discussed.

This chapter includes:

Introducing the Policy Director servers

The Policy Director server consists of the following server processes (daemons):

- Security server (secd)
- Security Manager (secmgrd)
- Authorization server (ivacld)
- Management server (ivmgrd)
- Directory Service Broker (DSB)

These servers are automatically configured during product installation.

The **Security server (secd)** is a DCE server only. The Security server provides authentication services. Also, the Security server maintains a centralized registry database. The user registry can be either LDAP or DCE, If the user registry is DCE, this centralized registry database contains account information for all valid users who participate in the secure domain.

The **Security Manager (secmgrd)** contains the WebSEAL and NetSEAL Security servers.

The **Authorization server (ivacld)** services authorization requests from any third-party applications that use the Policy Director Authorization API in remote mode. The Authorization server typically requires very little administration or configuration.

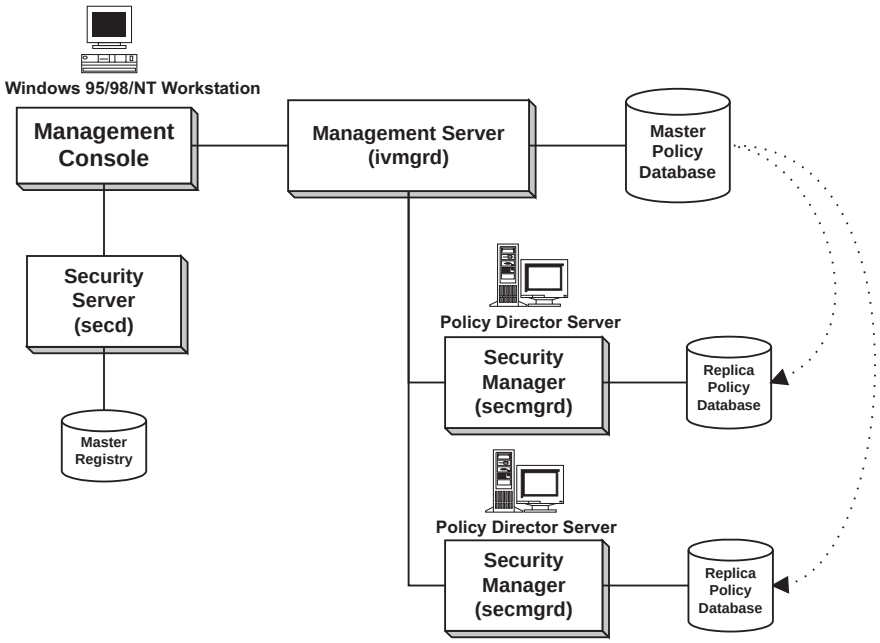
The **Management server (ivmgrd)** manages the primary ACL database, and maintains location information about other WebSEAL and NetSEAL servers in a secure domain. The Management server typically requires very little administration or configuration.

The **Directory Services Broker (DSB)** is distributed as part of the Management Server (IVMgr) package. The Management Console requires a Directory Services Broker in the secure domain when running on Windows NT, Windows 95, or Windows 98 workstations. The Directory Services Broker typically requires no administration or configuration after initial installation.

Server dependencies

Policy Director server dependencies include:

- There can be only one instance of the Management server and its primary authorization (ACL) database in any secure domain.
- The Management server replicates its ACL database to all other Policy Director servers in the secure domain.
- A Security Manager, with WebSEAL and NetSEAL traps, resides on every Policy Director server.
- Each Security Manager applies access control policy that is based on information from the replicated authorization, or ACL, database.



Overview of server administration tools

You can perform server administration through the following interfaces:

- **ivadmin** utility
- **wandmgr** utility (WebSEAL only)
- UNIX scripts
- Windows NT Services Control Panel

This chapter describes how to use each of these interfaces.

The **ivadmin**, **wandmgr**, and the startup scripts provide command-line interfaces. These are useful when automating server administration tasks within shell scripts.

The Management Console, ivadmin, and wandmgr can all be used remotely or locally. The startup scripts must be administered locally.

When finding and correcting problems, the command-line utilities can provide status information and control of individual servers.

ivadmin utility

Policy Director provides the **ivadmin** command-line utility to accomplish more advanced server tasks. Use **ivadmin**:

- To perform all Management Console tasks listed in the previous section
- To display server status

wandmgr utility

The **wandmgr** command-line utility is a Policy Director WebSEAL tool that is used to perform advanced Web client authorization and cache management tasks, such as:

- To display Web object cache status.
- To delete Web object caches from memory.

UNIX scripts

Policy Director uses scripts to automatically stop and start servers during a system boot, and to display server status. These scripts can be started manually:

- To stop servers.
- To display server status.
- To start servers.

Windows NT Services Control Panel

Use the Windows NT Services Control Panel:

- To start a server.
- To stop a server.
- To pause (suspend) a server.
- To continue (resume) a paused server.
- To list configured servers.

Server configuration files

Policy Director servers use configuration files to dictate functionality:

Server Name	Process	Configuration File

Security Manager	secmgrd	**UNIX:**
		`/opt/intraverse/secmgr/lib/secmgrd.conf`
		
		Windows:
		`\Program Files\ibm\Policy Director\secmgr\lib\secmgrd.conf`

Management server	ivmgrd	**UNIX:**
		`/opt/intraverse/ivmgrd/lib/ivmgrd.conf`
		
		Windows:
		`\Program Files\ibm\Policy Director\ivmgrd\lib\ivmgrd.conf`

Authorization server	ivacld	**UNIX:**
		`/opt/intraverse/ivacld/lib/ivacld.conf`
		
		Windows:
		`\Program Files\ibm\Policy Director\ivacld\lib\ivacld.conf`

Configuration files are American National Standard Code for Information Interchange (ASCII) based and can be edited using a common editor. File entries have the following format:

```
parameter=value
```

Initial installation of a Policy Director server sets default values for most parameters. Some parameters are static and never change; others can be adjusted or added to configure server functionality and optimize performance.

Note: After editing a configuration file, you must stop and restart the Policy Director server before the changes can take effect.

Each file contains sections, or **stanzas**, with settings for a particular configuration category. The stanza labels appear within brackets [ ].

For example, the `[intraverse]` stanza in iv.conf defines the general Policy Director configuration settings that apply to the entire secure domain. The stanza `[wand-mime-types]` defines MIME-type definitions that are supported by Policy Director WebSEAL on the local system.

The files are commented to explain the use of each parameter. If you must change any configuration settings, carefully edit the files to ensure their integrity.

UNIX: Stopping and starting Policy Director servers

Starting and stopping the server processes is normally accomplished through automated scripts that run at system startup and shutdown.

The administrator can also use scripts to manually start and stop the server processes. This technique is useful when customizing an installation or when finding and correcting problems. You can only apply scripts to the local machine. Use the Management Console or the **ivadmin** utility to stop and start servers remotely.

You can start and stop the Policy Director servers all at once or stop them one at a time individually. Generally, the servers must be stopped and be started in the correct order.

Processing Cell Directory Services (CDS) requests on behalf of NetSEAT clients requires the Directory Services Broker. (The Windows version of the Management Console uses the NetSEAT client.)

Stopping using the iv script

Use the iv script to stop all Policy Director servers on a particular machine in the correct order:

AIX:

```
# /etc/iv/iv stop
```

Solaris:

```
# /etc/init.d/iv stop
```

This script stops only in the following order: ivacld, secmgrd, and ivmgrd. The script waits until all servers have stopped before returning the prompt.

Manual shutdown

The servers can also be individually stopped using the **kill** command:

`# kill <pid>`	Forces the server to shutdown cleanly.
`# kill -9 <pid>`	Abruptly kills the server, without any cleanup.

Shut down the Policy Director servers in the following order:

1. Directory Services Broker (DSB)
2. Authorization server (ivacld)
3. Security Manager (secmgrd)
4. Management server (ivmgrd)

Starting using the iv script

Use the iv script to start all Policy Director servers on a particular machine in the correct order:

AIX:

```
# /etc/iv/iv start
```

Solaris:

```
# /etc/init.d/iv start
```

This script starts only in the following order: ivmgrd, secmgrd, and ivacld. The script waits until all servers have started before returning the prompt.

Manual startup

You can manually start servers individually by starting the server directly. The server initializes itself. If successful, the server demonizes itself.

You must perform the startup commands as an administrative user, such as root or ivmgr. Start the Policy Director servers in the following order:

1. Management server (ivmgrd):
   ```
   # /opt/intraverse/ivmgrd/bin/ivmgrd
   ```
2. Security Manager (secmgrd):
   ```
   # /opt/intraverse/secmgr/bin/secmgrd
   ```
3. Authorization server (ivacld):
   ```
   # /opt/intraverse/ivacld/bin/ivacld
   ```
4. Directory Services Broker (DSB):
   ```
   # /opt/intraverse/broker/bin/dsb
   ```

Displaying server status

To check whether a server is running, use the following command:

AIX:

```
# /etc/iv/iv status
```

Solaris:

```
# /etc/init.d/iv status
```

```
DCE servers:
Server                    Enabled                   Running
dced                      yes                       yes
secd                      -                         yes
cdsd                      -                         yes
dtsd                      -                         yes
dsb                       -                         yes

Policy Director servers:
Server                    Enabled                   Running
ivmgrd                    yes                       yes
secmgrd                   yes                       yes
ivacld                    yes                       yes
```

Windows: Stopping and starting Policy Director servers

Use the Windows NT Services Control Panel to start and stop the server processes manually. This can be useful when customizing an installation, or when finding and correcting problems. You must have administrative privileges to use this utility.

You can start and stop the Policy Director servers all at once or stop them one at a time individually. Generally, you must stop and start the servers in the correct order.

The Policy Director AutoStart Service automatically starts each of the Policy Director servers whenever you restart (reboot) the system. After the servers start, the AutoStart Service exits.

Use the Windows NT Services Control Panel to manually start and stop the individual Policy Director servers:

1. Open the Windows Control Panel.
2. Double-click the **Services** icon.

   The Services dialog box appears. Examples of services you might see include:

   ```
   Service                                 Status          Startup
   Director Services Broker                Started         Automatic
   Policy Director Authorization Server    Started         Manual
   Policy Director Auto-Start Service      Started         Automatic
   Policy Director Management Server       Started         Manual
   Policy Director Security Manager        Started         Manual
   Policy Director X.509 Authorization Server  Started     Manual
   ```

3. From the list box, select the Policy Director servers according to the sequence that is indicated in steps 4 on page 127 and 5 on page 127.

4. Stop the servers in the following order:
   - Security Manager
   - Management server
   - Directory Services Broker
5. Start the servers in the following order:
   - Directory Services Broker
   - Management server
   - Security Manager
   - Authorization server
6. Click the appropriate control option button (**Start, Stop, Startup**) from the right-hand side of the box.
7. To prevent the automatic startup of a Policy Director server by the Policy Director AutoStart Service, use the **Startup** option button. This button sets that Policy Director server to Disabled.

Automating server startup at boot time

The [intraverse] stanza of the iv.conf configuration file contains parameters for either automating, or not automating, server startup.

When you install, you can configure the Security server daemon (secmgrd) to start automatically after each restart of the system.

```
[intraverse]
boot-start-secmgrd = yes
```

To prevent automatic secmgrd startup, set:

```
boot-start-secmgrd = no
```

When you install the IVMgr package, the Policy Director Management server daemon (ivmgrd) automatically starts after each restart of the system.

```
[intraverse]
boot-start-ivmgrd = yes
```

To prevent automatic ivmgrd startup, set:

```
boot-start-ivmgrd = no
```

Note: Each secure domain (cell) requires exactly one Policy Director Management server daemon. Do not install and run **ivmgrd** on more than one server per cell.

When you install the IVAcld package, the Policy Director Authorization server daemon automatically starts after each restart of the system.

```
[intraverse]
boot-start-ivacld = yes
```

To prevent automatic **ivacld** startup, set:

```
boot-start-ivacld = no
```

Configuring RPC worker threads

The number of configured worker threads specifies the number of concurrent incoming requests that a server can service. When all worker threads are busy, Policy Director buffers other connections that arrive until a worker thread is available.

You can set the number of threads available to service incoming connections. Configure the number of worker threads carefully because of possible performance impacts.

The configuration parameters do not impose an upper boundary on the number of simultaneous connections. These parameters simply specify the number of threads that are made available to service a potentially unlimited work queue.

Choosing the optimal number of worker threads depends on understanding the quantity and type of traffic on your network.

By increasing the number of threads, the average time it takes to finish the requests decreases. However, increasing the number of threads increases server overhead and causes the average time to service a request to rise again.

Each of the configuration files for the secmgrd, ivmgrd, and ivacld servers contains the following parameters for configuring RPC worker threads:

- Maximum number of RPC worker threads
- TCP port to listen on for incoming RPCs
- User Datagram Protocol (UDP) port to listen on for incoming RPCs

Server Name	Process	Configuration File
Security Manager	secmgrd	secmgrd.conf
Management server	ivmgrd	ivmgrd.conf
Authorization server	ivacld	ivacld.conf

Setting the RPC worker threads pool

The Policy Director servers use RPC worker threads to process:

- Incoming RPC requests from NetSEAT clients
- Database updates generated by administrative tasks performed from the Management Console

The maximum RPC worker threads parameter, located in each server configuration file, contains the following default value:

```
max-rpc-worker-threads = 10
```

Consider increasing this value when the Policy Director server handles a large number of NetSEAT clients.

Configuring servers for incoming RPC requests

The following table lists the default port values for RPC listening by each server:

Server	Configuration File	Port Parameters with Default Values
secmgrd	secmgrd.conf	`rpc-tcp-port = 6052 rpc-udp-port = 0`
ivmgrd	ivmgrd.conf	`tcp-rpc-port = 6032 udp-rpc-port = 0`
ivacld	ivacld.conf	`tcp-rpc-port = 6031 udp-rpc-port = 0`

A port value of zero (0) disables RPC listening in that port. It is highly recommended that you use TCP listening. Activate UDP ports only when absolutely necessary.

You can set different ports as needed.

Example for secmgrd:

```
rpc-udp-port = 6052
```

TCP and UDP now listen on the same port.

Chapter 11. Managing the authorization service

The Policy Director Authorization Service enforces network security policy by controlling the authorization decision-making process. You can extend Policy Director authorization capabilities in several ways: Define and incorporate additional namespaces, define new access control permissions, and accommodate third-party *external authorization services*. This chapter discusses the tasks that are required to configure, maintain, and extend the Policy Director Authorization Service.

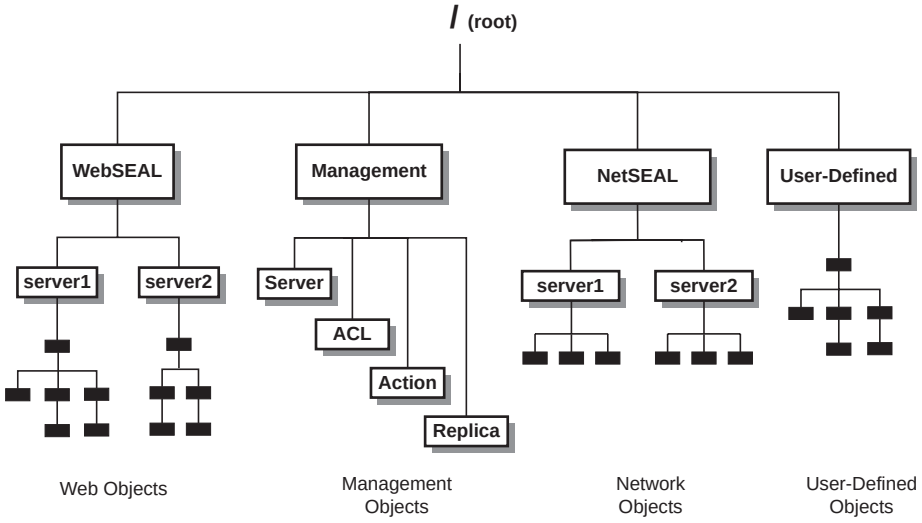
This chapter includes:

Defining third-party application namespaces

These factors define a Policy Director security policy:

- Who will be allowed to participate in the secure domain?
- What objects must be protected?
- What rules must protect those objects?

The Policy Director protected object namespace is the logical and hierarchical representation of resources that belong to the secure domain. The objects in the namespace represent the system resources that are to be protected (such as files and ports). To protect any resource in the secure domain, you attach policy templates (ACLs) to the object representations of those resources.



The protected object namespace uses two types of objects:

Container objects
Container objects are structural designations that allow you to organize the namespace hierarchically into distinct functional regions. Container objects contain resource objects.

Resource objects
Resource objects are the representations of actual system resources (such as services, files, and programs) in your secure domain.

Policy Director allows you to extend its authorization services to objects that belong to a third-party namespace. Integration of a third-party namespace with Policy Director requires these steps:

- Describe the third-party application's namespace to Policy Director.
- Apply policy templates (ACLs) to any namespace objects that require protection.

You describe the contents of a third-party namespace to Policy Director through a special mapping file. This file lists the specific resource objects belonging to the third-party namespace and indicates their hierarchical relationship.

In addition, you must define the root container object that holds this third-party namespace. The root container object name will appear as part of the Policy Director namespace when displayed by the Management Console (Object Space tab). Existing standard Policy Director container objects include /WebSEAL, /NetSEAL, and /Management.

The Management server configuration file (ivmgrd.conf) defines the name of the third-party container object and the location of the mapping file.

Root container object name and map file location

The `[object-spaces]` stanza of the Management server configuration file (ivmgrd.conf) defines both of these items:

- The name of the root container object for the third-party namespace.
- The location of the mapping file.

Each entry has the following format:

`object-space-root = map-file`

Where:

object-space-root	The name of the container object that holds the third-party namespace.
map-file	The full path name to the map file. The mapping file can be located anywhere.

The following example defines a third-party container object (`Notes`) and the location of a map file: `notemap.txt`:

UNIX: `/Notes = /opt/intraverse/lib/notemap.txt`

Windows: `/Notes = C:\Program Files\IBM\Policy Director\lib\notemap.txt`

Note: You must stop and restart the Management server to incorporate any edits to the ivmgrd.conf file.

Mapping-file format

The mapping file that describes the third-party namespace is an ASCII text file. Each line in the file is an absolute path name that represents each resource object in the namespace. The mapping file lists only resource objects; Policy Director implies container objects from the path names.

Additional map file rules include:

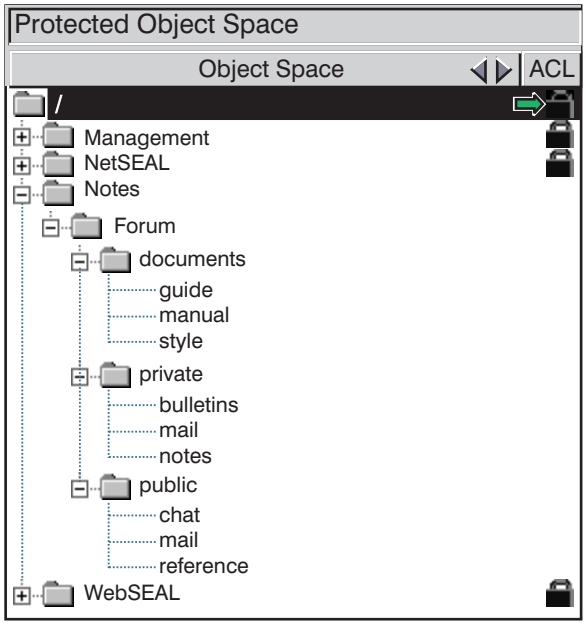
- List only one object and path name per line.
- Object path names always start with a forward slash (/).

Example mapping file:

```
/Forum/public/mail
/Forum/public/reference
/Forum/public/chat
/Forum/documents/style
/Forum/documents/guide
/Forum/documents/manual
/Forum/private/mail
/Forum/private/notes
/Forum/private/bulletins
```

Hierarchical display in the Management Console

The following Management Console display results from the example mapping file that is described in “Mapping-file format”.



Defining custom ACL permissions

Policy Director relies on policy templates to specify the conditions necessary to perform an operation on a protected object. Policy Director uses a specific type of policy template, known as an Access Control List (ACL).

ACL entries

You can attach an ACL to an object. When attached, entries in the ACL specify which operations Policy Director allows on this object and who can perform those operations. An ACL entry includes:

- User type or group type

  There is also a type for unauthenticated and any-authenticated users.
- Unique user identity or group identity
- Permissions

Permissions

Policy Director uses a standard set of permissions that cover a wide range of operations. Single printable ASCII characters represent the permissions. In the Management Console (**ACLs** tab), Policy Director displays each permission with a label that describes the operation it governs. In addition, Policy Director groups the ACLs together, according to their use in a particular namespace or for use across the entire namespace. Examples of group categories include: Base, Generic, WebSEAL, NetSEAL.

Operations on an object

Application software typically contains one or more operations that are performed on protected objects. These applications make calls into the authorization service before the requested operation is allowed to progress. This call is made using the Policy Director Authorization API for both Policy Director and third-party applications.

Information is contained in the ACL that protects the object. The authorization service uses the information to make a simple `yes` or `no` response to the question: `Does this user (group) have the "r" permission (for example) for the requested object?`

It is important to note that the authorization service knows nothing about the operation that requires the read (r) permission. All it cares about is the presence (or lack of presence) of the read (r) permission. The read (r) permission is in the ACL entry of the requesting user or group.

This permission is a very powerful feature of the Policy Director Authorization Service. The service is completely independent of the operations that are being requested, which is why it is easy to extend the benefits of the authorization service to third-party applications.

Requirements for custom permissions

The entire repertoire of standard Policy Director permissions is available to third-party applications. A third-party application can make use of a Policy Director permission. If done, the associated operation should very closely match the actual operation that is normally performed by Policy Director. For example, an operation that requires a read-only access to a protected object should use only the read (r) permission.

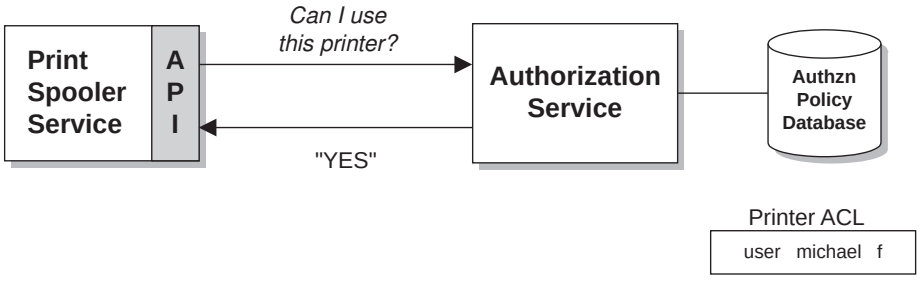
Note: A third-party application can use a standard Policy Director permission for a completely unrelated operation, which does not know or care about the operation. However, this would cause difficulty for an administrator who would have to distinguish between two dissimilar uses of the same permission.

A third-party application might use an operation that is not represented by the standard set of permissions. If used, Policy Director lets you define a new permission. This application uses this permission, which the authorization service recognizes.

Example: The requirement in this example is to protect a certain printer device from unauthorized use. A third-party print spooling service is written with the Policy Director Authorization API. This print spooling service would call the authorization service to perform ACL checks on requests that are made to the printer.

The standard Policy Director permissions do not include a permission for protecting printers. The newly created print permission in this example should protect the printer.

An ACL is now attached to the printer object. If a user requests the use of the protected printer, that user must have an ACL entry that contains the print permission. The authorization service returns a favorable response if the print permission is present and the printing operation proceeds. If the authorization service finds no existence of a print permission, the printing operation will not be allowed to proceed.



Managing permissions

As the Policy Director administrator, you can manage permissions by:

- Adding custom permissions
- Removing custom permissions
- Viewing all available permissions

Creating a custom permission

You use the **ivadmin action** commands to create, delete, and list new permissions. You must be logged in as a Policy Director administrator in order to use the **ivadmin** utility.

Use the following command syntax to create a new custom permission:

```
ivadmin> action create name description action-type
```

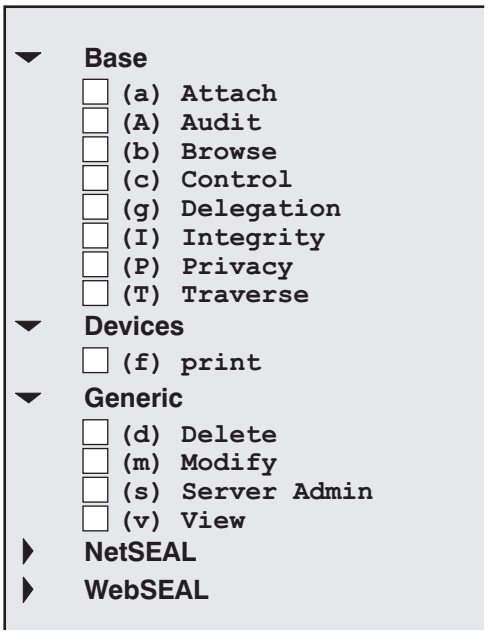
Where:

`name`	The printable ASCII character that represents the permission.
`description`	The descriptive label that appears to the right of the character in the Management Console display (ACLs tab).
`action-type`	The organizational category where this permission appears in the Management Console display (ACLs tab).

For example, typing:

```
ivadmin> action create f print Devices
```

Results in this new entry in the ACLs management panel of the Management Console.



Deleting a custom permission

Use the following command syntax to delete a custom permission:

```
ivadmin> action delete name
```

For example:

```
ivadmin> action delete f
```

Listing all available permissions

Use the following command syntax to list all available permissions:

```
ivadmin> action list
```

You would see a list of permissions similar to:

```
p "Proxy" NetSEAL
r "Read" WebSEAL
v "View" Generic
x "Execute" WebSEAL
A "Audit" Base
a "Attach" Base
b "Browse" Base
c "Control" Base
C "Connect" NetSEAL
d "Delete" Generic
f "Print" Devices
g "Delegation" Base
I "Integrity" Base
l "List Directory" WebSEAL
m "Modify" Generic
P "Privacy" Base
s "Server Admin" Generic
T "Traverse" Base
...
```

Defining external authorization services

An *external authorization service* allows you to impose additional authorization controls and conditions to supplement the standard Policy Director authorization process. A separate authorization server program dictates these additional controls and conditions.

The Policy Director Authorization Service automatically builds in the external authorization capability. If you configure an external authorization service, the Policy Director Authorization Service simply incorporates the new controls and conditions into its evaluation process.

Setting up an external authorization service requires these general steps:

1. Write a server program that can be referred to during an authorization decision.
   Refer to the *Policy Director Programmer's Guide and Reference*.
2. Register the external authorization service with Policy Director.
   Refer to "Registering an external authorization service" on page 138.

After you register the service, a new permission that represents this service will appear in the Policy Director Management Console. You can now use this permission in any ACL entry.

When Policy Director encounters the permission during an authorization check, it refers to the external authorization service for additional authorization decisions.

For further background information, refer to "External authorization capability" on page 48.

Registering an external authorization service

Use the **ivadmin server register** command to inform the Policy Director Authorization Service of the existence and location of an external authorization service.

The following syntax applies:

```
ivadmin> server register externauth server-name ns-location server-principal
         action-char action-name
```

Where:

server-name	A name (or label) for this external authorization service. This is the name that appears in the display of the object space on the Management Console and in the ivadmin server list command.
ns-location	The RPC entry in the CDS namespace where the external authorization server exports its RPC bindings.
server-principal	The LDAP name or DCE principal name for the external authorization server process.
action-char	The permission character used in an ACL to dictate the use of this external authorization service for supplemental authorization decisions.
action-name	The descriptive label that appears to the right of the character in the Management Console display (ACLs tab).

This command produces a default ACL organization category, which is called *external authorization*. The Management Console uses the default ACL organization category when displaying ACLs. The permissions for all registered external authorization services appear under this category.

For example, typing:

```
ivadmin> server register externauth timechecker /.:/subsys/timechk
         t-checker k time-check
```

Registers an external authorization server that is named `timechecker` with the authorization service. The RPC entry in the CDS namespace where timechecker exports its RPC bindings is `/.:/subsys/timechk`. The DCE principal name for the server is `t-checker`. The permission associated with this service is the time-check (k) permission.

The permission for this registered external authorization server appears on the Management Console similar to the following:

```
Base
  (a) Attach
  (A) Audit
  (b) Browse
  (c) Control
  (g) Delegation
  (I) Integrity
  (P) Privacy
  (T) Traverse
Generic
  (k) time-check
Generic
  (d) Delete
  (m) Modify
  (s) Server Admin
  (v) View
NetSEAL
WebSEAL
```

Deleting an external authorization server

Use the **ivadmin server delete** command to remove a registered external authorization service. The following syntax applies:

```
ivadmin> server delete /ExternAuthzn/server-name
```

Where:

`server-name`	A name (or label) for this external authorization service. This is the name that appears in the display of the object space on the Management Console.

For example:

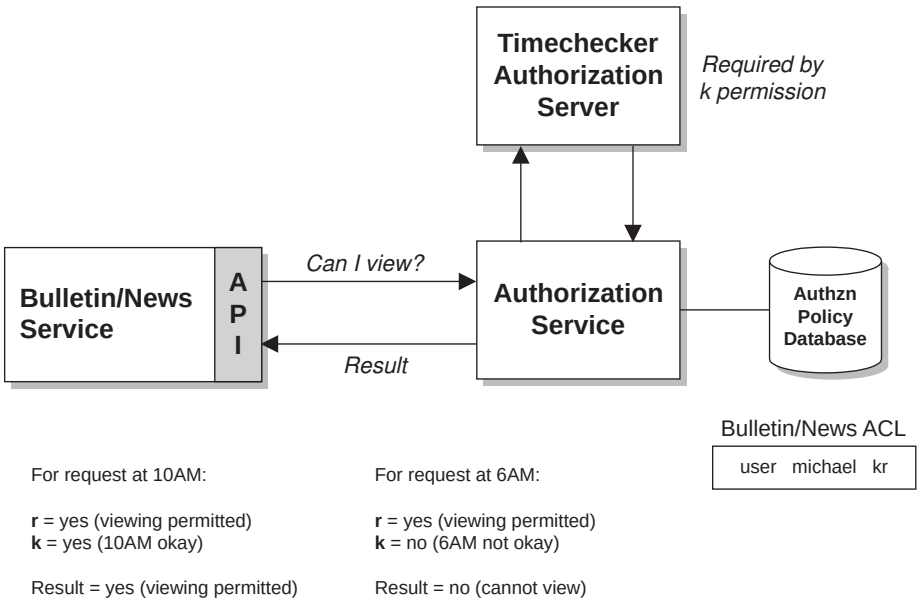
```
ivadmin> server delete /ExternAuthzn/timechecker
```

Example 1: A third-party bulletin board and news service has time restrictions placed on its operation. Users can view the information that is provided by this service only between the hours of 8AM and 5PM. An external authorization service is written to perform a time check on all requests that are made to the bulletin board and the news service.

Use the **ivadmin** command to set up the external authorization service.

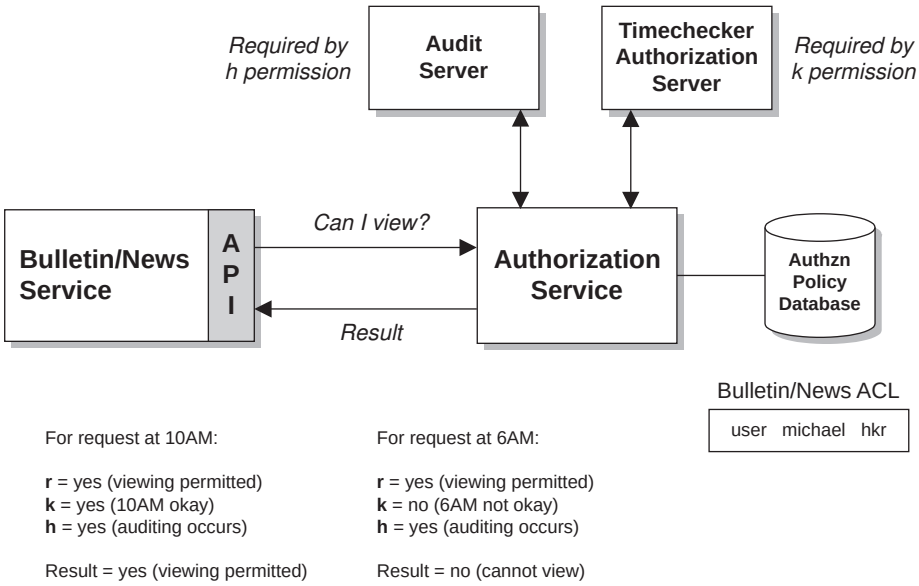
The following figure illustrates the possible scenarios for the authorization processing. The user must have the read (r) permission to view the bulletin board and the news information. The ACL on the news service also includes the time-check (k) permission. The time-check (k) permission dictates to the Policy Director Authorization Service that it include the timechecker authorization server in the final decision.

Policy Director bases the final authorization decision on the summation of all authorization server decisions.



Example 2: This example is the same as Example 1. However, this example adds a second external authorization service that audits activity on the bulletin board and the news service.

Note that, when the timechecker authorization service does not permit viewing, auditing of the activity still occurs. The presence of the **h** permission requires the involvement of the auditing authorization server during the ACL check.



Setting the number of update notifier threads

The Policy Director Management server (ivmgrd) manages the primary (master) authorization policy database. It also maintains location information about other WebSEAL and NetSEAL servers in the secure domain. The Management server typically requires very little administration or configuration. This section includes tasks available to the administrator.

The Management server (ivmgrd) is responsible for maintaining the primary authorization policy database. The Security Manager (secmgrd) and the Authorization Server (ivacld) are responsible for creating replicas of this primary database.

The Management server is then responsible for synchronizing all database replicas in the secure domain. When a change is made to the primary 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 primary database.

The Management server configuration file, ivmgrd.conf, contains a parameter 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, set the thread pool 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 parameter.

The performance goal of the update-notifier-threads value is to announce a database change as quickly as possible. Generally set the value to equal the number of existing replicas. Setting this value results in the performance advantage of a single pool of threads, which are quickly accomplishing the notification task to all replicas at once.

The default event-notifier-thread pool is set as:

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

Chapter 12. Logging and auditing server activity

Policy Director provides a number of logging and auditing capabilities. There are log files that capture any error messages and any warning messages that are generated by both Policy Director and DCE servers. There are also audit trail files that monitor Policy Director and DCE server activity.

This chapter includes:
- An introduction to logging and auditing.
- An explanation of each log file.
- An explanation of each audit file.

Logging and auditing overview

The contents of log and audit trail files can be a useful source of information. You can monitor or find and fix problems with the Policy Director and DCE servers activities using the log and audit trail file contents.

Log files

The Policy Director and DCE servers use log files to store warning and error messages. All log files are in text format.

Policy Director provides the following log files:
- Policy Director server log files

  See “Policy Director server log files” on page 144.
- DCE server log files

  See “DCE server log files” on page 145.
- DCE serviceability messages

  See “DCE serviceability messages” on page 145.
- Standard HTTP log files

  See “Standard HTTP logging” on page 146.

Audit trail files

The Policy Director and DCE servers use audit trail files to store records of server activity. A *record* refers to the output of a specific server event. An audit trail is a collection of multiple records that document the server activity. Most audit files are in ASCII format. DCE audit trail files are in binary format. You must use the **dcecp** utility to view these files.

The following audit trail files provide event information for Policy Director or DCE servers:
- These three Policy Director authorization audit trail files (audit.log):
  - Management server (ivmgrd)
  - Security Manager (secmgrd)
  - Authorization server (ivacld)

  See “Policy Director authorization audit trail files” on page 149.
- WebSEAL Audit Trail File (wand_audit_log)

  See “WebSEAL audit trail file” on page 151.

- Policy Director Management Audit Trail File
  See “Policy Director management command audit trail file” on page 153.
- DCE audit trail files
  See “DCE server audit trail files” on page 154.

Convention for the install-path variable

The *install-path* variable used throughout this chapter has the following interpretations, according to operating system platform:
UNIX: `/opt/intraverse/`
Windows:

```
C:\Program Files\IBM\
```

You cannot change this path name in UNIX because it is fixed.

The Windows platform allows you to define the *install-path* during the installation of the Policy Director software.

Policy Director server log files

Each Policy Director server dynamically generates warning and error messages that are directed to standard error and that are then redirected to specific log files.

Server	Process	Log File Location
Management server	**ivmgrd**	Defined in ivmgrd.conf: log-file=*install-path*/ivmgrd/log/ivmgrd.log
Security Manager	**secmgrd**	Defined in secmgrd.conf: log-file=*install-path*/secmgr/log/secmgrd.log
Authorization server	**ivacld**	Defined in ivacld.conf: log-file=*install-path*/ivacld/log/ivacld.log
Directory Services Broker	**nsid**	*install-path*/broker/nsid.log

Enabling and disabling server log files

Policy Director enables logging when there is a log file that is defined in the configuration file.

Example secmgrd.log

The secmgrd.log file contains content similar to the following:

```
1998-09-22-21:56:36.898-04:00I----- secmgrd FATAL ivc general
exec.c 344 0x00000006
Caught signal (15)
1998-09-22-21:56:37.309-04:00I----- secmgrd ERROR ivc rpc
IVServer.cpp 1039 0x00000001
Could not unexport bindings from name service
(/.:/subsys/ibm/secmgr/server/sun,0x16c9a093
1998-09-22-21:56:37.354-04:00I----- secmgrd ERROR ivc rpc
IVServer.cpp 1048 0x00000001
Could not unregister RPC endpoints (0x16c9a042)
```

DCE server log files

Each DCE server dynamically generates warning and error messages that are directed to standard error and that are then redirected to specific log files. These log files can be useful sources of information when finding and correcting a problem.

DCE server log files include:

Security server:

UNIX: `/opt/dcelocal/var/security/secd.log`

Windows: `\Program Files\IBM\dcelocal\var\security\secd.log`

DCE server:

UNIX: `/opt/dcelocal/var/dced/dced.log`

Windows: `\Program Files\IBM\dcelocal\var\dced\dced.log`

DCE serviceability messages

The routing file controls DCE serviceability messages:

UNIX: `/opt/dcelocal/var/svc/routing`

Windows: `\Program Files\IBM\NetSEAT\var\svc\routing`

Note: For Windows systems, the installation path is configurable during installation: `\Program Files\IBM\NetSEAT\` The environment variable (%NETSEAT%) is resolved into the configured path.

Default entries in the routing file

Entries in this configuration file determine the type of information that is logged. The *routing* file includes the following default entries:

UNIX:

```
FATAL:STDERR:-;FILE:/opt/dcelocal/var/svc/fatal.log
ERROR:STDERR:-;FILE:/opt/dcelocal/var/svc/error.log
WARNING:STDERR:-;FILE:/opt/dcelocal/var/svc/warning.log
```

Windows:

```
FATAL:STDERR:-;FILE:%NETSEAT%\var\svc\fatal.log
ERROR:STDERR:-;FILE:%NETSEAT%\var\svc\error.log
WARNING:STDERR:-;FILE:%NETSEAT%\var\svc\warning.log
```

NOTICE messages provide extra information about server activity. By default, Policy Director does not enable (no entry exists in the file) NOTICE messages.

To enable NOTICE messages (and direct them to standard error), add the following new NOTICE line to the end of the routing file:

UNIX:

```
FATAL:STDERR:-;FILE:/opt/dcelocal/var/svc/fatal.log
ERROR:STDERR:-;FILE:/opt/dcelocal/var/svc/error.log
WARNING:STDERR:-;FILE:/opt/dcelocal/var/svc/warning.log
NOTICE:STDERR:-;FILE:/opt/dcelocal/var/svc/notice.log
```

Windows:

```
FATAL:STDERR:-;FILE:%NETSEAT%\var\svc\fatal.log
ERROR:STDERR:-;FILE:%NETSEAT%\var\svc\error.log
WARNING:STDERR:-;FILE:%NETSEAT%\var\svc\warning.log
NOTICE:STDERR:-;FILE:%NETSEAT%\var\svc\notice.log
```

Debug mode for directing messages to standard output

Normally Policy Director redirects warning and error messages, which include NOTICE messages, to the appropriate log files.

To direct these messages to standard output (terminal), use the **-debug** command option when starting a server. This option causes the server to run in the foreground (that is, the server does not daemonize itself). Policy Director writes warning and error messages to standard output.

For example, to start the Security Manager (secmgrd) in debug mode, use the following command:

```
# /opt/intraverse/secmgr/bin/secmgrd -debug
```

You can also use the UNIX **tee** command to capture the server output to a single file.

The following example illustrates starting the Policy Director Security Manager in this mode:

```
# secmgrd -debug 2>&1 | tee /tmp/secmgrd.log
```

Debug notes

When debugging, keep the following in mind:

1. When you have completed gathering server activity information, be sure to restore the routing file to its normal condition. Remove the NOTICE entry. NOTICE generates a large amount of information which can rapidly accumulate.
2. You can use **Ctrl + c** to interrupt a server process started in debug mode. The server process shuts down correctly and exit.

Standard HTTP logging

The Policy Director WebSEAL server also maintains three conventional HTTP log files that record activity rather than messages:

wand_request_log	See “Displaying wand_request_log” on page 148.
wand_agent_log	See “Displaying wand_agent_log” on page 149.
wand_referer_log	See “Displaying wand_referer_log” on page 149.

By default, Policy Director maintains these log files under the following directory:

UNIX: `/opt/intraverse/www/log`

Windows: `\Program Files\IBM\Policy Director\www\log`

Configuring standard HTTP logging

The `[wand]` stanza of the iv.conf configuration file contains parameters for configuring standard HTTP logging.

The following table illustrates the relationship between the HTTP log files and the configuration file parameters:

Log Files	Location Parameter	Enable/Disable Parameter (= yes or no)
wand_request_log	reqlog =	logreqs =
wand_referer_log	reflog =	logrefs =
wand_agent_log	agentlog =	logagents =

For example, the entry in iv.conf for the default location of the **wand_request_log** appears as follows:

```
reqlog = log/wand_request_log
```

The root directory for this location is:

UNIX: `/opt/intraverse/www/`

Windows: `\Program Files\IBM\Policy Director\www\`

Enabling and disabling HTTP logging

By default, Policy Director enables all HTTP logging:

```
[wand]
logreqs = yes
logrefs = yes
logagents = yes
```

To disable logging, set:

```
<enable-parameter> = no
```

Specifying the timestamp type

You can choose to have the timestamps in each log that is recorded in Greenwich mean time (GMT) instead of the local time zone. By default, Policy Director uses the local time zone:

```
[wand]
loggmttime = no
```

To use GMT timestamps, set:

```
loggmttime = yes
```

Note: Consider keeping all log files in synchronized time easier reading of audit and log files from all the security-related products.

Specifying the maximum log file size

The maximum size of the each HTTP log files is set by default:

```
[wand]
logsize = 2000000
```

Policy Director backs up the log file when a log reaches this size.

Note that this parameter also affects the Policy Director **wand_audit_log** audit trail file.

Check the size of the log files often to be sure they are not growing too large and occupying too much space. Archive the log files periodically during regular system maintenance.

Using HTTP common log format

Every response (success or failure) that is sent back by the Policy Director server is recorded with a one-line entry in the following HTTP Common Log Format:

```
host - authuser [date] request status bytes
```

Where:

host	Specifies the Internet Protocol (IP) address of the requesting machine.
authuser	Takes the value of the **From:** header of the received HTTP request. In addition, this field also conveys a secure RPC request by setting the value to `dce-rpc`. This field is blank for an unauthenticated user.
date	Specifies the date and time of the request.
request	Specifies the first line of the request as it came from the client.
status	Specifies the HTTP status code sent back to the requesting machine.
bytes	Specifies the number of bytes sent back to the requesting machine. In other words, the content length of the document is transferred.

Displaying wand_request_log

The wand_request_log records standard logging of HTTP requests. Examples of standard logging include information on URLs that have been requested and information on the client (for example, IP address) that made the request.

The wand_request_log file contains content similar to the following:

```
130.105.1.90 - - [Tue, 23 Apr 1996 17:23:33 EDT]
  "GET /~smith/private_html/ HTTP/1.0" 403 77
130.105.1.90 - - [Tue, 23 Apr 1996 17:23:47 EDT]
  "GET /icons HTTP/1.0" 302 93
130.105.1.90 - - [Tue, 23 Apr 1996 17:23:59 EDT]
  "GET /icons/ HTTP/1.0" 403 77
130.105.1.90 - - [Tue,   23 Apr 1996 17:24:04 EDT]
  "GET /~smith/private_html/ HTTP/1.0" 403 77
130.105.1.90 - - [Tue, 23 Apr 1996 17:24:11 EDT]
  "GET /~smith/ HTTP/1.0" 403 77
dce-rpc - - [Tue, 23 Apr 1996 17:24:51 EDT]
  "GET / HTTP/1.0" 200 919
```

Displaying wand_agent_log

The wand_agent_log records the contents of the `User_Agent:` header in the HTTP request. This log reveals information about the client browser, such as architecture or version number, for each request.

The following example shows a sample version of a wand_agent_log file:

```
Mozilla/4.01 [en] (WinNT; U)
Mozilla/4.01 [en] (WinNT; U)
Mozilla/4.01 [en] (WinNT; U)
Mozilla/4.01 [en] (WinNT; U)
```

Displaying wand_referer_log

The wand_referer_log records the header of the HTTP request. For each request, the log records the document that contained the link to the requested document.

The log uses the following format:

```
referer -> object
```

This information is useful for tracking external links to documents in your Web space. The log reveals that the source indicated by `referer` contains a link to a page *object*. This log allows you to track stale links and to find out who is creating links to your documents.

The following example shows a sample version of a wand_referer_log file:

```
http://manuel/maybam/index.html -> /pics/ibm_logo.gif
http://manuel/maybam/ivdl/index.html -> /pics/ibm_logo.gif
http://manuel/maybam/ -> /ivdl/index.html
http://manuel/maybam/ -> /ivdl/index.html
http://manuel/maybam/ivdl/index.html -> /pics/ibm_logo.gif
http://manuel/maybam/ -> /ivdl/index.html
```

Policy Director authorization audit trail files

Each Policy Director server can capture audit events whenever any security-related auditable activity occurs. Policy Director saves audit events as audit records that document the specific activity of that server. Multiple audit records make up an audit trail file.

The following table illustrates the relationship between each Policy Director server and its associated audit trail file:

Server	Process	Authorization Audit File
Management server	**ivmgrd**	Defined in ivmgrd.conf: audit-file= *install-path*/ivmgrd/log/audit.log
Security Manager	**secmgrd**	Defined in secmgrd.conf: authzn-audit-file= *install-path*/secmgr/log/audit.log
Authorization server	**ivacld**	Defined in ivacld.conf: audit-file= *install-path*/ivacld/log/audit.log

Policy Director writes Authorization information to the appropriate audit trail file whenever you set the audit (A) permission for a user or group in an ACL. The resulting audit records include all access attempts, including authorization failure.

Audit trail administration

The audit (A) permission in an ACL entry triggers the Policy Director authorization audit trail files record activity information. The activation of auditing through the audit (A) permission is easy to accomplish.

The following conditions apply to managing authorization audit trail files:

- The object where the ACL is attached determines which file of the three audit.log files collects the data.

  For example, you can attach the ACL that contains the audit (A) permission on one or more entries to the /Management object of the protected object namespace. If attached, data will be collected in the audit.log file for the Management server (ivmgrd). The Management server controls the authorization policy database (ACL) and database replication (Replica).
- Only activity information is collected for users, groups, or both when you set the audit (A) permission in the appropriate ACL entry.

  For example, you can give the unauthenticated entry the audit (A) permission in an ACL that is attached to an HTML page object. With this permission, the Security Manager's audit.log file collects information on all attempts to access the object that were not allowed.

Example: The following ACL represents the default-webseal ACL. The `cell_admin` user entry and the `authenticated` entry have audit (A) permissions set.

```
user cell_admin                 aAbcTdmlrx
group iv-admin                  abdTdmlrx
group ivmgrd-servers            Tl
group webseal-servers           gTdmlrx
any-authenticated               Tr
unauthenticated                 ATr
```

An ACL attached to the /WebSEAL object means that it is attached to the root of the WebSEAL region of the protected object namespace. If attached, Policy Director will record the activity that involves the Security Manager server (WebSEAL and NetSEAL) in the Security Manager audit.log file.

The WebSEAL namespace might not contain other explicit ACLs that change permission conditions, If there are no other explicit ACLs, auditing will occur on all requests to any object within the Web space.

The audit trail file only records activity that is initiated by the cell_admin user and any unauthenticated access attempts.

An explicit ACL attached to an object somewhere below the /WebSEAL object will break the chain of ACL inheritance. The entries in this explicit ACL might not contain any audit permissions, If the entries do not contain any audit permissions, no audit trail will be generated for this object or for any other object below this point.

Note: You must remember to add the audit permission to the relevant entries of any ACL that you are explicitly attaching to objects below the /WebSEAL object.

Example Management server audit trail file

A Management server audit trail file contains content similar to the following:

```
START RECORD
  Protected object: /WebSEAL
  Requested permissions: 0x00000100
  Principals:
principal 0: DCE principal 00000064-35ee-21d2-a000-0800207b48c5
  quality of protection: none    result: authorized
END RECORD

START RECORD
  Protected object: /WebSEAL/sun
  Requested permissions: 0x00000100
  Principals:
principal 0: DCE principal 00000064-35ee-21d2-a000-0800207b48c5
  quality of protection: none    result: authorize
END RECORD

START RECORD
  Protected object: /WebSEAL/sun/icons
  Requested permissions: 0x00000100
  Principals:
principal 0: DCE principal 00000064-35ee-21d2-a000-0800207b48c5
  quality of protection: none    result: authorize
END RECORD

START RECORD
  Protected object: /WebSEAL
  Requested permissions: 0x00000100
  Principals:
principal 0: DCE principal 00000064-35ee-21d2-a000-0800207b48c5
  quality of protection: none    result: authorize
END RECORD
```

WebSEAL audit trail file

You can also monitor Policy Director WebSEAL server activity. Policy Director saves audit events as audit records that document the specific activity of that server. Multiple audit records make up an audit trail file.

WebSEAL auditing

Parameters for configuring WebSEAL audit trail files are located in the `[wand]` stanza of the iv.conf configuration file.

The following table illustrates the relationship between WebSEAL and the audit trail file:

Server	Audit File
WebSEAL	Defined in iv.conf: auditlog= *install-path*`/www/log/wand_audit_log`

Enabling and disabling WebSEAL auditing

By default, WebSEAL auditing is disabled:

```
[wand]
logaudit = no
```

To turn on auditing, set:

```
logaudit = yes
```

Note: There must be no space after the `yes` or `no` when editing this parameter in the iv.conf configuration file.

Specifying the log file location

The default location of the WebSEAL auditing file is:

```
[wand]
auditlog = log/wand-audit-log
```

Specifying the maximum log file size

Policy Director sets a default maximum size of the auditing log file:

```
[wand]
logsize = 2000000
```

When a log reaches this size, Policy Director copies the log file as a backup copy. A new and empty auditing log file becomes the default auditing log file. Note that this parameter also affects the following standard HTTP log files:

- wand_request_log
- wand_referer_log
- wand_agent_log

WebSEAL audit trail file syntax

Each response (success or failure) that is sent back by the WebSEAL server is recorded with a one-line entry in the following format:

```
host call_type uri iv_status_code [date] uuid group_uuid_list
```

host (and endpoint)	IP address and endpoint information of the remote host. If there is no endpoint information, display [-].
call_type	`0` for TCP connection, `1` for UNAUTH RPC, and `2` for AUTH RPC
uri	Universal Request Indicator for the request.
iv_status_code	Status code for this Policy Director subset of the standard auditing facility.
date	Date and time of the request.
uuid (indicated with **-p** flag)	UUID for the client. If there is no UUID information, then display nothing.
group_uuid_list (indicated with **-g** flag)	List of group UUIDs. If there is no group UUID information, then display nothing.

Example audit trail file contents

The audit trail file contains content similar to the following:

```
204.30.81.188[33380] 2 /audit_report.html 0x18a2141a
[21/Aug/1997:14:36:23 -0700]
-p 00000064-0f4c-21d1-9300-00c078500371
-g 0000000c-0f4c-21d1-9301-00c078500371
-g 0000044c-0f4c-21d1-8601-00c078500371
-g 0000044d-0f4c-21d1-8601-00c078500371
```

Details:

host:	204.30.81.188
endpoint:	[33380]
call_type:	2
uri:	/audit_report.html
iv_status_code:	0x18a2141a
date:	[21/Aug/1997:14:36:23 -0700]
uuid:	-p 00000064-0f4c-21d1-9300-00c078500371
group_uuid_list:	-g 0000000c-0f4c-21d1-9301-00c078500371 -g 0000044c-0f4c-21d1-8601-00c078500371 -g 0000044d-0f4c-21d1-8601-00c078500371

Note: It is possible that the URI string might appear only as a hyphen. The causes for this situation could include an early ending of the request or a malformed request string.

Policy Director management command audit trail file

Each Policy Director server can capture audit events whenever any management related auditable activity occurs. Policy Director saves audit events as audit records that document the specific activity of that server. Multiple audit records make up an audit trail file.

The following table illustrates the relationship between each Policy Director server and its associated audit trail file:

Server	Process	Management Audit File
Management server	**ivmgrd**	Defined in ivmgrd.conf: mgr-audit-file= *install-path*/ivmgrd/log/mgraudit.log

The responsibilities of the Management server include maintaining the primary authorization policy database.

This database includes the description of the protected object namespace for the secure domain, the ACL policy templates, and where the ACLs are attached to objects.

This database includes:

- The description of the protected object namespace for the secure domain.
- The ACL policy templates,
- The information about where the ACLs are attached to objects.

You can capture any management command event, from the Management Console, or by using the **ivadmin** utility, in the mgraudit.log file.

Audit record contents

The audit records are written in records that are tagged with XML-style bracketing. An audit event captures the following information:

Originator ID
Derived from the incoming RPC client handle that is printed as a list of UUIDs or the string `unauthenticated`, as appropriate.

Tag: `P`

Event ID
A number which uniquely identifies a management command, defined in the ../ivmgrd/cmdConst.h header.

Tag: `I`

Command outcome
A number corresponding to the status code returned to the caller.

Tag: `O`

Time stamp
A record of the time the command was completed in the same format that is currently used by the ACL bit auditing.

Tag: `D`

Command argument vector
A representation of the command input arguments.

Tags: `V` and `A`

Example of Management server audit trail file

The Management server audit trail file contains content similar to the following:

```
<E><D>Fri May 30 00:00:00 1999<\D><I>3008</I><O>0</O><P>[1]
069d9fb6-943e-11cd-a35c-0000c08adf56</P><V><A> argument
1</A><A>argument 2</A></V></E>
```

DCE server audit trail files

The following DCE server audit trail files use the DCE auditing service. The file formats are binary. You must use the **dcecp** utility to view the files.

1. DCE Security Service (secd) Audit Trail
   /opt/dcelocal/var/security/sec_audit_trail
   /opt/dcelocal/var/security/sec_audit_trail.md_index
2. DCE Audit Service (auditd) Audit Trail
   /opt/dcelocal/var/security/central_trail
   /opt/dcelocal/var/security/central_trail.md_index
3. DCE Time Service (dtsd) Audit Trail
   /opt/dcelocal/var/security/dts_aud_trail
   /opt/dcelocal/var/security/dts_aud_trail.md_index

sec_audit_trail example:

```
dcecp> login cell_admin
Enter Password:
dcecp>

--- Event Record number 261 ---
o Event Information:
   - Event Number:          0x101 /* 257 */
   - Event Name:          AS_Request
   - Event Outcome:          success
o Server:          /.:/hosts/eggman
o Client:          /.../eggman_cell/cell_admin
o Number of groups:          0
o Authorization Status:          Authorized with a name
o Date and Time recorded: 1998-10-20-10:42:56.248-04:00I-----
--- End of Event record number 261 ---
```

Chapter 13. WebSEAL: Setting up authentication

Policy Director WebSEAL supports LDAP, Kerberos, and public and private key authentication mechanisms. An important by-product of the authentication process is the acquisition of user credentials. User credentials are used during the authorization of requests for access to protected resources.

This chapter includes:

Introducing WebSEAL authentication

This section discusses WebSEAL support for:

- Secure communication over the SSL protocol.
- Authentication mechanisms.
- Methods of providing identity information.
- Extension of the standard authentication mechanisms.

SSL support

WebSEAL supports secure communication over the secure socket layer (SSL) protocol. These sections discuss secure communication over the SSL protocol:

- "Configuring WebSEAL for SSL" on page 158.
- "Setting up a server-side certificate for WebSEAL" on page 161.

Authentication mechanisms

Authentication is the process of identifying an individual that is attempting to log in to a secure domain. WebSEAL supports the following authentication mechanisms:

- LDAP secret key
- Kerberos Version 5
- Public and private key

Client identity information

The authentication process requires the client to submit some form of identity information during login. WebSEAL supports the following methods of providing this identity information:

1. Username and password (that are used by LDAP and Kerberos)
   - Basic authentication
   - Forms-based login

   See "Using username and password authentication" on page 166.
2. Client-side X.509 certificate (used by public/private key)

   See "Using X.509 certificate authentication" on page 170.

Credentials acquisition

You can use a credentials acquisition service to extend the standard authentication mechanisms that are supported by WebSEAL. See “Policy Director Credentials Acquisition Service” on page 172.

Configuring WebSEAL for SSL

The Policy Director WebSEAL server supports secure communication with client browsers that use the secure socket layer protocol (SSL).

When using this protocol, clients can use one of two methods to pass identity information to WebSEAL:

- Username and password
- Client-side X.509 digital certificate

In both modes, WebSEAL authenticates itself to the client by using its server-side digital certificate. A Certificate Authority (CA) issues this X.509 certificate. Policy Director stores the certificate and the associated private key in either a PEM formatted file or a PKCS#12 formatted file.

When using the PEM format, Policy Director stores in separate files the server’s private key and the signed public key. When using PKCS#12 format, the key pairs that are generated and stored reside together in a single file.

It is recommended that the common name (CN) in the server certificate be the same as the fully qualified host name of the WebSEAL server.

Although not required, most client browsers verify that a valid Certificate Authority has issued a server certificate. The verification is done through the root CA certificate database. If the signer of the certificate does not match an entry in the root CA certificate database, a warning will appear. The user has the responsibility to accept or reject the connection with that server.

See “Setting up a server-side certificate for WebSEAL” on page 161 for complete information for obtaining and installing a server-side X.509 certificate for WebSEAL.

In client-side certificate mode, WebSEAL authenticates itself to the client by using its server-side digital certificate, as described above. Additionally, WebSEAL requires an X.509 root CA certificate from a CA that can validate the client-side certificate. A client request, made by using the client-side certificate, provides true mutual authentication.

Using server-side certificates and root CA certificates

A server-side X.509 certificate identifies a specific WebSEAL server to a client. Policy Director stores server-side certificates in PEM or PKCS#12 format, as follows:

- PEM format—separate files that store the server’s private key and the signed public key
- PKCS#12 format—a single file that contains the server’s private and public keys.

Note: WebSEAL can contain and support only one server certificate at a time. WebSEAL does not support multiple logical Web server instances on the same machine.

A *root CA certificate* identifies a specific Certificate Authority (CA). WebSEAL requires the root CA certificate to validate a client-side certificate. WebSEAL can maintain a list of CA root certificates in either PEM format, PKCS#12 format, or a combination of both, as follows:

- PEM format—accumulates root certificates in a single file
- PKCS#12 format—stores root certificates as separate files under a common directory

Storing certificates

The secmgrd.conf configuration file defines certificate storage parameters. The parameters are different, depending on whether the parameter you use is for UNIX or for Windows.

Parameter	Description
For UNIX: `ca-directory = /opt/intraverse/lib/certs` **For Windows:** `ca-directory = C:\Program Files\ibm\Policy Director\lib\certs`	
	Base directory for certificate storage.
For UNIX: `ca-cert-file = /lib/certs/cacert.pem` **For Windows:** `ca-cert-file = C:\Program Files\ibm\Policy Director\lib\certs\cacert.pem`	
	The X.509 root CA certificate of a recognized Certificate Authority in PEM format. WebSEAL accepts client-side X.509 certificates from a trusted CA in PEM format. You can append root certificates from other CAs to this file.
For UNIX: `ca-cert-p12-dir = /opt/intraverse/lib/certs/ca_p12` **For Windows:** `ca-cert-p12-dir = C:\Program Files\ibm\Policy Director\lib\certs\ca_p12`	
	The designated directory for a file that contains the X.509 root certificate of a recognized Certificate Authority in PKCS#12 format. WebSEAL accepts client-side X.509 certificates from a trusted CA in PKCS#12 format. You can store root certificate files from other CAs in this directory.
For UNIX: `certificate-file = /opt/intraverse/lib/certs/svrcert.pem` **For Windows:** `certificate-file = C:\Program Files\ibm\Policy Director\lib\certs\svrcert.pem`	
	The X.509 server certificate obtained from the CA in PEM format. This certificate is presented to SSL clients. The sample certificate contained in this file should be replaced by a legitimate certificate from a trusted CA.
For UNIX: `key-file = /opt/intraverse/lib/certs/srvkey.pem` **For Windows:** `key-file = C:\Program Files\ibm\Policy Director\lib\certs\srvkey.pem`	
	The server private key in PEM format. The sample key contained in this file at installation should be replaced by a legitimate key that you generate.

For UNIX: `certificate-file = /opt/intraverse/lib/certs/svrcert.p12` **For Windows:** `certificate-file = C:\Program Files\ibm\Policy Director\lib\certs\svrcert.p12`	
	The X.509 server certificate obtained from the CA in PKCS#12 format. The file includes the private key. The sample certificate and key contained in this file should be replaced by a legitimate certificate and key from a trusted CA.
UNIX and Windows: **pass-key =***passphrase*	
	The key password (*passphrase*) used to unlock the private key file.

Configuring certificate handling

The `[wand]` stanza of the iv.conf configuration file contains the parameter for handling client-side X.509 certificates. You can specify how WebSEAL is to handle client-side X.509 certificates by setting the **verify-clients** parameter. The allowed values for verify-clients include:

Value	Description
never	Do not request X.509 certificates from clients. Force clients to access using user name and password.
optional	Ask clients for an X.509 certificate, and use certificate-based authentication, when supplied. When client does not present a certificate, force clients to use basic authentication.
required	Ask clients for an X.509 certificate, and use certificate-based authentication. When client does not present a certificate, do not allow a connection.

By default, WebSEAL does not request client-side certificates:

```
[wand]
verify-clients = never
```

Setting SSL session cache timeout

The `[ssl]` stanza of the secmgrd.conf configuration file contains the parameter for setting the static SSL session cache timeout.

WebSEAL caches credential information internally. This credentials expiration parameter dictates the length of time authorization-credential information remains in memory on WebSEAL.

The parameter is not an inactivity timeout. The value maps into a “credential lifetime” rather than a ”credential timeout.” Its purpose is to enhance security by forcing the user to re-authenticate when Policy Director reaches the specified time-out limit.

The default cache timeout (in seconds) is:

```
[ssl]
ssl-cache-timeout = 3600
```

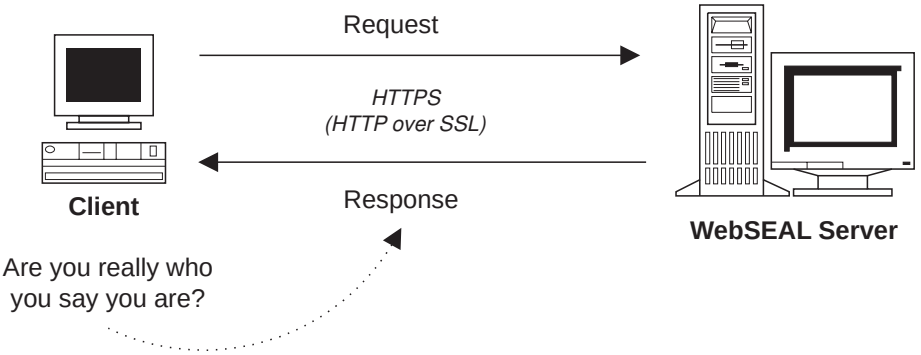
Adjust this value to balance server performance against user convenience, depending on the volume of SSL requests that the server must handle.

Note: Some browsers perform automatic session renegotiation. If this is the case for your browser, this parameter will be ineffective.

Setting up a server-side certificate for WebSEAL

You can set up a Policy Director WebSEAL server to allow SSL-enabled clients to verify the authenticity of the server. This section walks you through the administrative tasks that are required if you were setting up a server-side certificates in PEM format.

Specifically, the task involves registering with a valid CA or with an internally controlled, certificate-generating product. The registration must obtain a site server certificate that allows Policy Director to properly accept and respond to requests from SSL-enabled browsers.

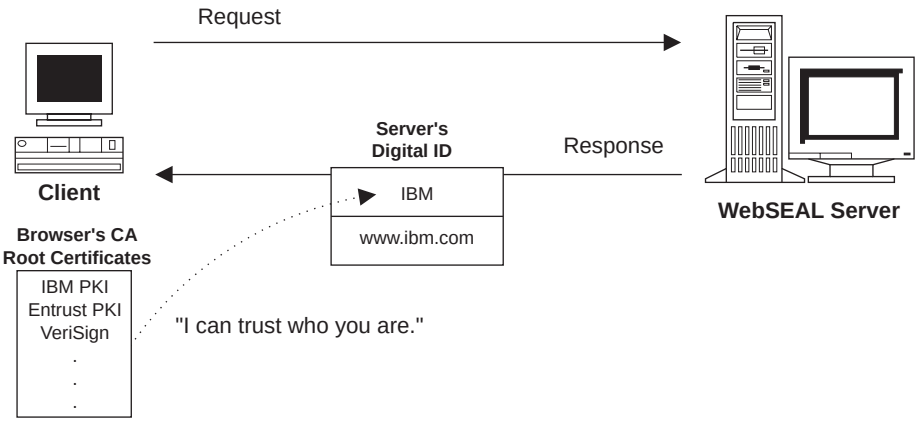


Ensuring secure communication over SSL

The Policy Director WebSEAL server supports authentication of clients accessing using HTTP over SSL (HTTPS). WebSEAL must have an installed server-side public X.509 certificate that it uses to respond to these clients. The X.509 certificate proves to the client that the response from WebSEAL is coming from an allowed server.

For secure SSL communication over the Internet, the browser must authenticate the server. The browser authenticates by checking the server's public key certificate against a matching root CA certificate. The matching CA certificates either comes bundled with, or is acquired by, the browser.

An allowed server certificate, signed by a CA, prevents the possibility of impersonation.



WebSEAL ships with a sample server certificate that is signed by a sample IBM certificate authority. This sample certificate allows WebSEAL to respond to an SSL-enabled browser request. However, the browser cannot verify the sample certificate because the certificates contain no IBM root CA certificate. Therefore, it offers no true secure communication.

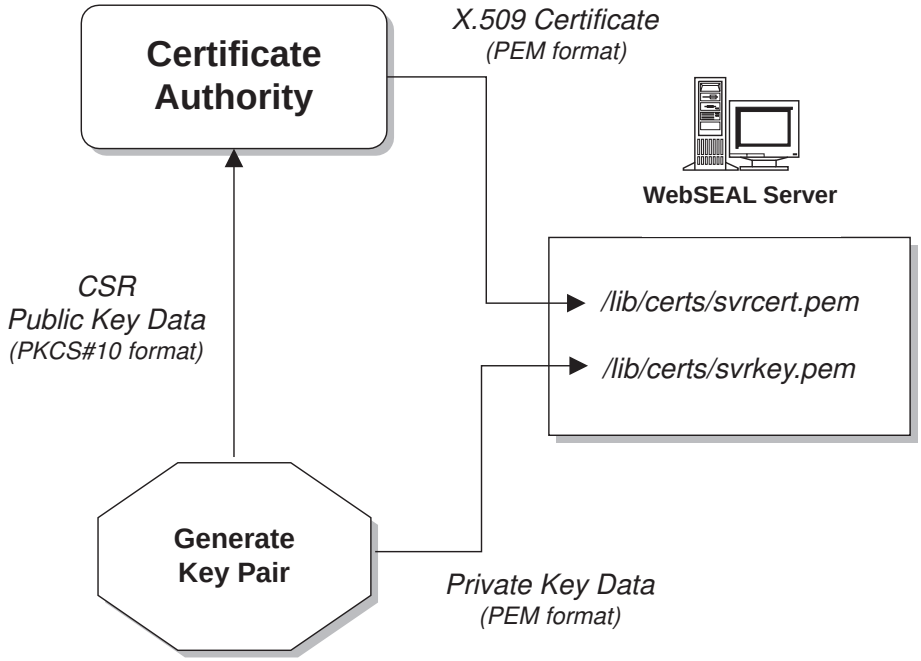
To ensure secure communication over SSL, it is very important to register for a site server certificate from a trusted Certificate Authority. You can obtain a site server certificate from a recognized CA, or generate you own “in-house” certificate using software, such as IBM SecureWay Trust Authority.

Setting up Policy Director for communication over SSL involves the following tasks:

- “Generating a public key and private key”.
- “Using the gencsr utility (optional)” on page 163 (optional).
- “Registering the CSR with the certificate authority” on page 164.
- “Installing the server certificate” on page 165.
- “Updating the Security Manager configuration file” on page 165.
- “Testing the new certificate installation” on page 165.

Generating a public key and private key

To obtain a site server certificate from a CA, you must first generate a public/private key pair for your server.



You keep the private key portion.

The public key portion, containing user identity information, is known as the Certificate Signing Request (CSR). The CSR is the information you must send to the CA when you register for a site server certificate. The CA uses this information to construct your X.509 server-side certificate, used to respond to SSL-enabled clients.

You must store your private key and the server-side X.509 certificate from the CA in specially designated locations. The Policy Director secmgrd.conf configuration file defines these locations.

To generate a public and private key pair, use the generating tool and instructions provided by the CA. Policy Director provides a utility (**gencsr**) that you can use when no other utility is available. “Using the gencsr utility (optional)” describes the task of generating a key pair with **gencsr**.

Using the gencsr utility (optional)

Policy Director includes an optional utility, the **gencsr** utility that generates a public and private key pair. The utility is part of the Policy Director installation and is in the /bin directory:

UNIX: *install-path*/bin/gencsr

Windows: *install-path*\bin\gencsr

PKCS#10 format

The **gencsr** utility generates the key pair. This utility writes the private key information to a file in PEM format. This utility stores, in a file with the other certificate-signing request information, the public key. The utility stores the public key information in PKCS#10 format.

Public-Key Cryptography Standards (PKCS) describes the syntax for certification requests. A certification request consists of a distinguished name, a public key, and an optional set of attributes, collectively signed by the organization by requesting certification. A certificate signing request is sent to a Certificate Authority. Then, the CA generates a unique X.509 public-key certificate for your server.

gencsr utility command syntax

gencsr [-csrfile *csr_filename*] [-keyfile *key_filename*] [-keylen *key_length*] [-version]

Options	Description
-csrfile	Specifies public key (CSR) output to a file (PKCS#10 format). The specified file (*csr_filename*) holds the CSR in American National Standard Code for Information Interchange (ASCII) format. Default is standard output.
-keyfile	Specifies private key output (PEM format) to a file. Default is standard output.
-keylen	Specifies the key length (in bytes) of the public/private key pair. Default is 512.
-version	Displays the utility version number and copyright information.
-help	Displays command syntax and option descriptions.

Gencsr utility procedures

To use the Policy Director **gencsr** utility:

1. Start the **gencsr** utility with appropriate arguments for the names of the CSR file, the private key file, and optionally, a key length:

   UNIX: `$ gencsr -csrfile filename -keyfile filename -keylen 1024`

   Windows: `gencsr -csrfile filename -keyfile filename -keylen 1024`

   You can use any file name for the public and private keys; you will rename them in a later step.

   Note: The default key length is 512 bytes.
2. The utility prompts for personal information, which includes the *PEM pass phrase*.

   You must remember this pass phrase. Store this pass phrase in the secmgrd.conf configuration file in a later step. The pass phrase provides protection for your private key.
3. The utility generates a CSR file and private key file. “Registering the CSR with the certificate authority” describes how to send the CSR to the Certificate Authority.
4. Back up the sample, private-key file that comes with Policy Director:

   UNIX: `# cp svrkey.pem svrkey.pem.orig`

   Windows: `copy svrkey.pem svrkey.pem.orig`
5. Store the newly generated private key file in this same directory and name it svrkey.pem:

   UNIX: `# cp newkey.txt svrkey.pem`

   Windows: `copy newkey.txt svrkey.pem`

Note: You must protect this private key. There should be only one instance of the private key, which is vital for verifying communication between client and server.

Registering the CSR with the certificate authority

To register the CSR with the certificate authority:

1. The Certificate Authority (CA) generally has an online registration form. Use a Web browser to fill out this form. The exact procedures vary, depending on the CA.
2. The registration form requires you to provide the CSR generated in “Generating a public key and private key” on page 162 or “Using the gencsr utility (optional)” on page 163. You can paste the contents of the CSR file into the form or e-mail the file.
3. The CA sends you the new X.509 server-side public certificate in PEM format. This could take several days.

WebSEAL requires the certificate be in PEM format. PEM encoding is a base64 transform that is applied to a binary certificate. PEM format is an ASCII file where the line lengths are limited to 64 characters per line. This ASCII file begins with:

```
-----BEGIN CERTIFICATE----
```

And ends with:

```
-----END CERTIFICATE----
```

Installing the server certificate

To install the server certificate:

1. Back up the sample server certificate file that comes with Policy Director:

   For PEM format:

   UNIX: `# cp svrcert.pem svrcert.pem.orig`

   Windows: `copy svrcert.pem svrcert.pem.orig`

2. Store the new server certificate file from the CA in this same directory and name it svrkey.pem (overwrite the old value):

   UNIX: `# cp newcert.txt svrcert.pem`

   Windows: `copy newcert.txt svrcert.pem`

Updating the Security Manager configuration file

Check and update, as necessary, the following entries in the secmgrd.conf configuration file:

`certificate-file =`	The path name of the file containing the PEM-formatted certificate received from the CA. Default: lib/certs/svrcert.pem
`key-file =`	The path name of the locally generated private key file. Default: lib/certs/svrkey.pem
`pass-key =`	The PEM passphrase used to protect the private key.

Change the `certificate-file` entries and the `key-file` entries only when you use file names other than the default file names that are listed.

Testing the new certificate installation

To test the new certificate installation:

1. Stop and restart Policy Director to begin using the new certificate:

   UNIX:

   ```
   # /etc/init.d/iv stop
   # /etc/init.d/iv start
   # /etc/init.d/iv status
   ```

   Windows: Use the Services Control Panel.

2. Ensure that the Security Manager (secmgrd) has started successfully.

   When secmgrd fails to start, check the following log file to see why it failed:

   UNIX: *install-path*`/secmgr/log/secmgrd.log`

   Windows: *install-path*`\secmgr\log\secmgrd.log`

   When you do not find a meaningful error message, manually start secmgrd in debug mode. Refer to “Debug mode for directing messages to standard output” on page 146. Also, you can find the most up-to-date information on correcting problems at the IBM SecureWay Policy Director Web site:

   `http://www.ibm.com/software/security/policy/library`

3. From a browser, connect to the server that uses HTTPS, and ensure that the browser accepts the server certificate.

   For example, the browser should already store the widely recognized VeriSign root certificate by default. Therefore, you should not receive any warning messages or dialog boxes that are displayed prior to the Policy Director login prompt.

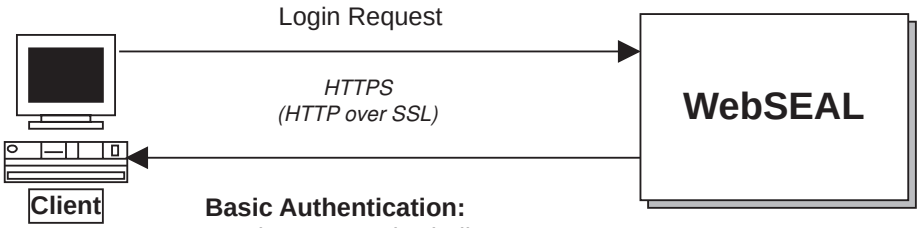
Warning messages will appear if you use the sample certificate that is shipped with Policy Director. The warning messages appear because the browser contains no IBM root certificate to verify the sample server certificate. These messages prompt you to accept or reject the server certificate. With no root certificate, the browser is unable to verify the legitimacy of the server certificate. The browser has to pass the responsibility of accepting or denying the certificate to you for this reason.

You now have a site server certificate validated by a trusted Certificate Authority in place. With the validated server certificate in place, SSL-enabled clients can successfully and securely authenticate your Policy Director WebSEAL server.

Using username and password authentication

Kerberos and LDAP Secret Key authentication mechanisms require client identity information in the form of a username and password. WebSEAL supports two methods of providing user names and passwords for authentication:

- "Basic authentication method".
- "Policy Director forms-based login method" on page 168.



Basic authentication method

WebSEAL supports the SSL protocol, used by Netscape Communicator/Netscape Navigator and Microsoft Internet Explorer (IE), to obtain user information such as user names and passwords. By convention, URLs that reflect use of a secure SSL connection start with **https:** instead of **http:**.

For a successful login, Policy Director requires the clients to use their Policy Director identity as recorded in the security registry. *Basic authentication* is a standard method for providing a username and password to the authentication mechanism.

In the first step, the server authenticates to the client by using its server-side certificate. If the client accepts the certificate, the server will issue a challenge to the client. The browser will produce a login prompt by requesting a username and password.

Note: The browser caches this login information. Standard basic authentication requires the username information and the password information for each subsequent request. The cached authentication information is sent to the user transparently.

Important points about basic authentication are:

- Policy Director uses SSL as the secure communication channel.
- Policy Director transmits user names and passwords across the secure SSL channel.

You might see a login prompt, which results from a basic authentication challenge, that is similar to:

```
Enter username for Policy Director [/.../www.ibm.com] at www.ibm.com

User Name:
Password:
```

Where you must enter the required information in the **User Name** and **Password** fields.

Basic authentication model

The process for the basic authentication models includes:

1. A client browser contacts the server by using SSL.
2. The server sends back its signed public key server certificate, obtained from a CA.
3. The client browser takes one of the following actions:
   - Finds an associated root CA certificate in its database and accepts the server's certificate.
   - Has no associated root CA certificate in its database and prompts the user with a warning. The user now has the responsibility to accept or reject the certificate.
4. When accepted, the server issues a challenge to the browser.
5. The browser responds by producing a login prompt, requesting a username and password.
6. After the user types in the username and password information, the browser sends the information to the Policy Director server.
7. Policy Director produces a credential when the username and password information matches existing information in the Policy Director user registry. Policy Director uses the credential to make authorization decisions. WebSEAL caches this credential for the duration of the SSL session.
8. The browser caches the user name information and the password information.

   Standard basic authentication requires this username and password for each subsequent browser login or request. This requirement is satisfied transparently using the cached authentication information.

   Note: Because the browser caches the username and password information in basic authentication, the **pkmslogout** command does not work correctly. To log out completely, you must close down the browser session. Use forms-based login when you require the **pkmslogout** feature.

Required administrative tasks

The administrator must perform the following tasks to prepare the WebSEAL server for SSL access in basic authentication mode:

- Install the server-side X.509 certificate on the WebSEAL server.
- Create a Policy Director account for each user who will participate in the secure domain.

Policy Director forms-based login method

Policy Director provides an alternative to the standard basic authentication mechanism: Policy Director *forms-based login*. This method results in an HTML login form from Policy Director instead of the standard login prompt resulting from a basic authentication challenge.

When you use forms-based login, the browser does not cache the username and password information as it caches in basic authentication. This allows the successful use of the special SSL session logout command, **pkmslogout**. Because Policy Director requires credential information only once (and caches it), there is no requirement to repeat login requests with each browser request.

Use the **https-forms-auth** parameter in the `[wand]` stanza of the iv.conf configuration file to accomplish forms-based login. This parameter can be set to yes or no. The default is no.

```
[wand]
https-forms-auth = no
```

Policy Director includes seven sample HTML forms. You can customize these HTML forms to contain site-specific messages or perform site-specific actions.

The `[wand]` stanza of the iv.conf configuration file defines the file locations for these forms under `SSL HTML Page Locations`.

The default directory location is:

UNIX: *install-path*`/www/lib/html/`

Windows: *install-path*`\www\lib\html\`

Form	Description
login.html	Request for user name and password
login_rep.html	Login error message
logout.html	Message for successful logout from the SSL session
passwd.html	Change password form
passwd_exp.html	Password expiration message
passwd_rep.html	Change password error message
help.html	Command reference

There are also two macros available for use in these pages. You can place these macro strings in the template files. The routine dynamically substitutes appropriate values.

Macro	Description
%USERNAME%	The name of the logged in user.
%ERROR%	The hard-coded error message returned from Policy Director.

Forms-based authentication model

The forms-based authentication model follows this process:

1. A client browser contacts the server by using SSL.
2. The server sends back its signed public key server certificate, obtained from a CA.
3. The client browser takes one of the following actions:
   - Finds an associated CA certificate in its database and accepts the server's certificate.
   - Has no associated CA certificate in its database and prompts the user with a warning. The user now has the responsibility to accept or reject the certificate.
4. When accepted by the client, WebSEAL prompts the client for a username and password by using a customized Policy Director HTML form.

   Policy Director uses this form to send username and password information back to WebSEAL.
5. Policy Director produces a credential when the username and password information matches existing information in the Policy Director user registry. Policy Director uses the credential in authorization decisions. WebSEAL caches this credential for the duration of the SSL session.

   Unlike basic authentication, the browser does *not* cache the username and password information. The **pkmslogout** command now works correctly.

Required administrative tasks

The administrator must perform the following tasks to prepare the WebSEAL server for SSL access in forms-based login mode:

1. Install the X.509 server-side CA certificate on the WebSEAL server.
2. Create a Policy Director account for each user who will participate in the secure domain.
3. Customize the Policy Director forms and define their location in the iv.conf configuration file.

Commands for username and password methods

Policy Director provides the following commands for supporting SSL-enabled clients authenticating using a username and password method:

- pkmslogout
- pkmspasswd

pkmslogout

Use the pkmslogout command to log out from the current SSL session. This command is appropriate for use with the forms-based login method.

```
https: //Web URL install-path/pkmslogout
```

For example:

```
https: /www.ibm.com/pkmslogout
```

You define the file that is displayed in response to this logout in the iv.conf configuration file:

```
# SSL HTML page locations
pkms-logout-page = lib/html/logout.html
```

You can change the logout.html file to suit your requirements.

The **pkmslogout** utility also supports multiple logout response pages when the network architecture requires different exit screens for users logging out of distinctly different back-end systems.

The following expression identifies a specific response file:

```
https: //pkmslogout?filename=custom_logout_file
```

Where *custom_logout_file* is the file name of the logout response. This file must reside in the same /lib/html/ directory that is defined for the default logout.html file.

pkmspasswd

Run this command to change your password.

```
https://Web URL install-path/pkmspasswd
```

For example:

```
https: //www.ibm.com/pkmspasswd
```

Using X.509 certificate authentication

WebSEAL supports authentication by using a client-side X.509 certificate over SSL. The X.509 certificate, instead of a username and password, provides the client's identity information.

Set-up tasks for client-side X.509 certificate support

Perform the following tasks to set up WebSEAL to accept client-side X.509 digital certificates:

Client tasks

To perform client tasks:

1. Obtain an X.509 client-side digital certificate (signed public key) from a CA.
2. Install the certificate on the client system.

WebSEAL server tasks

To perform WebSEAL server tasks:

1. Obtain the root CA certificate of the same Certificate Authority.

   The certificate can be in PEM or PKCS#12 format.
2. Copy the root CA certificate to the appropriate location on the system and indicate this location in the secmgrd.conf configuration file:

   PEM format:

   Append root certificates to the following file:

   UNIX: `ca-cert-file = lib/certs/cacert.pem`

   Windows: `ca-cert-file = lib\certs\cacert.pem`

   PKCS#12 format:

   Add each root certificate as a separate file in the following directory:

   UNIX: `ca-cert-p12-dir = lib/certs/ca_p12`

   Windows: `ca-cert-p12-dir = lib\certs\ca_p12`

   Note: These PEM-formatted and PKCS#12-formatted certificates are certificates of the Certificate Authorities that Policy Director trusts.

3. Specify how WebSEAL is to handle client-side X.509 certificates by setting the **verify-clients** parameter. Enter one of the following allowed values for verify-clients in the [wand] stanza of the iv.conf configuration file: never, optional, or required.

   See “Configuring certificate handling” on page 160 for descriptions of these values.

4. Configure WebSEAL to use a CAS server by editing the iv.conf file and changing the **cert-cdas** parameter in the [authentication-mechanisms] stanza, as needed, for your appropriate platform:

   ```
   [authentication-mechanisms]
   cert-cdas = &entry=/.:/subsys/intraverse/cdas/servers/hostname
   ```

   Choices for *cas module* include cdasauthn.dll for Windows NT, libcdasauthn.a for AIX, and libcdasauthn.so for Solaris.

   See “Basic Policy Director CAS configuration” on page 173 for details on the **cert-cdas** parameter.

5. Define the server identity using the **certificate-file** and **key-file** parameter in the secmgrd.conf configuration file. Note that the format of the server private key is PEM format. The parameters are different, depending on which platform you are using:

   For the certificate-file parameter in PEM format:

   UNIX: `certificate-file = /opt/intraverse/lib/certs/svrcert.pem`

   Windows: `certificate-file = C:\Program Files\ibm\Policy Director\lib\certs\svrcert.pem`

   For the key-file parameter in PEM format:

   UNIX: `key-file = /opt/intraverse/lib/certs/srvkey.pem`

   Windows: `key-file = C:\Program Files\ibm\Policy Director\lib\certs\srvkey.pem`

   See “Storing certificates” on page 159 for information about the secmgrd.conf certificate storage parameters.

6. Define the server identify using the certificate-file parameter in the secmgrd.conf configuration file. Note that the format of the server certificate is PKCS#12 format:

   For the certificate-file parameter in PKCS#12 format:

   For UNIX: `certificate-file = /opt/intraverse/lib/certs/svrcert.p12`

   For Windows: `certificate-file = C:\Program Files\ibm\Policy Director\lib\certs\svrcert.p12`

7. Use the Policy Director Credentials Acquisition Service (CAS) for credentials acquisition and mapping.

   Or, you can write and install your own credentials acquisition and mapping-service program on the server system. Refer to the *Policy Director Programmer’s Guide and Reference* and “Policy Director Credentials Acquisition Service” on page 172 for further information.

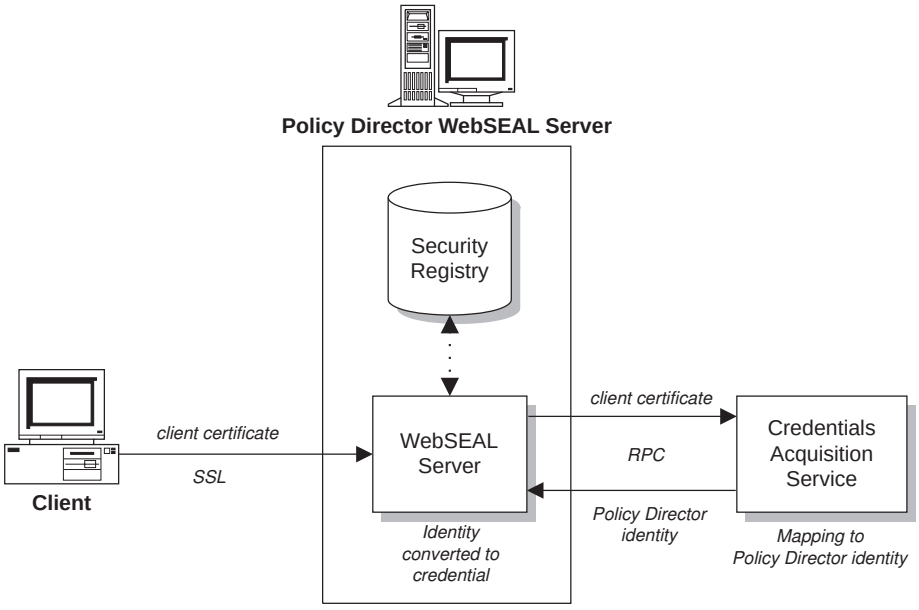
Policy Director Credentials Acquisition Service

The Policy Director Credentials Acquisition Service (CAS) is a customizable component that can be used to extend the standard authentication mechanisms that are supported by WebSEAL. The default Policy Director Credentials Acquisition Service uses the cdas_server(.exe) file. Refer to “Configuring WebSEAL to use the Policy Director CAS”.

Or, you can write and install your own credentials acquisition and mapping-service program on the server system. Refer to the *Policy Director Programmer’s Guide and Reference* for information on writing and installing a credentials acquisition service.

Introducing the Policy Director CAS

The Policy Director Credentials Acquisition Service (CAS) allows authentication and mapping of user identity information (such as X.509 certificate) to a Policy Director user identity. The Security Manager (using its default registry) returns credentials for this user identity.



Refer to “CAS provided with Policy Director” on page 29 and “Configuring WebSEAL to use the Policy Director CAS” for information on the Policy Director CAS.

Configuring WebSEAL to use the Policy Director CAS

You can configure all authentication mechanisms supported by WebSEAL in the `authentication-mechanisms` stanza of the iv.conf configuration file. All authentication mechanisms supported by WebSEAL represent both local (in-process) authenticators and any customized remote authenticators that are put into effect by CAS servers.

Local plugin modules

In the configuration file, you associate each authenticator with a local plugin module. On UNIX platforms, these modules are shared libraries. On Windows NT, these modules are Dynamic Link Libraries (DLL). These modules are provided as a standard part of the Policy Director distribution and are not customizable.

Policy Director provides a standard plugin module. Use this standard plugin module to interface to any third-party CAS server:

Platform	CAS Module Name
Solaris	libcdasauthn.so
AIX	libcdasauthn.a
Windows NT	cdasauthn.dll

Basic Policy Director CAS configuration

You can configure a Policy Director Credentials Acquisition Service that responds to X.509 certificate information, which accomplishes the client-side authentication of the WebSEAL CAS interface. The Policy Director CAS configuration requires an additional argument that represents the location in the DCE CDS namespace where the binding information for the CAS server is stored.

To configure WebSEAL to use a CAS server, edit the iv.conf file and change the cert-cdas parameter in the [authentication-mechanisms] stanza, as needed for your appropriate platform.

For example, the following configuration sequence (for Windows NT) identifies a single CAS server that supports X.509 certificate-based authentication:

```
[authentication-mechanisms]
cert-cdas = cdasauthn.dll&entry=/.:/subsys/intraverse/cdas/servers/hostname
```

Where `cdasauthn.dll` represents the CAS module for the appropriate platform, *hostname* is the simple host name, and `&entry=/.:/subsys/intraverse/cdas/servers/`*hostname* represents the location in the DCE CDS namespace where the binding information for the CAS server is stored.

Configuration entry syntax

An authentication configuration entry uses the following format:

authn-mechanism = *module*[&*arg1*[*arg2*]...[*argN*]]

Multiple Policy Director CAS servers

More than one authentication mechanism can be supported by a single Policy Director Credentials Acquisition Service. In this case, each mechanism has duplicated configuration information.

Distinguished name mapping

The Policy Director CAS maps *client digital certificates* from the SSL-enabled browser to a Policy Director user identity. When the user tries to access a protected Web page, the SSL-enabled browser contacts the WebSEAL server. If WebSEAL is configured for client certificate-based authentication, WebSEAL requests an X.509 certificate from the browser. When WebSEAL receives the certificate from the browser, it passes the certificate to the CAS server. The Policy Director CAS attempts to map the received certificate to a user identity understood by Policy Director.

Within the Policy Director CAS cdas.conf configuration file, the Policy Director administrator can create a table that is used to associate a certificate distinguished name (DN) with the DN of a Policy Director user. When the Policy Director CAS is called by WebSEAL with a certificate, it first extracts the DN from the certificate and searches the table for a match. If a match is found, the Policy Director Credentials Acquisition Service returns the associated Policy Director user correctly formatted DN to WebSEAL. This method is known as *DN mapping*.

WebSEAL then uses this DN to identify the Policy Director user. If a match is not found, the CAS returns the DN from the certificate to WebSEAL. In this case the DN in the certificate is used to identify the Policy Director user. The WebSEAL server uses the returned DN to retrieve the user's credentials.

The cdas.conf configuration file for mapping distinguished names can found at:

UNIX: `/opt/intraverse/cdas_server/lib/cdas.conf`

Windows: `C:\Program Files\IBM\Policy Director\cdas_server\lib\cdas.conf`

The cdas.conf configuration contains the following information:

```
# DN mapping

# If the certificate DN is in the following table, use the corresponding LDAP
# DN.  Otherwise, use the certificate DN as is.
# Each entry should be on a single line with the following format:
# [DN in the certificate] LDAP DN to map to
# For example:
# [/C=US/O=IBM/CN=Policy Director User] cn=Policy Director User,o=IBM,c=US
# [/C=US/O=IBM/CN=User1] cn=IBM Policy Director User,o=IBM,c=US
```

The certificate distinguished name (DN) is always located on the left side of the table and is always enclosed in brackets: `[/C=US/O=IBM/CN=Policy Director User]`. The DN of the Policy Director user (same as the LDAP registry DN) is always located on the right side of the table: `cn=Policy Director User,o=IBM,c=US`. The DN of the Policy Director user always follows the certificate DN right bracket ( ]) and a required space. Both sides of the mapping table entry must be completed in order to work properly.

SSL-enabled browsers, such as Netscape Communicator/Netscape Navigator and Microsoft Internet Explorer, allow you to view certificate DN information. The DN information for these browsers might be displayed differently; however, the certificate information for both browsers should contain all the distinguished name elements.

Chapter 14. WebSEAL: General administration tasks

This chapter contains information that describes general administration tasks and configuration tasks you can perform to customize WebSEAL for your network.

This chapter includes:

Enabling and disabling WebSEAL security

Use the **ivadmin** utility to enable and disable WebSEAL.

To enable WebSEAL on a specific Policy Director server:

```
ivadmin> server enable /WebSEAL/
```

Where *hostname* is the server's name, excluding the domain name.

When the service is enabled already or when the service specification is not valid, Policy Director returns errors.

By default, Policy Director enables WebSEAL.

To disable WebSEAL on a specific Policy Director server, use the **ivadmin server disable** command:

```
ivadmin> server disable /WebSEAL/
```

To check WebSEAL server status, use the **ivadmin server status** command:

```
ivadmin> server status hostname
```

The status report displays the following information:

- Whether the WebSEAL server is enabled or disabled.
- Whether the WebSEAL server is reachable using ping.
- The state of the WebSEAL configuration database.

Managing the Web space

This section describes the tasks that are required to manage the WebSEAL namespace, including:

Specifying the Web document tree locations

The Web document tree location is the absolute path to the root of the document tree for documents that are made available by the server. The default location is initially established during the installation of the Security Manager:

UNIX: *install-path*/www/docs

Windows: *install-path*\www\docs

You can change this location through the installation script. After installation, you must use the **junctioncp** utility to change this location. Refer to “Using junctioncp to manage smart junctions” on page 192 for complete command information.

The following UNIX example illustrates how to use the **junctioncp** utility to change the location:

1. Run **junctioncp**:

```
# junctioncp -e hostA
Attempting to bind to hostA at
/.:/subsys/intraverse/secmgr/server/hostA
junctioncp>
```

2. Use the **list** command to display all current junction points:

```
junctioncp> list
/
```

3. Use the **show** command to display details of the junction:

```
junctioncp> show /

Junction point: /
Type: Local Root
Directory: /opt/intraverse/www/docs
```

4. Create a new local junction to replace the current junction point:

```
junctioncp> create -t local -d /tmp/docs /
WARNING: A junction already exists at /
Do you want to replace it [no]? yes
Created junction at /
```

5. List the new junction point:

```
junctioncp> list
/
```

6. Display the details of this junction:

```
junctioncp> show /

Junction point: /
Type: Local Root
Directory: /tmp/docs
```

Configuring directory indexing

You can specify the name of the default file that is returned by the server. You specify this when you give a directory name as the URL. Policy Director returns this default file to the client when it exists. If it does not exist, Policy Director dynamically generates and returns a directory index to the client.

Note: Policy Director does not store the generated index to disk. Policy Director either retrieves the index from the server's wand or dirindex cache, or regenerates the index each time the directory is accessed.

Parameters for configuring directory indexing are located in the `[wand-indexing]` stanza of the iv.conf configuration file.

The value for the default file is:

```
[wand-indexing]
dirindex = index.html
```

If your site uses a different convention, you will want to change this file name:

```
[wand-indexing]
dirindex = default.html
```

Each parameter used for indexing directories has a default icon (.gif file) that is displayed for each document type and MIME type found:

```
[wand-indexing]
image/*= /icons/image2.gif
video/* = /icons/movie.gif
audio/* = /icons/sound2.gif
text/html = /icons/html.gif
text/* = /icons/text.gif
application/* = /icons/binary.gif
```

You can specify other icons for each parameter. Also, you can locate icons remotely, and you can use URLs as parameter values. For example:

```
application/* = http://www.acme.com/icons/binary.gif
```

Specifying the file extension types for CGI programs

Parameters contained in the `[wand-cgi-types]` stanza of the iv.conf configuration file allow you to specify the Windows file extension types. Policy Director recognizes the Windows file extension types that can be started as CGI programs.

The UNIX operating system has no file name extension requirements. However, you must define file extension types for Windows NT. The `[wand-cgi-types]` stanza lists all valid extension types and maps each extension (when necessary) to an appropriate CGI program.

By default, Policy Director starts only those files with extensions that match those listed in the stanza as CGI programs. By default, Policy Director runs files with the .exe extensions as programs, and these files require no mapping. You must supply the appropriate interpreter programs for extensions that represent interpreted script files. Examples extension types include: Shell scripts (.sh and .ksh), Perl scripts (.pl), and Tcl scripts (.tcl).

The following example illustrates a typical [wand-cgi-types] stanza configuration:

```
#
# CGI file extension to command mappings (Windows NT Only)
#
# For WIN32 servers we nominate CGI file extensions and the program
# that is used to execute them. If a CGI has an extension that is not
# in this list then it is not executed.
#
[wand-cgi-types]
.exe =
.bat =
.cmd =
.pl = perl
.sh = sh
.tcl = tclsh76
```

Note: There are serious security issues involved in the use of .bat files. Do not use .bat files.

Configuring HTTP and HTTPS worker threads

The number of configured worker threads specifies the number of concurrent incoming requests that a server can service. When all worker threads are busy, Policy Director buffers other connections that arrive until a worker thread is available.

You can set the number of threads available to service incoming connections. Configure the number of worker threads carefully because of possible performance impacts.

This configuration parameter does not impose an upper boundary on the number of simultaneous connections. This parameter simply specifies the number of threads that are made available to service a potentially unlimited work queue.

Choosing the optimal number of worker threads depends on understanding the quantity and type of traffic on your network.

By increasing the number of threads, in general you are decreasing the average time it takes to finish the requests. However, increasing the number of threads affects other factors that could have an adverse effect on server performance.

Setting the worker threads pool value for WebSEAL

WebSEAL maintains a single, generic worker list and also maintains a worker threads pool for handling requests from clients using TCP, SSL tunneling, or GSS tunneling. This enhanced mechanism enables WebSEAL to use fewer system resources while handling a significantly greater load.

You control the worker thread pool size by setting the worker-threads parameter in the [wand] stanza portion of the iv.conf configuration file.

```
worker-threads = 50
```

Configuring WebSEAL for HTTP requests

WebSEAL typically handles many HTTP requests from unauthenticated users. For example, it is desirable to allow unknown (and therefore unauthenticated) users read-only access to selected materials on a public Web site.

The [wand] stanza of the iv.conf configuration file contains parameters for handling HTTP requests over TCP.

Enabling and disabling HTTP listening

By default, Policy Director enables (allows) listening for HTTP requests over TCP:

```
allow-tcp-http = yes
```

Setting the parameter value to no disables HTTP listening.

Setting the Port Value

The default port for HTTP over TCP listening is 80:

```
http-tcp-port = 80
```

To change to port 8080, set:

```
http-tcp-port = 8080
```

Configuring WebSEAL for HTTPS requests

The [wand] stanza of the iv.conf configuration file contains parameters for handling HTTPS requests over SSL.

Enabling and disabling HTTPS listening

By default, Policy Director enables (allows) listening for HTTPS requests over SSL:

```
allow-ssl-http = yes
```

Setting the parameter value to no disables HTTPS by listening.

Setting the Port Value

The default port for HTTPS over SSL listening is 443:

```
ssl-port = 443
```

To change to port 4343, set:

```
ssl-port = 4343
```

Specifying time-out parameters

Policy Director time-out parameters that can be set include:

- The time-out parameters for HTTP communication
- Additional WebSEAL server time-out parameters in the [wand] stanza of the iv.conf configuration file

Time-out parameters for HTTP communication

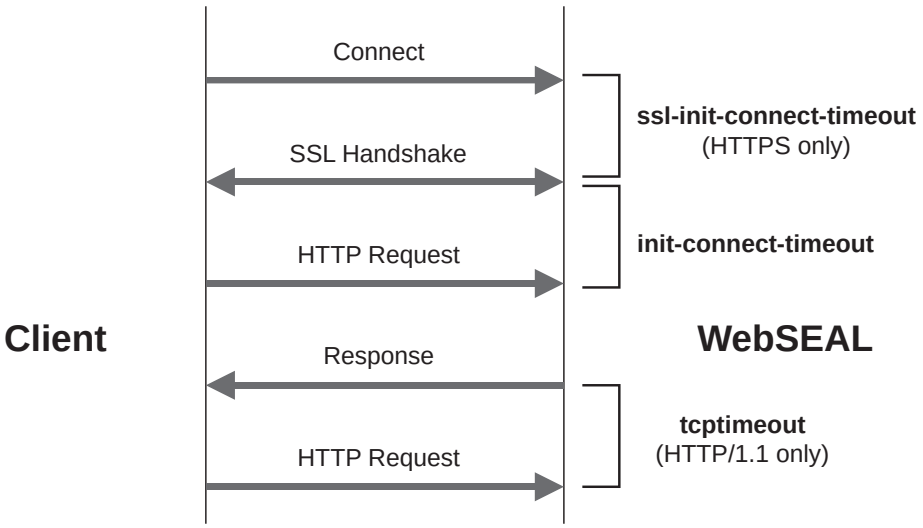
WebSEAL supports the following time-out parameters for HTTPS communication:

ssl-init-connect-timeout (HTTPS only)
When WebSEAL accepts an SSL connection from a browser, an SSL protocol *handshake* needs to take place. A handshake is the process of exchanging signals to set up communications between two modems. This parameter controls how long the Security Manager waits for an SSL browser to initiate an SSL handshake. This initialization occurs at the start of an SSL connection, before shutting down the connection.

init-connect-timeout
After an SSL handshake has occurred, this parameter dictates how long WebSEAL waits for an initial HTTP request. This connection could be HTTP, HTTPS, or NetSEAT (HTTP and GSS).

tcptimeout

This parameter is specific to HTTP/1.1 (not HTTP/1.0) connections. After the first HTTP/1.1 request and server response, this parameter controls the maximum number of seconds the server holds an HTTP/1.1 persistent connection open. It is shut down after the maximum number of seconds is reached.



Parameter	Configuration File	Default Value (seconds)
ssl-init-connect-timeout	secmgrd.conf [ssl] stanza	120
init-connect-timeout	iv.conf [wand] stanza	120
tcptimeout	iv.conf [wand] stanza	5

Additional WebSEAL server time-out parameters

The [wand] stanza of the iv.conf configuration file is where you set following time-out parameters:

Parameter	Description	Default Value (seconds)
tcp-junction-timeout	The timeout for sending to and reading from a back-end server over a TCP junction.	120
ssl-junction-timeout	The timeout for sending to and reading from a back-end server over an SSL junction.	120
cgi-timeout	The timeout for sending to and reading from a local CGI process.	120
junction-ping-time	WebSEAL performs a periodic background ping of each junctioned server to determine whether it is running. WebSEAL does not try more often than once every 300 seconds (or whatever value is set).	300

Configuring HTTP error messages

Sometimes the WebSEAL server attempts to service a request and fails. There can be many causes of this failure. For example:

- A file does not exist.
- Permission settings forbid access.
- Incorrect UNIX file permissions or something similar that prevents CGI programs from starting.

When a failure to service a request occurs, the server returns an error message to the browser, such as `403 Forbidden`, in an HTML error page. There are several error messages available. A separate HTML file stores each message.

The following directory contains these files:

UNIX:

install-path`/www/lib/errors/`*locale-dir*

Windows:

install-path`\www\lib\errors\`*locale-dir*

The errors directory contains a number of locale subdirectories. The subdirectories contain localized versions of the error message files.

The messages in this directory are in HTML format so that they appear correctly in a browser. You can edit these HTML pages to customize their contents. The names of the files are the hexadecimal values of the internal error codes that are returned when the operations fail. Do not change these file names.

The following table contains a list of the file names and contents for some of the more common error messages:

Filename	Title	Description	HTTP Error Code
1354a2fa.html	Non-Empty Directory	The requested operation requires the removal of a non-empty directory. This is an illegal operation.	
1898d25a.html	Could Not Sign User On	The resource requested requires the WebSEAL server to sign user on to another Web server. However, a problem occurred while WebSEAL was attempting to retrieve the information.	
1898d25b.html	User Has No Single Sign-on Information	WebSEAL could not locate the GSO user for the requested resource.	
1898d25c.html	No Single Sign-on Target for User	WebSEAL could not locate the GSO target for the requested resource.	
1898d25d.html	Multiple Sign-on Targets for User	Multiple GSO targets are defined for the requested user. The GSO targets are incorrectly configured.	

1898d25e.html	Login Required	The resource requested is protected by a junctioned back-end Web server, requiring WebSEAL to sign user on to that Web server. In order to do this, user must first log in to WebSEAL.	
1898d25f.html	Could Not Sign User On	The resource requested requires WebSEAL to sign user on to another Web server. However, the sign-on information for the user account is incorrect.	
1898d260.html	Unexpected Authentication Challenge	WebSEAL received an unexpected authentication challenge from a junctioned back-end Web server.	
1898d421.html	Moved Temporarily	The requested resource has been temporarily moved, which usually occurs when there has been a mishandled redirect.	302
1898d424.html	Bad Request	WebSEAL received an invalid HTTP request.	400
1898d425.html	Login Required	The resource you have requested is secured by WebSEAL, and in order to access it, you must first log in.	
1898d427.html	Forbidden	User does not have permission to access requested resource.	403
1898d428.html	Not Found	Requested resource cannot be located.	404
1898d432.html	Service Unavailable	A service required by WebSEAL to complete request is currently not available.	503
1898d437.html	Server Suspended	The WebSEAL server has been temporarily suspended by the System Administrator. No requests are handled until the server is returned to service by the Administrator.	
1898d439.html	Session Information Lost	The browser and server interaction was a *stateful session* with a junctioned back-end server that is no longer responding. WebSEAL requires a service located on this server to complete your request. See “Maintaining a state (-s option)” on page 197.	
1898d7af.html	CGI Program Failed	A CGI program failed to execute properly.	
default.html	Server Error	WebSEAL could not complete your request because of an unexpected error.	500

Macro support

The following macros are available for use in a customized HTML error message page. Macros dynamically substitute appropriate information that is available.

Macro	Description
ERROR_CODE	The numeric value of the error code.
ERROR_TEXT	The text associated with an error code in the message catalog.
METHOD	The HTTP method requested by the client.
URL	The URL requested by the client.
HOSTNAME	Fully qualified host name.
HTTP_BASE	Base HTTP URL of the server `http://host:tcpport/`
HTTPS_BASE	Base HTTPS URL of the server: `https://host:sslport/`
REFERER	The value of the referer header from the request, or `Unknown` (if none).
BACK_URL	The value of the referer header from the request, or / (if none).
BACK_NAME	The value `BACK` if a referer header is present in the request, or the value `HOME` if it is not present.

Chapter 15. WebSEAL: smart junction administration

WebSEAL can function as a basic standalone Web server or as a junction server that provides authentication and authorization services for back-end application servers. The primary strength of WebSEAL is its ability to integrate and protect additional Web resources on back-end application servers. WebSEAL integrates and protects Web resources by using Smart Junction technology.

This chapter contains:

Introducing WebSEAL as a smart junction server

Policy Director provides authentication, authorization, and management services for a network. In a Web-based network, these services are best provided by a front-end WebSEAL server. The front-end WebSEAL server protects Web resources that are located on back-end application servers.

The connection between a WebSEAL server and a back-end server is known as a *smart junction*, or junction. Use a junction to combine the physical Web spaces of the WebSEAL and back-end servers to build a single logical Web space representation.

The client never needs to know the physical location of a Web resource. WebSEAL translates logical URL addresses into the physical addresses that a back-end server expects. You can move Web objects from server to server without affecting the way the client accesses those objects.

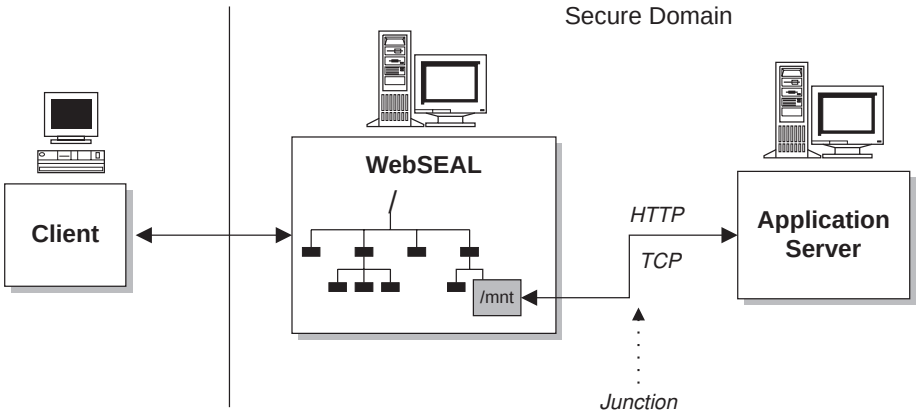
As a junction server, WebSEAL can perform authentication and authorization checks on all requests before passing those requests on to a back-end server. Junctions provide a scalable, secure environment that allows load balancing, high availability, and state management capabilities—all performed transparently to clients. Administrators benefit from centralized management of the namespace.

Most commercial Web servers do not have the ability to define a logical Web namespace. Instead, their access control is connected to the physical file and the directory structure. Smart junctions can transparently define a namespace that reflects organizational structure instead of the physical machine and the directory structure that is commonly encountered on standard Web servers.

Smart junctions also allow you to create single sign-on solutions. A single sign-on configuration allows a user to access a resource, regardless of the resource's location, using only one initial login. Any further login requirements from back-end servers are handled transparently to the user.

Understanding smart junctions

A *smart junction* is a physical TCP/IP connection between a front-end WebSEAL server and a back-end application server. The back-end server can be another WebSEAL server or a third-party application server. The back-end application Web space *connects* to the WebSEAL server at a specially designated *junction point* (mount point) in the WebSEAL Web space.

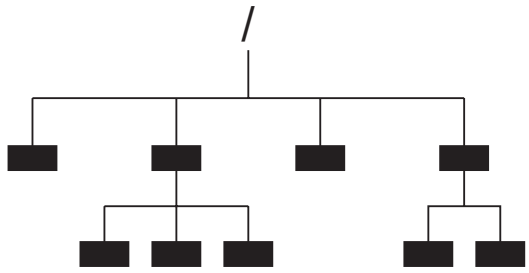


A smart junction allows WebSEAL to provide protective services on behalf of the back-end application server. The back-end server requires fine-grained access control on its objects. When it requires this control, you must perform additional configuration steps to describe the third-party Web space to the Policy Director security service.

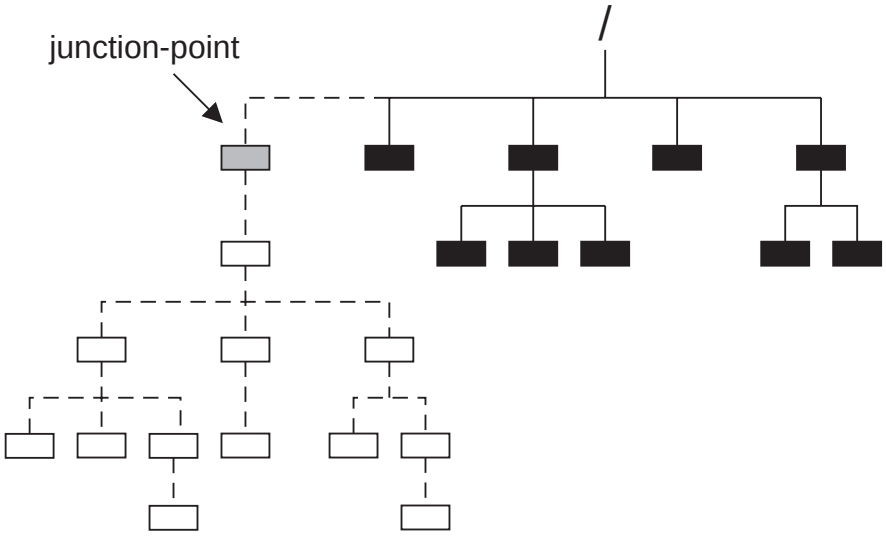
When properly configured, WebSEAL protects its own resources, and the resources on the junctioned server, by performing security services such as authentication, authorization, and auditing.

Smart junctions provide the added value of logically combining the Web space of a WebSEAL server with the Web space of the back-end server. Junctions between cooperating servers result in a single, unified, distributed Web space that is seamless and transparent to users.

A unified Web space simplifies the management of all resources for the system administrator. Additional administrative benefits include scalability, load balancing, and high availability.



WebSEAL Web space



Combined Web space:
WebSEAL plus junctioned server

Smart junctions are an important tool for making your Web site scalable. Junctions allow you to respond to increasing demands on a Web site by attaching additional servers.

Smart junctions and Web site scalability

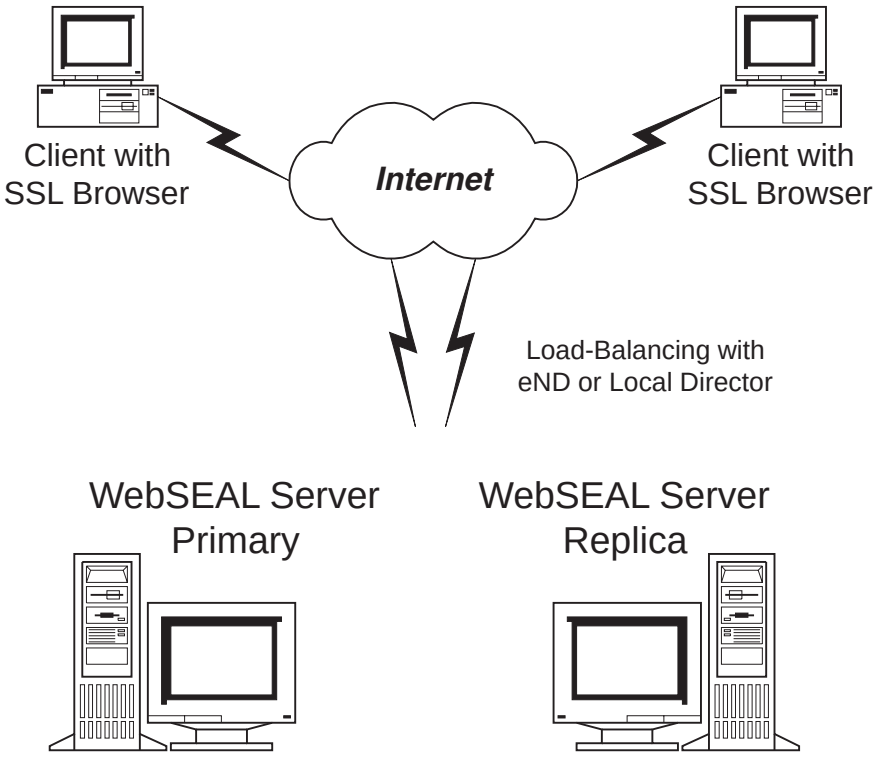
Use smart junctions to create a scalable Web site. As the demands on the Web site grow, you can easily add more servers to expand the capabilities of the site. You might want to add additional servers for these reasons:

- To extend the site with additional content
- To duplicate existing content for load balancing, failover, and high availability capabilities

Replicated front-end WebSEAL servers

Junction support for back-end servers starts with at least one front-end WebSEAL server. Replicated front-end WebSEAL servers provide the site with load balancing during periods of heavy demand. An application (such as Policy Director eND or Cisco Local Director) handles the load balancing mechanism.

Front-end replication also provides the site with fail-over capability. If a server fails for some reason, the remaining replica servers will continue to provide access to the site. Successful load balancing and fail-over capability results in high availability for users of the site.



There are two key points to remember about replicating front-end WebSEAL servers:

- Each server must contain an exact copy of the Web space.
- You must replicate the user account database for consistent authentication.

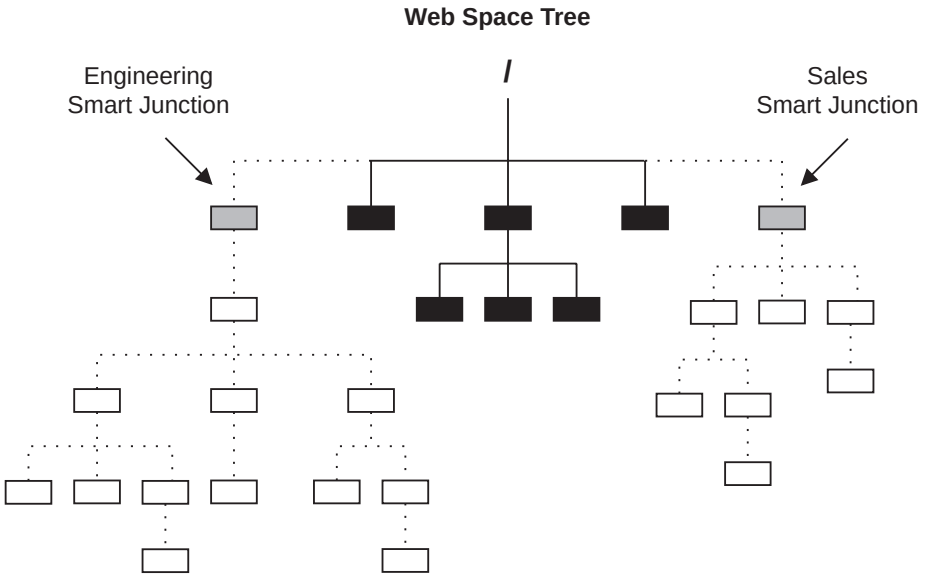
The Policy Director Authorization Service automatically replicates the authorization database information where required.

Supporting back-end servers

The WebSEAL server itself, a back-end server or servers, or a combination of both the WebSEAL server and a back-end server can serve Web site content. Smart junction support for back-end servers allows you to scale the Web site through additional content and resources.

Each unique back-end server must be junctioned to a separate junction (mount) point. As the demand for additional content and resources grows, add more servers by using smart junctions. This scenario provides a solution for networks that have a large existing investment in third-party Web servers.

The following diagram illustrates how smart junctions provide a unified, logical Web space. This Web space is transparent to the user and allows for centralized management.



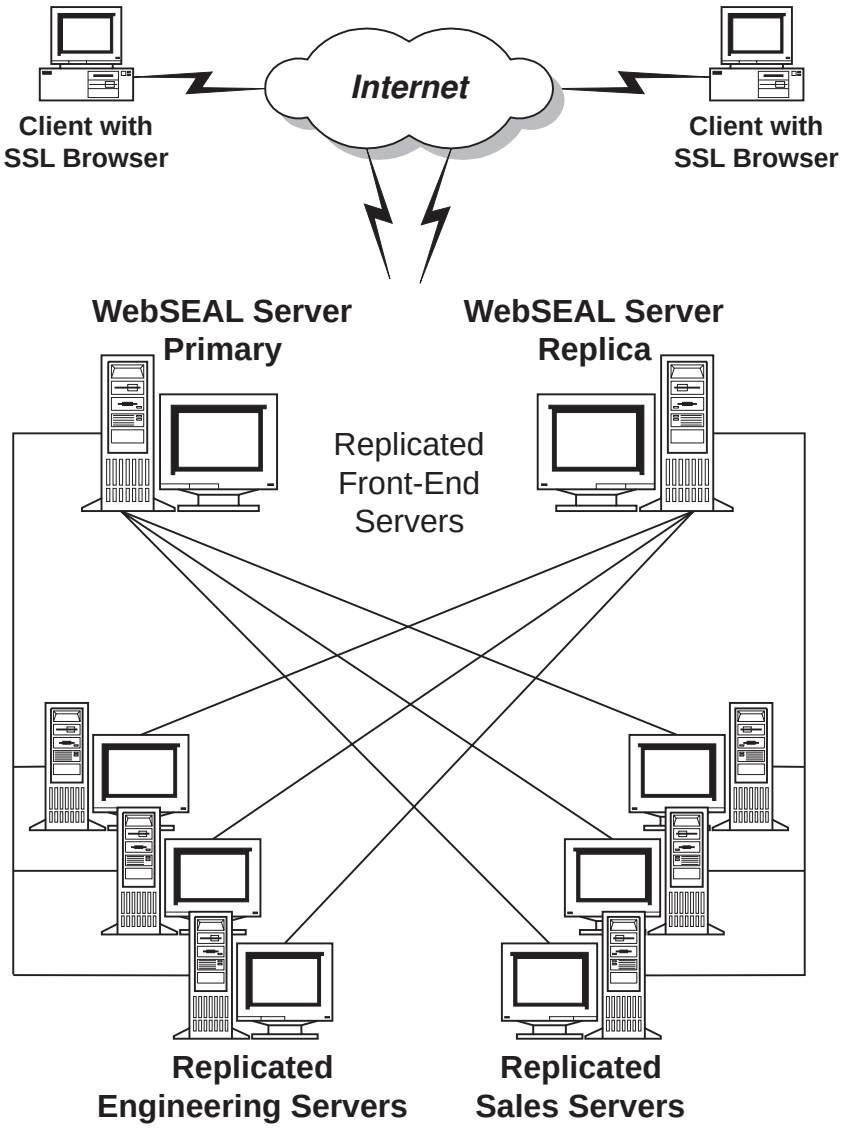
Replicated back-end servers are junctioned to the same junction point, as discussed in “Replicated back-end servers” on page 190.

Replicated back-end servers

To extend scalability features to a back-end server configuration, you can replicate the back-end servers. As is the case with replicated front-end servers, replicated back-end servers must contain Web spaces that are mirror images of each other.

WebSEAL load balances across the replicated servers using a “least-busy” scheduling algorithm. WebSEAL also correctly fails over when a server is down and starts using that restarted server again.

If the back-end application requires that the state be maintained over several pages, you can use stateful junctions. These *stateful junctions* ensure that each session returns to the same back-end server. See “Maintaining a state (-s option)” on page 197.



Task summary for creating junctions

You can create the following junction types:

- Policy Director to third-party; TCP connection
- Policy Director to third-party; SSL connection
- Policy Director to local file system

The following steps summarize the tasks you need to perform to junction a back-end application Web space to the WebSEAL Web space:

1. Decide where to junction (mount) the additional servers in the WebSEAL Web space.
2. Determine what security conditions you require to ensure the integrity of your network.

Coarse-grained access control

To provide coarse-grained access control:

1. Create a smart junction between the back-end third-party application Web space and the front-end WebSEAL server by using the Policy Director **junctioncp** utility.
2. Place an appropriate policy template (ACL) on the junction point to provide coarse-grained control to the back-end server.

Fine-grained access control

To provide fine-grained access control:

1. Create a smart junction between the back-end third-party application Web space and the front-end WebSEAL server by using the **junctioncp** utility.

   WebSEAL cannot automatically *see* and understand a third-party file system. You must inform WebSEAL of the third-party namespace by using **query_contents**. This special application inventories the third-party Web space and reports the structure to WebSEAL.
2. Copy the **query_contents** program to the third-party server.
3. Use the Management Console to apply policy templates (ACLs) to appropriate objects in the unified Web space.

Guidelines for creating smart junctions

The following guidelines summarize the rules for smart junctions:

- You can add a smart junction anywhere in the primary WebSEAL namespace.
- You can junction multiple replica servers at the same mount point (junction point).
- Multiple replica servers mounted at the same junction point must be of the same type (TCP or SSL).
- You cannot chain third-party servers (such as Policy Director — third-party — third-party).

Access control and administrative privileges

By default, the cell administrator account has full rights to the entire Web space, which include junctioned servers. The administrator of a junctioned server maintains the Web space only for that server. When necessary, this administrator can remove administrative privileges for the server from the cell administrator.

Policy Director inherits ACLs across smart junctions to third-party servers.

Using junctioncp to manage smart junctions

Use the **junctioncp** utility to perform all junction management tasks:

- Create a new junction point
- Add a server to the junction point
- Remove a server from a junction point
- Delete a junction point
- Display a list of junction points
- Display the details of a junction

The **junctioncp** utility provides an interactive command prompt where you can perform junction tasks.

Before starting **junctioncp**, you must log in to a secure domain as an administrative user. You can use dce_login in a UNIX or Windows environment. You can use netseat_login in a Windows environment.

Start the **junctioncp** utility with the **-e** option to specify the server (host name) where you want to perform junction tasks. The **junctioncp** command prompt appears.

For example:

UNIX:

```
#
# junctioncp -e server-name

junctioncp>
```

Windows:

```
junctioncp -e <server-name>

junctioncp>
```

Using the junctioncp commands

The following commands are available with **junctioncp**:

Command	Description
create	Create a new junction for an initial server.
add	Add additional servers to an existing junction point.
remove	Remove a server from a junction point.
delete	Remove the junction point.
list	List all junction points on this server
show	Display the details of a junction
help	List **junctioncp** commands.
help *command*	Display detailed help on a specific **junctioncp** command.
exit	Exits the **junctioncp** utility.

See “Creating a new junction for an initial server” for a discussion of these commands and associated options.

Creating a new junction for an initial server

Operation: Creates a new junction point and junctions an initial server.

Syntax: `create -t type -h hostname options junction-point`

type		
One of: • **tcp** • **ssl** • **local**		*Required.* Defines the type of junction. Use **tcp** for back-end third-party servers. When you use the **ssl** option, the default TCP port is changed from 80 to 443. See “Creating secure SSL smart junctions” on page 199.
options		
	TCP and SSL junction options (Use with **-t tcp** or **-t ssl**)	
	-2	Forces communication with the back-end server using SSL Version 2 only. See “Creating secure SSL smart junctions” on page 199.
	-b *ba-value* One of: • **filter** (default) • **ignore** • **supply** • **gso**	Defines how the WebSEAL server passes authentication information to the back-end servers. See “Supplying authentication information to junctioned servers” on page 204.
	-c	Inserts Policy Director client identity in HTTP headers. See “Inserting client identity information (-c option)” on page 198.

	-i	Causes WebSEAL server to treat URLs as case insensitive. See “Supporting case-insensitive URLs (-i option)” on page 196.
	-h *hostname*	*Required.* Defines the target back-end server’s host name (DNS). You can alternately supply its IP address.
	-p *port*	Defines the TCP port of the back-end, third-party server. Default is 80 for TCP junctions; 443 for SSL junctions.
	-q *url*	Defines the URL for **query_contents** script. Policy Director looks for **query_contents** in /cgi_bin/. When this directory is different or the **query_contents** file renamed, use this option to indicate to WebSEAL the new URL to the file. When you create a smart junction for the third-party Win32 server, use this **-q** option. See “Configuring a smart junction to find query contents” on page 213.
	-s	Specifies that the junction should support stateful applications. By default, junctions are not *stateful*. See “Maintaining a state (-s option)” on page 197.
	-T *resource*	Defines the name of the application resource for GSO resource credentials. Required for and used with only the **-b gso** option. See “Integrating GSO and WebSEAL single sign-on” on page 207.
	-v *hostname*	Defines the virtual host name for the server.
	-w	Defines the Win32 file system support. See “Disallowing the short file name form (-w option)” on page 197.
	Local and DFS junction options (use with **-t dfs** or **local**)	
	-d *dir*	Defines the distributed file system (DFS) or local directory to junction. Required.
junction-point		
	Defines the location in the WebSEAL namespace to create the junction.	

Adding an additional server to an existing junction

Operation: Adds an additional server to an existing junction point.

Syntax: `add -h hostname options junction-point`

options		
	TCP and SSL junction options	
	-i	Causes the WebSEAL server to treat URLs as case insensitive. See “Supporting case-insensitive URLs (-i option)” on page 196.
	-h *hostname*	*Required.* Defines the target back-end server host name (DNS). You can alternately supply its IP address.
	-p *port*	Defines the TCP port of back-end third-party server. Default is 80 for TCP junctions; 443 for SSL junctions.
	-q *url*	Defines the URL for **query_contents** script. Policy Director looks for **query_contents** in /cgi_bin/. When this directory is different or the **query_contents** file renamed, use this option to indicate to WebSEAL the new URL to the file.
	-v *hostname*	Defines the virtual host name for server.
	-w	Defines the Win32 file system support. See “Disallowing the short file name form (-w option)” on page 197.
junction-point		
	Adds a server to this existing junction point.	

Using other junctioncp commands

"Creating a new junction for an initial server" on page 193 and "Adding an additional server to an existing junction" on page 195) discuss the **junctioncp create** and **junctioncp add** commands. The following table lists the other **junctioncp** commands available:

Command	Descriptions
remove	**Operation:** Removes a back-end server from a junction point. **Syntax:** `remove -i server-id junction-point` **Options: -i** *server-id* Identification of the server to remove. Use the show command to determine the ID of a particular server.
delete	**Operation:** Removes a junction point. **Syntax:** `delete junction-point`
show	**Operation:** Displays the details of a junction. **Syntax:** `show junction-point`
list	**Operation:** Lists all junctions. **Syntax:** `list`
help	**Operation:** Displays list of **junctioncp** commands. **Syntax:** `help`
help *command*	**Operation:** Displays information about a specific **junctioncp** command, including available options. **Syntax:** `help command`
exit	**Operation:** Exits the **junctioncp** utility and returns the operating system prompt. **Syntax:** `exit`

Supporting case-insensitive URLs (-i option)

Use the **-i** option when junctioning third-party servers to specify that WebSEAL treat URLs as case-insensitive. This means that the server does not distinguish between uppercase characters and lowercase characters when parsing URLs. By default, expect servers to be case-sensitive.

Although most HTTP servers support the HTTP specification, which defines URLs as case-sensitive, some HTTP servers treat URLs as case-insensitive.

For example, on case-insensitive servers, you can view the two URLs as the same URL:

```
http://server/sales/index.htm

http://server/SALES/index.HTM
```

This behavior requires Policy Director to place the same access controls (ACLs) on both URLs. By default, Policy Director treats URLs as case-sensitive when applying access controls. By junctioning a third-party server with the **-i** option, WebSEAL treats the URLs that are directed to that server as case-insensitive.

Disallowing the short file name form (-w option)

The goal is to restrict access control to one object representation only. Do not allow *back doors* that bypass the security mechanism.

WebSEAL performs security checks on client requests to junctioned back-end servers based on the file paths specified in the URL. A compromise in this security check can occur because Win32 file systems provide for two different methods for accessing long file names.

The first method acknowledges the entire file name (`abcdefghijkl.txt`). The second method uses the old 8.3 file name format for backward compatibility (`abcdef~1.txt`).

The **-w** option to the **junctioncp** command disallows the 8.3 file name format. A user cannot avoid an explicit ACL on a long file name by using the short (8.3) form of the file name. The server returns a `403 Forbidden` error on any short form file name entered.

Windows treats a file name of “foo.” the same as it treats a file name of “foo” without the trailing dot. The **-w** option removes trailing dots from file names in a URL before sending the request to the back-end server. Policy Director bases ACL checks on the file name—without the trailing dot.

Note: The **-i** option addresses the problem with Win32 case-insensitivity (`abced.txt = AbCdE.txt`).

Example: On Windows NT 4.0, you can access the following paths for the file \Program Files\ibm corp\readme.txt these ways:

1. \program files\ibm corp.\readme.txt
2. \program files\ibm corp\readme.txt
3. \progra~1\ibm~2\readm~3.txt

Example 1 above illustrates the “case-insensitivity” effect. The **-i** option (not the **-w** option) addresses this effect.

Example 2 illustrates how Windows NT ignores a trailing extension dot.

Example 3 illustrates how Windows NT creates an alias for Disk Operating System (DOS) compatibility. The alias must contain no spaces in the file names and conform to the 8.3 format.

The **-w** option addresses the potential security holes that are illustrated by Examples 2 and 3. The **-w** option dictates that Policy Director ignore trailing dots. This option also dictates that Policy Director disallow access to shortened file names that contain a tilde character (~) in request URLs for this junctioned server.

Maintaining a state (-s option)

Most Web-enabled applications maintain a *state* across a sequence of HTTP requests that are contained in each client session. For example, use this state to:

- Track a user’s progress through the fields in a data entry form that is generated by a CGI program
- Maintain a user’s context when performing a series of database inquiries
- Maintain a list of items in an online shopping cart application where a user randomly browses and selects items to purchase

You can replicate servers that run Web-enabled applications, like any server, in order to improve performance through load sharing. The Policy Director server can provide a smart junction to these replicated servers. When it does, it must ensure that all the requests contained within a client session are forwarded to the correct server. Also, it must ensure that all the requests are not distributed among the replicated servers according to the load balancing rules.

By default, Policy Director balances server load by distributing requests across all available replicated servers. Policy Director uses a “least-busy” algorithm.

To override this load balancing and create a *stateful junction*, use the **junctioncp** command with the **-s** option. The stateful junction ensures that a client’s requests are forwarded to the same server throughout an entire session.

Inserting client identity information (-c option)

The **-c** option allows you to insert Policy Director-specific client identity and group membership information into the headers of the HTTP requests. The HTTP requests are destined for junctioned third-party servers. The Policy Director-specific HTTP headers enable applications on junctioned, third-party servers to perform user-specific actions. User-specific actions are based on the client’s Policy Director identity.

You must transform HTTP header information to environment-variable format for use by a service on the back-end server. Transform header information into a CGI environment-variable format by replacing all dashes (-) with underbars (_) and pre-pending “HTTP” to the beginning of the string. The value of the HTTP header becomes the value of the new environment variable.

The Policy Director-specific HTTP header entries include:

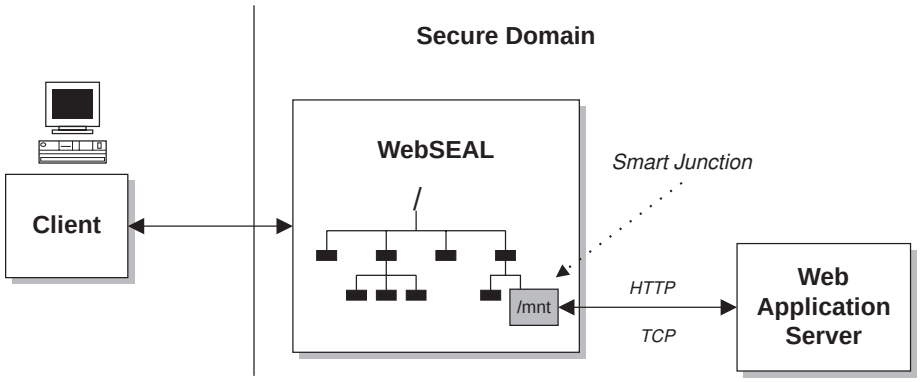
Policy Director-specific HTTP Headers	CGI Environment Variable Format	Description
iv-user	HTTP_IV_USER	The name of the client. Defaults to **Unauthenticated** when the client is unauthenticated (unknown).
iv-groups	HTTP_IV_GROUPS	A list of groups to which the client belongs. Consists of space-separated entries.
iv-creds	HTTP_IV_CREDS	Encoded opaque data structure representing a Policy Director resource credential. Used in conjunction with the Policy Director Authorization API. For more details, refer to the *Policy Director Programmer’s Guide and Reference*.

The HTTP header entries are available to CGI programs as the environment variables HTTP_IV_USER, HTTP_IV_GROUP, and HTTP_IV_CREDS. For other application framework products, see the product’s documentation for instructions on extracting headers from HTTP requests.

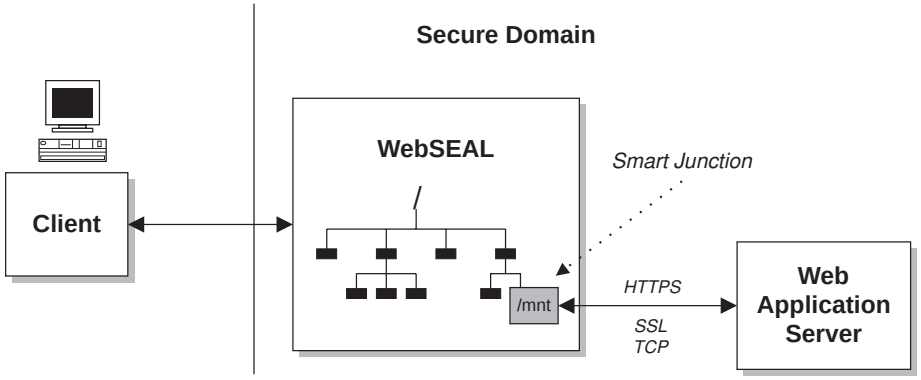
Creating secure SSL smart junctions

WebSEAL supports both standard TCP (HTTP) and secure SSL (HTTPS) junctions between WebSEAL and the back-end servers. SSL junctions function exactly like TCP junctions, with the added value that all communication between WebSEAL and the back-end server is encrypted.

The following figure represents a nonsecure TCP (HTTP) junction.



The following figure represents a secure SSL (HTTPS) junction.



The junction between WebSEAL and the back-end server is independent of the type of connection (and its level of security) between the client and the WebSEAL server.

SSL junctions allow secure end-to-end, browser-to-application transactions. Use SSL to secure communications from the client to WebSEAL and from WebSEAL to the back-end servers.

Configuring a secure SSL junction

SSL smart junction functionality requires that the back-end Web server be HTTPS-enabled.

You use the **junctioncp** utility to create a smart junction. “Using junctioncp to manage smart junctions” on page 192 describes details of the **junctioncp** utility.

To create a secure SSL junction and add an initial server, use the **junctioncp create** command. The following example illustrates the syntax of the create command for creating a secure SSL junction:

```
junctioncp> create -t ssl [-2] -h hostname [-p port] junction-point
```

The **-2** option forces Policy Director to communicate with the back-end server by using SSL Version 2 only.

Normally Policy Director automatically negotiates the version of the SSL protocol (either Version 2 or 3). Policy Director introduced the **-2** option because some IIS servers cause Policy Director to fail when it tries to negotiate SSL Version 3. When this occurs, mounting does not succeed. Forcing the use of Version 2 solves this problem.

Reviewing examples of SSL junctions

Junction host sales.ibm.com at junction point /sales that uses the SSL protocol:

```
create -t ssl -h sales.ibm.com /sales
```

Junction host admin.ibm.com at junction point /admin that uses the SSL Version 2 protocol only:

```
create -t ssl -2 -h admin.ibm.com /admin
```

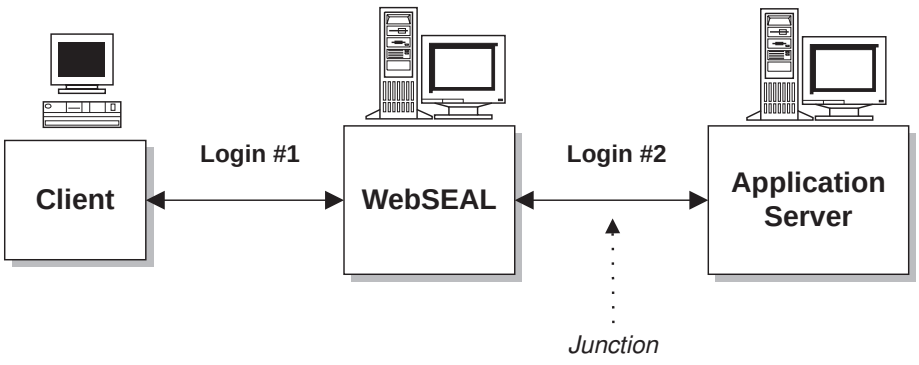
Note: In the above two examples, the **-t ssl** option dictates a default port of 443.

Junction host travel_svr on port 4343 at junction point /travel that uses the SSL protocol:

```
create -t ssl -p 4343 -h travel_svr /travel
```

Using the Policy Director single sign-on solution

When you locate a protected resource on a back-end server, a client requesting that resource might be required to perform multiple logins. The multiple logins include one for the WebSEAL server and one for each of the back-end servers. Each login likely requires different login identities.



You can solve the problem of administering and maintaining multiple login identities with a single sign-on mechanism. A single sign-on solution allows the user to access a resource, regardless of the resource's location, using only one initial login. Any further login requirements from back-end servers are handled transparent to the user.

The network security administrator must make three important decisions when configuring a Policy Director single sign-on mechanism:

1. Is authentication information required by the back-end servers?

   WebSEAL uses the HTTP *basic authentication* header to convey authentication information.
2. If authentication information is required by the back-end servers, where does this information come from? (What information does WebSEAL place in the HTTP header?)
3. Does the connection between WebSEAL and the back-end servers need to be secure? (TCP junction or SSL junction?)

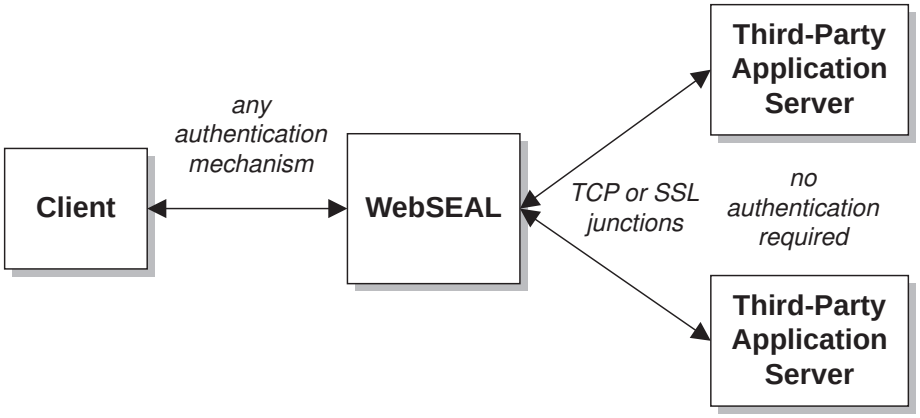
The following sections examine some typical WebSEAL single sign-on configurations.

Back-end servers not requiring authentication

When the back-end server does not require authentication information, the following conditions exist:

- You do not need to configure WebSEAL to send authentication information across the junction.
- You can only access back-end servers through WebSEAL.
- WebSEAL handles authentication on behalf of all back-end servers.
- There is still a special option to pass user identity information to the back-end servers for further authorization actions, if required. This special option is the **-c** option of the **junctioncp** utility.

See "No authentication information" on page 206.

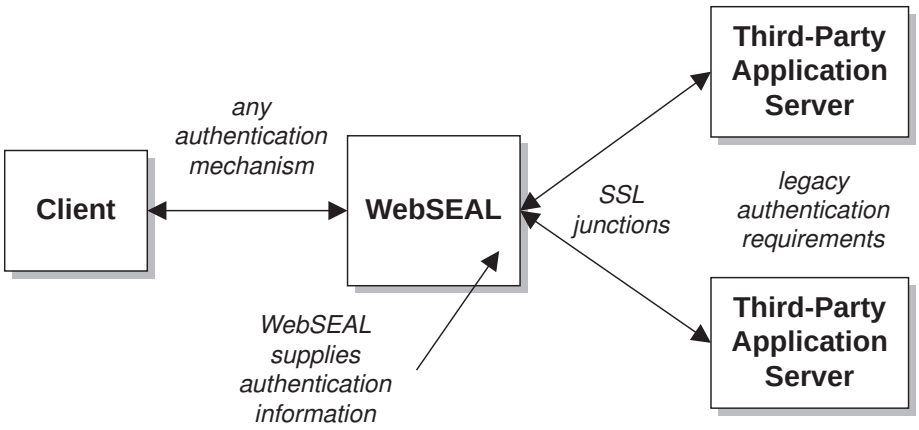


Back-end servers with legacy authentication needs

When the back-end server contains legacy authentication mechanisms that must be supported, the following conditions exist:

- You must configure WebSEAL to supply the appropriate authentication information to the back-end servers.
- The authentication information would most likely come from a mechanism such as GSO.

  See “Integrating GSO and WebSEAL single sign-on” on page 207.
- Because Policy Director passes sensitive authentication information (username and password) across the junction, the security of the junction is important. Therefore, Policy Director recommends the use of SSL junctions.

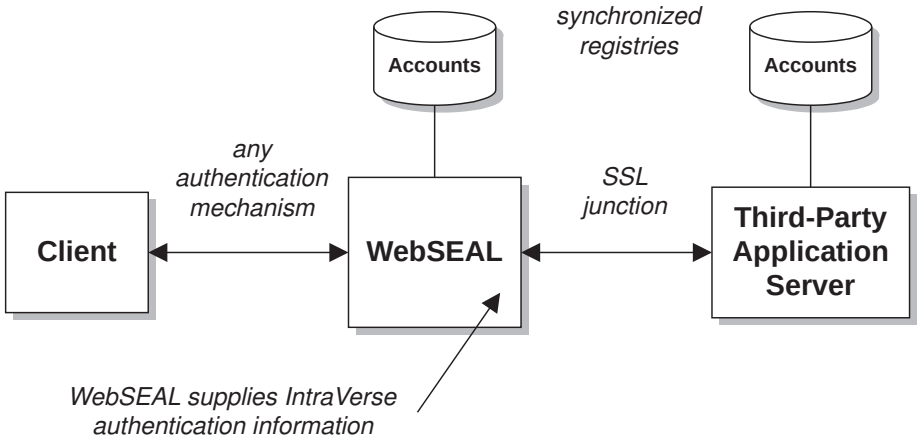


Policy Director single sign-on

When the back-end server requires Policy Director authentication information, the following conditions exist:

- You must configure WebSEAL to supply the back-end servers with username and password that are contained in the original client request.
- The back-end servers must understand the Policy Director identity and password supplied in the HTTP basic authentication (BA) header. Therefore, WebSEAL and the back-end servers must have synchronized user registries.
- Because Policy Director passes sensitive authentication information (username and password) across the junction, the security of the junction is important. Policy Director recommends the use of SSL junctions.

See “Original client BA header information” on page 205.

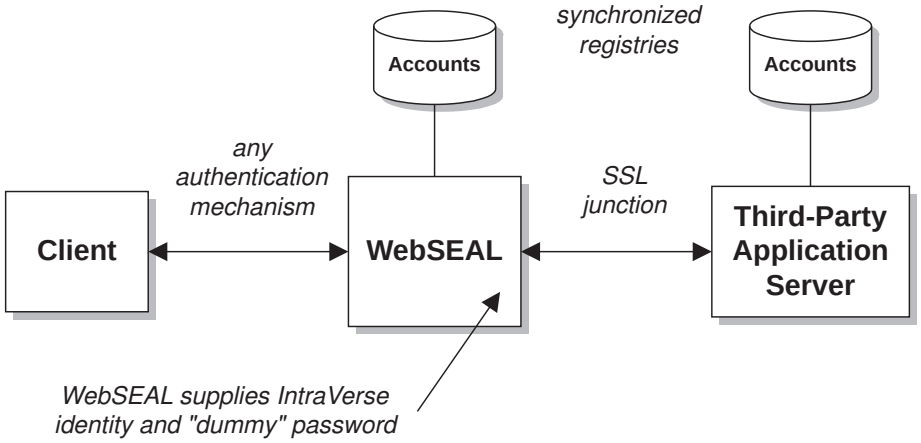


Limited Policy Director single sign-on

A limited Policy Director single sign-on is available where the HTTP BA header supplies the Policy Director identity (user name). A static, generic password is also supplied. This scenario is appropriate for per user-application usage. The solution can be advantageous because it requires no on-going password administration. The following conditions exist:

- WebSEAL must be configured to supply the back-end servers with the user name that is contained in the original client request plus a generic (dummy) password.
- Configure the dummy password in the iv.conf configuration file.
- The back-end servers must understand that the Policy Director identity supplied in the HTTP BA header. Therefore, WebSEAL and the back-end servers must have synchronized account registries.
- Because Policy Director passes sensitive authentication information (username and password) across the junction, the security of the junction is important. Policy Director recommends the use of SSL junctions.

See “Policy Director identity and generic password” on page 204.



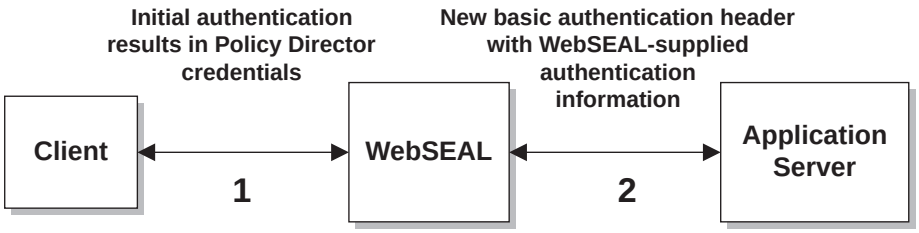
Supplying authentication information to junctioned servers

There are a number of possible scenarios for passing authentication information from WebSEAL to a back-end server or servers. As the administrator, you must decide whether or not to send authentication information that is placed in the HTTP basic authentication header to the back-end server.

What is the source of this authentication information?

After the initial authentication between the client and WebSEAL, WebSEAL builds a new basic authentication header. The request uses this new header for its journey across the junction to the back-end server. You use the options that are provided by the **junctioncp** utility to dictate what authentication information is supplied in this new header.

You must analyze your network architecture and your security requirements, and then decide what header information (if any) needs to go across the junction.

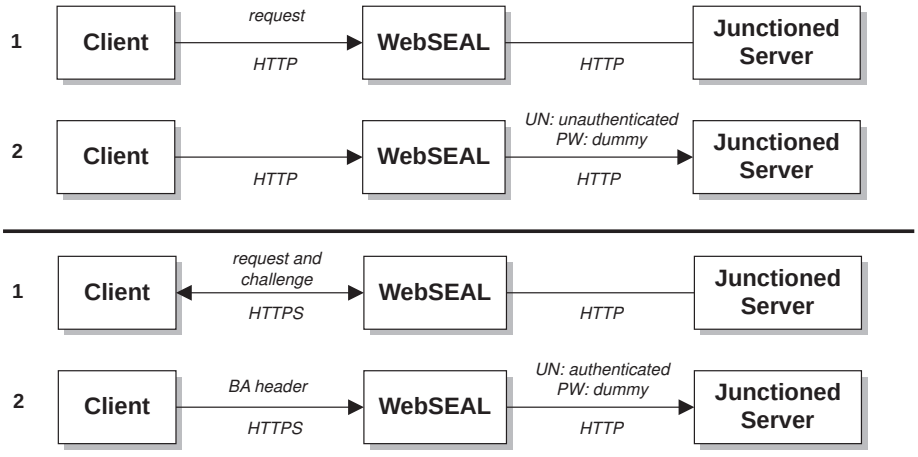


Policy Director identity and generic password

junctioncp option: -b supply

This option instructs WebSEAL to supply the authenticated Policy Director user name (client's original identity) with a generic (dummy) password. This scenario does not use the original client password.

The scenario assumes that the back-end server requires authentication from a Policy Director identity. By mapping a client user to a known Policy Director user, Policy Director manages authentication for the back-end server. Also, Policy Director provides a simple domain-wide, single sign-on solution.



Limitations:

Policy Director uses the same dummy password for all requests; all users have the same password in the back-end registry. If clients always go through WebSEAL to access the junctioned server, this arrangement will not present any security problems.

A generic password eliminates password administration and supports the application on a per-user basis. The **basic_auth_passwd** parameter of the iv.conf configuration file sets the dummy password.

```
basic_auth_passwd = password
```

Because this scenario has no password-level security, the back-end server must implicitly trust WebSEAL to verify the legitimacy of the client.

The server must also understand the Policy Director identity in order to accept it. This requires synchronization of the back-end server registry with the WebSEAL registry.

The use of the common dummy password offers no basis for the back-end server to prove the legitimacy of the client logging in with that user name. For this reason, it is important to also physically secure the back-end server from other possible means of access.

Original client BA header information

junctioncp option: -b ignore

This option instructs WebSEAL to ignore the basic authentication (BA) header that is supplied by the client. This option instructs WebSEAL to forward the header, without modification, to the third-party server. No login is performed on the WebSEAL server.

This scenario is appropriate for a back-end server that:

- Supports basic authentication
- Is not configured to use Policy Director security
- Must maintain client-supplied passwords

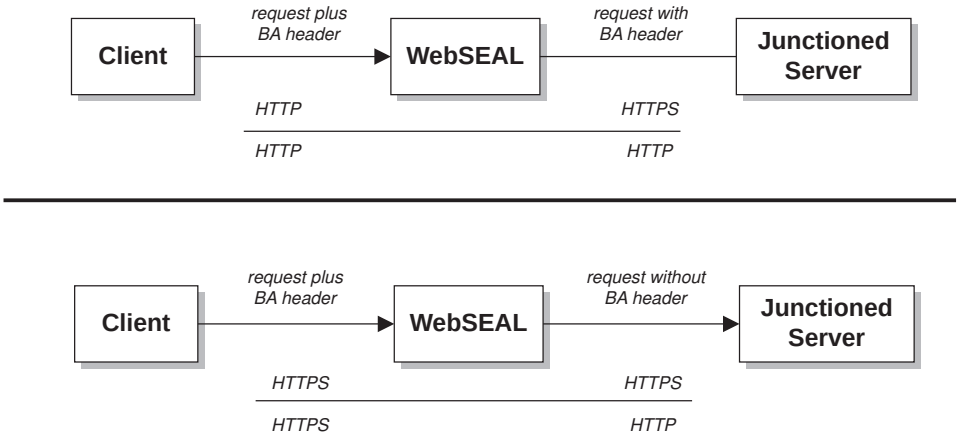
WebSEAL passes the original client request straight to the back-end server without interference.

The back-end server sends a basic authentication challenge back to the client. The client responds with username and password information, which the WebSEAL server passes through without modification.

This is not a true single sign-on mechanism—but rather a direct login to the third-party server, transparent to WebSEAL.

CAUTION:
This option does not operate when the client has authenticated to WebSEAL by using SSL (basic authentication). In this case, the basic authentication header is removed (as in -b filter) before it is sent across a potentially unsecured junction.

If the back-end server uses basic authentication, it will send a challenge back to the client. However, the client's returning basic authentication information would once again be removed. The request never reaches the back-end server.

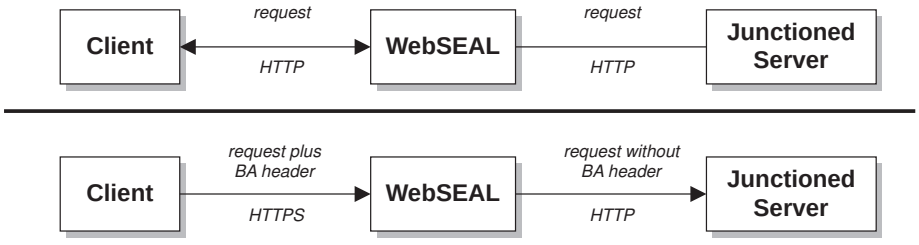


No authentication information

junctioncp option: -b filter

This option instructs WebSEAL to remove all basic authentication headers from any client requests before forwarding the requests to the back-end servers. WebSEAL becomes the single security provider. This option is appropriate when you know that the back-end server does not require basic authentication.

You can combine this option with the **-c** option to insert Policy Director client identity information into the HTTP headers. See "Inserting client identity information (-c option)" on page 198.



User names and passwords from GSO

junctioncp option: -b gso

This option instructs WebSEAL to supply the back-end server with authentication information (username and password) that is obtained from GSO. When you are interested in security at both the WebSEAL and back-end servers, this scenario is appropriate. The back-end server applications also require different user names and passwords that are not contained in the WebSEAL registry.

“Integrating GSO and WebSEAL single sign-on” fully describes this mechanism.

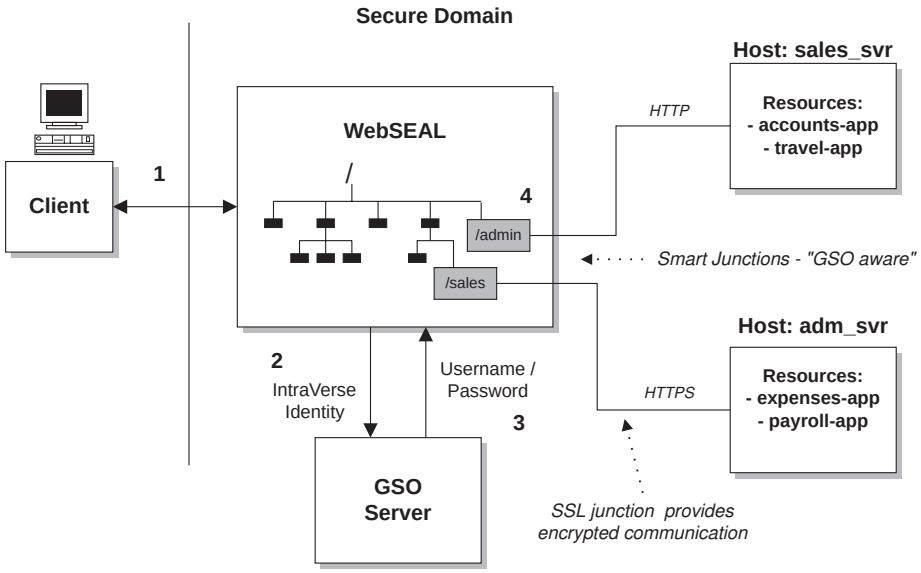
Integrating GSO and WebSEAL single sign-on

Policy Director supports a more flexible single sign-on solution by integrating with IBM Global Sign-On (GSO). IBM Global Sign-On is a component of IBM’s SecureWay technology. The combination of WebSEAL and GSO provides a complete single sign-on Web solution with the extra benefits of encrypted data, high availability, and scalability.

The figure that follows illustrates the integration of WebSEAL and GSO to retrieve user names and passwords for back-end application resources.

1. The client authenticates to WebSEAL with a request for access to an application resource on a back-end server. A Policy Director identity is obtained.

   Note: The single sign-on process is independent of the initial authentication method.
2. WebSEAL passes the Policy Director identity to GSO.
3. GSO returns a username and password appropriate for the user and the requested application resource.
4. WebSEAL inserts the username and password information in the HTTP basic authentication header of the request. WebSEAL sends the request across the junction to the back-end server.



Obtaining authentication information from GSO

The following example illustrates how GSO provides authentication information to WebSEAL. When the user Michael wants to run the travel-app application resource, WebSEAL asks GSO for Michael's authentication information.

GSO maintains a complete database of authentication information in the form of resources mappings to specific authentication information. The mapping of application resources to username and password combinations is known as *GSO resource credentials*. You can create GSO resource credentials for registered users only.

GSO contains a specific resource credential for Michael that maps the resource "travel-app" to a specific username and password combination.

The following table illustrates the structure of the GSO resource database:

Michael	**Paul**

`resource: travel-app`
`username=mike`
`password=123`
 	
`resource: payroll-app`
`username=powell`
`password=456`
	`resource: travel-app`
	`username=bundy`
	`password=abc`
	
	`resource: payroll-app`
	`username=jensen`
	`password=xyz`

In this example, GSO returns user name `mike` and password `123` to WebSEAL. WebSEAL uses this information when it constructs the HTTP basic authentication header in the request that is sent across the junction.

Configuring a GSO-enabled smart junction

Configure WebSEAL support for GSO at the smart junction between WebSEAL and a back-end application server.

To create a junction that enables GSO, use the **junctioncp create** command with the **-b gso** option. The following example illustrates the syntax for the create command:

`create -t tcp -h` *`hostname`* `-b gso -T` *`resource jct-point`*

The following table lists the options for setting up GSO smart junctions:

Options	**Description**
-b gso	Specifies that GSO should provide authentication information for all requests crossing this junction.
-T *resource*	Specifies the name of the application resource. The resource name used as the argument to this option must exactly match the resource name as listed in the GSO database. Required for GSO junctions.

As discussed in "Creating secure SSL smart junctions" on page 199, you can make a junction used in a WebSEAL and GSO solution secure through SSL. Make it secure by applying the **-t ssl** option when creating the junction.

Always use SSL junctions with GSO to ensure encryption of GSO resource credentials and all data.

Examples of GSO-enabled smart junctions:

Junction the application resource “travel-app” on the host “sales_svr” to junction point “/sales”:

```
create -t tcp -b gso -T travel-app -h sales_svr /sales
```

Junction the application resource “payroll-app” on the host “adm_svr” to junction point “/admin” and make the junction secure with SSL:

```
create -t ssl -b gso -T payroll-app -h adm_svr /admin
```

Note: In the above example, the **-t ssl** option dictates a default port of 443.

Using smart junctions

Each unique back-end server must be junctioned to a separate junction (mount) point. As the demand for additional content and resources grows, you can add more servers by using smart junctions.

These procedures related to using smart junctions:

- “Mounting multiple servers at the same junction”.
- “Filtering URLs through junctioned servers” on page 210.
- “Controlling CGI processing (x permission)” on page 211

Mounting multiple servers at the same junction

You can mount multiple replicated servers at the same junction point. There can be any number of servers that are mounted at the same point.

All servers mounted at one junction point must be replicas (mirrored Web spaces), and must use the same protocol—HTTP or HTTPS. Do not mount dissimilar servers on the same junction point.

From the primary Policy Director server Web space, access pages that belong to the junctioned servers. You should be able to access these pages (subject to permissions, of course), and the pages should appear consistent. Occasionally, when you cannot find a page or when the page changes, it means that Policy Director did not replicate that page properly.

Check that the document exists and is identical in the document tree of both replicated servers.

Filtering URLs through junctioned servers

Only documents of mime type “text” or “html” that are received from junctioned servers are filtered.

There are two sets of URLs that WebSEAL can change: Absolute and Host Relative.

Host relative URLs

Host relative URL indicates a URI position in relation to the document root of the junctioned server, for example:

```
/dir/file.html
```

Change these URLs to reflect the junction point of the junctioned server. For example:

```
/jct/dir/file.html
```

Absolute URLs

Absolute URL indicates a Universal Resource Indicator (URI) position in relation to both a HOST name or IP address, and a network port. For example:

```
http://servername[:port]/file.html
```

Or:

```
https://servername[:port]/file.html
```

You can change these URLs according to the following set of rules:

1. When the URL is HTTP and the host or port matches a TCP junctioned server, the URL changes to reflect the junction point. For example:

   ```
   /jct/...
   ```

2. When the URL is HTTPS and the host or port matches an SSL junctioned server, the URL changes to reflect the junction point. For example:

   ```
   jct...
   ```

3. Only URLs for TAG and attribute pairs that are defined in the iv.conf configuration file are filtered.
4. Always filter META tags for refresh requests. For example:

   ```
   META HTTP-EQUIV="Refresh" CONTENT="5;URL=http://server/url"
   ```

5. If a BASE tag contains an HREF attribute, the tag will be removed from the response to the client.

The `[url-filter]` stanza of the iv.conf configuration file contains parameters for filtering URLs through junctioned servers.

The `[url-filter]` stanza contains a list of HTML tags. The WebSEAL server filters or changes these tags to adjust absolute URLs that are obtained through a junctioned server.

By default, Policy Director configures all commonly used HTML tags. The administrator might need to add additional HTML tags that contain URLs.

Controlling CGI processing (x permission)

The Policy Director execute (x) permission has no meaning across junctions. You cannot control the processing of a CGI script with the x permission. WebSEAL has no means of accurately determining whether or not a requested object on a back-end server is a CGI program file or a regular HTTP object. Access to objects, which include CGI programs, across junctions is always controlled using the read (r) permission.

Using query_contents with third-party servers

You can protect the resources of the third-party application Web space by using the Policy Director security service. When you want to, you must provide WebSEAL with information about the contents of the third-party Web space.

A CGI program called **query_contents** provides this information. The **query_contents** program searches the third-party Web space contents and provides this inventory information to the Management Console on WebSEAL. The program comes with the WebSEAL installation, but you must manually install the program on the third-party server. There are different program file types available, depending on whether you base the third-party server on UNIX or on Windows.

The Object Space manager of the Management Console can automatically run **query_contents**. It runs it any time the portion of the protected object space, which represents the junction, is expanded in the Object Space management panel. Now that the Console knows about the contents of the third-party application space, you can display this information and apply policy templates to appropriate objects.

Installing query_contents

Installing **query_contents** involves copying one or two files from the Policy Director server to the third-party server and editing a configuration file.

The following Policy Director directory contains a template of the program:

UNIX: *root-directory*/www/lib/query_contents

Windows: *root-directory*\www\lib\query_contents

The contents of the directory includes:

File	Description
query_contents.exe	Main executable program for Win32 systems. Should be installed in the cgi-bin directory of the third-party Web server.
query_contents.sh	Main executable program for UNIX systems. Should be installed in the cgi-bin directory of the third-party Web server.
query_contents.c	Source code. The source is provided in case you need to change the behavior of **query_contents**. In most cases, this is not necessary.
query_contents.html	Help file in HTML format.
query_contents.cfg	A sample configuration file which identifies the document root for the Web server.

Installing query_contents on third-party UNIX servers

Locate the shell script that is named query_contents.sh in the following directory:

UNIX: *install-dir*/www/lib/query_contents

To install the **query_contents** utility on third-party UNIX servers:

1. Copy query_contents.sh into a functioning /cgi-bin directory on the third-party Web server.
2. Remove the .sh extension.
3. Set the UNIX “execute” bit for the user that owns the Web server.

Installing query_contents on third-party Win32 servers

Locate the executable program that is named query_contents.exe and the configuration file that is named query_contents.cfg in the following directory:

Windows: *install-dir*\www\lib\query_contents

To install the **query_contents** utility on third-party Win32 servers:

1. Ensure that the third-party Web server has a CGI directory correctly configured.
2. Ensure that a valid document exists in the document root of the third-party Web server.
3. Copy query_contents.exe into the CGI directory of the third-party Web server.
4. Copy query_contents.cfg into the Windows directory.

   The following table shows the default values for this directory:

Operating System	Windows Directory
Windows 95	c:\windows
Windows NT 3.5x	c:\winnt35
Windows NT 4.x	c:\winnt

5. Edit the query_contents.cfg file to correctly specify the document root directory for the third-party Web server.

   The file currently contains sample entries for the Microsoft Internet Information Server and Netscape FastTrack servers. The lines in this file that start with a semicolon (;) are comments, and ignored by the **query_contents** program.

Testing the configuration

To test the configuration:

1. On the Win32 server, change to the directory containing the **query_contents** program.
2. From a DOS prompt on the Win32 server, run the program:

   ```
   query_contents dirlist=/
   ```

   You should see something similar to the following output:

   ```
   100
   index.html
   cgi-bin//
   pics//
   ```

   The number 100 is a return status that indicates success. It is most important to see at least the number 100 as the first (and perhaps only) value.

When you see an error code instead, then the configuration file is not in the correct place or does not contain a valid document root entry. Check the configuration of the query_contents.cfg file and make sure that the document root exists.

3. From a browser, enter the following URL:

   ```
   http: //Win32 machine name/cgi-bin/query_contents.exe?dirlist=/
   ```

   This command should return the same result as the preceding step. If it does not return this result, the CGI configuration of your Web server will not be correct. See the server's documentation to correct the problem.

Configuring a smart junction to find query contents

You can defines the URL for **query_contents** script. Policy Director looks for **query_contents** in /cgi_bin/. When this directory is different or the **query_contents** file is renamed, use this option to indicate to WebSEAL the new URL to the file.

If you create the smart junction for a third-party Win32 server, use the **junctioncp** command with the **-q** option:

```
junctioncp> create -t tcp -h hostname -q /cgi-bin/query_contents.exe /jct_mount_point
```

For a summary of all the **junctioncp** command options, see "Creating a new junction for an initial server" on page 193.

Performing query_contents

Use **query_contents** to return the contents of directories that are included in a URL request. For example, to get the contents of the root directory for a server's Web space, the browser runs **query_contents** on a URL such as:

```
http://third-party-server/cgi-bin/query_contents?dirlist=/
```

The **query_contents** script performs the following actions:

1. Reads $SERVER_SOFTWARE, a standard CGI environment variable, to determine the server type.

   Policy Director sets the variable $DOCROOTDIR to a typical document root location, based on the Web server type.

2. Reads the environment variable $QUERY_STRING from the requested URL to obtain the requested operation and get the object path.

   The variable $OPERATION stores the operation value, and $OBJPATH stores the option path. The $OPERATION is dirlist, and $OBJPATH is " / " in the example.

3. Performs a directory listing (ls) on the object path and places the results on standard output, for use by the Policy Director server. Entries that indicate subdirectories have a double slash (//) appended to them.

   Typical output looks similar to:

   ```
   100
   index.html
   cgi-bin//
   pics//
   ```

   The number 100 is a return status that indicates success.

Customizing query_contents

To customize **query_contents** for your server, you might need to change the document root directory setting.

If **query_contents** returns an error status (a number other than 100) and lists no files, examine the script. Change the $DOCROOTDIR variable, if needed, to match your server's configuration.

If you specify the document root directory correctly and the script still fails, the cgi-bin location specification might be incorrect. Examine the $FULLOBJPATH variable and change the value that is assigned to it to reflect the correct cgi-bin location.

Additional Functionality

The source code for the **query_contents** program (query_contents.c) is distributed with Policy Director.

You can add additional functionality to this program to support special features of some third-party Web servers. These features include:

1. Directory mapping—where a subdirectory not below the document root is mapped into the Web space.
2. Generation of a Web space that is not file system based.

   This might be the case for a database-hosted Web server.

Chapter 16. WebSEAL: Application integration

WebSEAL supports third-party application integration through environment variables and dynamic URL capability. WebSEAL extends the range of environment variables and HTTP headers to enable third-party applications to perform operations that are based on a client's identity. In addition, WebSEAL can provide access control on dynamic URLs, such as those that contain query text.

This chapter includes:

Supporting CGI programming

To support CGI programming, WebSEAL adds three additional environment variables to the standard set of CGI variables. CGI applications can use these environment variables that run on either the local WebSEAL server or a junctioned back-end server. The variables provide Policy Director-specific user, group, and credential information to the CGI application.

On a local WebSEAL server, the Policy Director credential information of the client that makes the request directly produces these environment variables.

Environment variables, which are used by a CGI application that run on a junctioned third-party server, are produced from the HTTP header information. WebSEAL passes the HTTP header information to the server. You must use the **junctioncp -c** option to create a *junction*. The junction then inserts Policy Director-specific header information into the HTTP requests that are destined for a back-end server.

See "Inserting client identity information (-c option)" on page 198.

Additional Policy Director-specific environment variables

Following are additional Policy Director-specific CGI environment variables formats:

CGI Environment Variable Format	Description
HTTP_IV_USER	The Policy Director user account name of the requester.
HTTP_IV_GROUPS	The Policy Director groups to which the requester belongs. Specified as a comma-separated list of double-quote enclosed group names.
HTTP_IV_CREDS	Encoded opaque data structure representing a Policy Director credential. Supplies credentials to remote servers so mid-tier applications can use the Authorization API to call the Authorization Service. Refer to the *Policy Director Programmer's Guide and Reference*.

The REMOTE_USER variable on a local WebSEAL server

On a local server environment that is controlled by WebSEAL, the value of the HTTP_IV_USER variable is provided as the value for the standard REMOTE_USER variable. Note that the REMOTE_USER variable can also be present in the environment of a CGI application that runs on a junctioned back-end server. However, in this situation, WebSEAL does not control its value.

CGI Environment Variable Format	Description
REMOTE_USER	Contains the same value as the HTTP_IV_USER field.

Supporting back-end server-side applications

WebSEAL also provides support for executable code that runs as an embedded component of a back-end Web server. Examples of such server-side executable code include:

- Java servlets
- Cartridges for Oracle Web Listener
- Server-side plugins

You can create a junction to a back-end server by using the **-c** option of the **junctioncp** utility. Then, WebSEAL inserts Policy Director-specific client identity and group membership information into the headers of the HTTP requests that are destined for that server.

See “Inserting client identity information (-c option)” on page 198.

The Policy Director-specific HTTP headers enable applications on junctioned, third-party servers to perform user-specific actions that are based on the client’s Policy Director identity.

WebSEAL provides the following Policy Director-specific HTTP header information:

Policy Director-specific HTTP Headers	Description
iv-user	The name of the client. Defaults to “Unauthenticated” if client is unauthenticated (unknown).
iv-groups	A list of groups to which the client belongs. Consists of comma-separated list of double quote-enclosed group names.
iv-creds	Encoded opaque data structure representing a Policy Director credential. Supplies credentials to remote servers so mid-tier applications can use the Authorization API to call the Authorization Service. Refer to the *Policy Director Programmer’s Guide and Reference*.

These HTTP header entries are available to CGI applications as the environment variables HTTP_IV_USER, HTTP_IV_GROUP, and HTTP_IV_CREDS. For other non-CGI application frameworks, see their associated product documentation for instructions on extracting headers from HTTP requests.

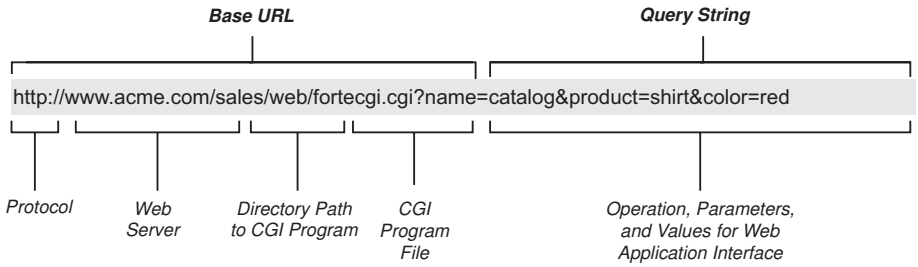
Providing access control to dynamic URLs

The current Web environment gives users immediate access to rapidly changing information. Many Web applications dynamically generate Uniform Resource Locators (URLs) in response to each user request. These dynamic URLs might exist only for a short time. Despite their temporary nature, dynamic URLs still need strong protection from unwanted use or access.

Understanding dynamic URLs

Some sophisticated Web application tools use standard Web browsers to communicate with application servers through the CGI interface of a Web server.

All these tools use dynamic URLs and hidden form elements to communicate the requested operation (with its parameter value) to the application server. A dynamic URL augments the standard URL address with information about the specific operation and its parameter values. The query string portion of the URL provides operations, parameters, and values to the Web application interface.



Mapping ACL namespace objects to dynamic URLs

WebSEAL uses the protected object namespace model and the policy templates (ACLs) to secure dynamically generated URLs, such as those URLs that are generated by database requests. Each request to WebSEAL is resolved to a specific namespace object as the first step in the authorization process. An ACL applied to the namespace object dictates the required protection on any dynamic URL mapped to that object.

Because dynamic URLs exist only temporarily, it is not possible to have entries for them in a pre-configured authorization policy database. WebSEAL solves this problem by providing a mechanism that maps many dynamic URLs to a single static protected object.

A text file contains the mappings from objects-to-namespace patterns:

UNIX: `/opt/intraverse/www/lib/dynurl.conf`

Windows: `C:\Program Files\IBM\Policy Director\www\lib\dynurl.conf`

Edit this file to change these mappings. You must create this file; the file does not exist by default. Entries in the file are of the format:

object pattern

WebSEAL uses a subset of UNIX shell pattern (including wildcards) that define the set of parameters that constitute one object in the namespace. WebSEAL maps any dynamic URL that matches those parameters to that object. WebSEAL supports the following UNIX shell pattern-matching characters:

Character	Description
****	The character that follows the backslash is part of a special sequence. For example, it is the TAB character.
?	Wildcard that matches a single character. For example, the string `abcde` is matched by the expression `ab?de`.
*****	Wildcard that matches zero or more characters.
[ ]	Defines a set of characters, from which any can match. For example, the string `abcde` is matched with the regular expression `ab[cty]de`.
^	Indicates a negation. For example, the expression ^[ab] matches anything but the 'a' or 'b' characters.

The following example illustrates the form of a dynamic URL that performs credit balance lookup:

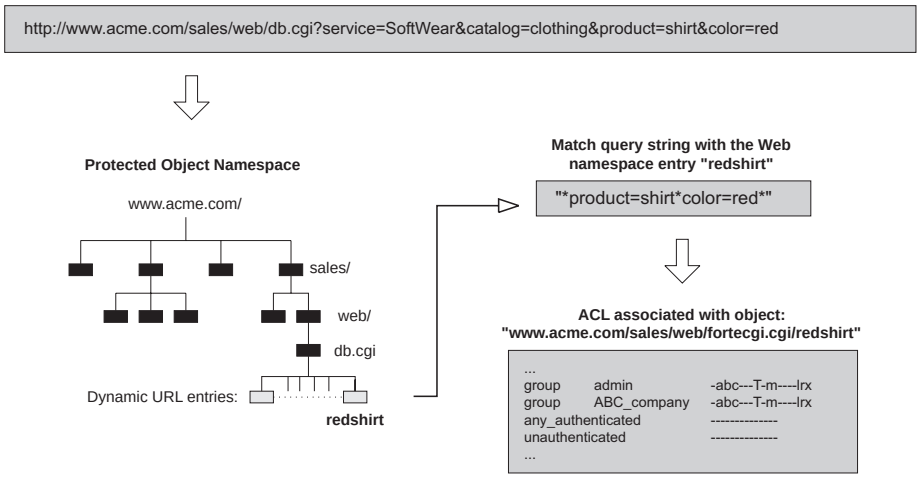
`http: //`*`server-name`*`/home-bank/owa/acct.bal?acc=`*`account-number`*

The namespace object that represents this dynamic URL would appear as follows:

`http: //<server-name>/home-bank/owa/acct.bal?acc=*`

Careful examination of the dynamic URL in this example shows that it describes a specific account number. The namespace object, for account balances at `home-bank`, shows that the ACL permissions apply to any account. They apply to any account because the last portion of the entry (`acc=*`) uses the asterisk wildcard, which matches all characters.

This figure illustrates a complete scenario of a specific dynamic URL mapped to a specific protected namespace object—this example does not use any wildcards:



Updating WebSEAL for dynamic URLs

Use the **dynurlcp** utility to update the WebSEAL protected object namespace with entries that are made in the dynurl.conf configuration file. You must log in to the secure domain by using dce_login.

To use the **dynurlcp** utility to update the WebSEAL protected object namespace:

1. Create, edit, or delete a dynamic URL entry in the dynurl.conf configuration file.
2. After making your changes, use the **dynurlcp** utility to update the server:

   ```
   dynurlcp -e /.:/subsys/intraverse/secmgr/server/ host update
   ```

Resolving dynamic URLs in the namespace

The resolving of a dynamic URL to a namespace object depends on the ordering of the entries in the dynurl.conf configuration file.

When attempting to map a dynamic URL to a namespace entry, the list of mappings in the dynurl.conf configuration file is scanned. The file is scanned from top to bottom, until the first matching pattern is found.

WebSEAL uses the corresponding namespace entry for the subsequent authorization check when the first match is found. When no matches are found, WebSEAL uses the URL itself, minus the `http://server` portion of the path.

Keep the mappings which correspond to the most restrictive ACLs higher up in the list. For example, the book sale procedure (of an sales order application) is restricted to just a book club group. However, the rest of the sales order application can be accessed by all users. Then, the mappings should be in the order that is shown in this table:

Namespace Entry	URL Pattern
/ows/sales/bksale	/ows/db-apps/owa/book.sales*
/ows/sales/general	/ows/db-apps/owa/*

Note that if the mapping entries were in the reverse order, all stored procedures in the /ows/db-apps/owa directory would map to the /ows/sales/general namespace object. This could lead to possible breaches of security, because of this incorrect namespace resolution.

GET and POST methods of data transmission

When you map a URL regular expression to a namespace entry, the URL format should assume the format as produced from the GET method. For this assumption, it does not matter whether the POST or GET method is being used.

In the GET method of data transmission, WebSEAL appends the *dynamic data* to the URL. An example of dynamic data is the data that is supplied for a user in a form.

In the POST method of data transmission, WebSEAL includes the dynamic data in the body of the request.

ACL evaluation

After the dynamic URL has been resolved to a namespace entry, the standard Policy Director ACL inheritance model is used. This model determines whether the request should be processed or forbidden (because of insufficient privilege).

Illustrating a dynamic URL: The Travel Kingdom

The following example illustrates how a corporate intranet can secure URLs that are generated by an Oracle Web Listener.

The dynamic URL Web server used in this example is the Oracle Web Listener. You can apply this technology equally to other dynamic URL Web servers.

The application

Travel Kingdom is an organization that offers clients a travel booking service over the Internet. The business intends to operate two Oracle database applications on their Web server—accessible from within the corporate firewall and from across the Internet.

- **Travel booking system**

  Authorized customers can make bookings remotely and query their own current bookings. Travel Kingdom staff can also make bookings for telephone customers, process changes, and perform many other transactions. Because external customers pay for services with credit cards, WebSEAL must strongly secure the transmission of that information.
- **Administration Manager**

  Like most other companies, Travel Kingdom maintains an administration database that contains salary, position, and experience information. A photograph of each staff member accompanies this data.

The interface

You can configure an Oracle Web server to provide access to the following stored procedures in the database:

/db-apps/owa/tr.browse	Gives all users the ability to inquire about travel destinations, prices, and so forth.
/db-apps/owa/tr.book	Used to place a booking (travel agent staff or authenticated customers).
/db-apps/owa/tr.change	Used to review or change current bookings.
/db-apps/owa/admin.browse	Used by any staff member to view unrestricted staff information, such as extension number, email address, and photograph.
/db-apps/owa/admin.resume	Gives a staff member the ability to view or change their resume information in the Administration database.
/db-apps/owa/admin.update	Used by Administration staff to update information on staff.

Web space structure

Use a WebSEAL server to provide a secure interface to the unified Web space of Travel Kingdom.

You can make a junction (/ows) to the Oracle Web server that runs both the travel booking application and the administration application.

The security policy

To provide suitable security to Web resources, while keeping an easy-to-use system, the business has established the following security goals:

- Travel agent staff members have complete control over all bookings.
- Authenticated customers can make and change their own bookings, but cannot interfere with the travel data of other authenticated customers.
- Administration staff members have complete access to all of the administration information.
- Travel Kingdom staff other than the Administration Department can change their own resume information and view partial information of other staff members.

Dynamic URL to namespace mappings

To achieve the security policy goals, the mappings from dynamic URLs to ACL namespace entries need to be configured. Remember that the ordering of these mappings is an important part of achieving the security goals.

The mappings should be configured as shown in this table:

Namespace Entry	URL Pattern
/ows/tr/browse	/ows/db-apps/owa/tr.browse\?dest=*&date=??/??/????
/ows/tr/auth	/ows/db-apps/owa/tr.book\?dest=*&depart=??/??/????&return=??/??/????
/ows/tr/auth	/ows/db-apps/owa/tr.change
/ows/admin/forall	/ows/db-apps/owa/admin.resume
/ows/admin/forall	/ows/db-apps/owa/admin.browse\?empid=[th]???
/ows/admin/auth	/ows/db-apps/owa/admin.update\?empid=????

The secure clients

Client authenticates to WebSEAL over a secure, encrypted channel.

Customers who want to use the Web interface must additionally register with the Travel Kingdom Web master to receive an account.

Account structure and group structure

Create the following groups on the system:

Staff	Members of the Travel Kingdom organization.
TKStaff	Travel Kingdom travel agents.
AdminStaff	Members of the Travel Kingdom Administration Department. Note that Administration staff members are also in the Staff group.
Customer	Customers of Travel Kingdom who want to make their travel bookings across the Internet.

Give each user an account in the secure domain so that the WebSEAL server can identify them individually. WebSEAL passes the user's identity to the Oracle Web servers and provides a single sign-on solution to all of the Web resources.

The access control

The following table lists the access controls that result from application of the preceding information:

/ows/tr/browse	unauthenticated	Tr
	any_authenticated	Tr
/ows/tr/auth	unauthenticated	-
	any_authenticated	-
	group TKStaff	Tr
	group Customer	PTr
/ows/admin/forall	unauthenticated	-
	any_authenticated	-
	group Staff	Tr
/ows/admin/auth	unauthenticated	-
	any_authenticated	-
	group AdminStaff	Tr

Customers and TKStaff have the same privileges on the booking and travel plan maintenance objects, with one exception. The one exception is that the customers must encrypt information (privacy permission) to give them further security when submitting *sensitive data*across the untrusted Internet. Sensitive data includes credit card information, for example.

The conclusion

This simple example illustrates the concepts of deploying a system that is capable of:

- Securing sensitive information
- Authenticating users
- Authorizing access to sensitive information

In addition, the identities of the authenticated users of the system are known to both the WebSEAL and Oracle Web servers. The identities are used to provide an auditable, single sign-on solution.

Chapter 17. NetSEAL: Overview

Policy Director NetSEAL is a virtual private network (VPN) solution for securing incoming TCP/IP communication. As a resource manager, NetSEAL controls a user's ability to connect to a specific TCP application. NetSEAL performs access control that is based on the destination port and the client identity. NetSEAL integrates any network application server with the Policy Director Security Services.

This chapter includes:

Introducing NetSEAL

NetSEAL allows controlled access to TCP-based applications and services in a Policy Director secure domain. Windows clients have secure communication to NetSEAL through the Policy Director NetSEAT client.

Communications between NetSEAT and NetSEAL can be secured using either an SSL tunnel or a GSS tunnel. In each scenario, the secure link established between a NetSEAT client and a Policy Director server is authenticated using the requesting client's username and password.

- Use an *SSL channel* to secure communication between NetSEAT and NetSEAL.
- Use *GSS tunnels* or NetSEAL junctions to secure NetSEAL server-to-NetSEAL server communication. GSS tunnels established between Policy Director servers are always authenticated using the Policy Director server user that is making the connection on behalf of the requesting client. Through the GSS tunnel, the two servers mutually authenticate each other; the second server always performs access control on the client.

NetSEAL junctions establish the forwarding direction for communication through a Policy Director server to another Policy Director server or network. Use a GSS tunnel to secure communication across a NetSEAL junction.

NetSEAL access control is coarse-grained. There is control up to the specific port where the application is listening. Resources manipulated by the application do not benefit from any fine-grained access control.

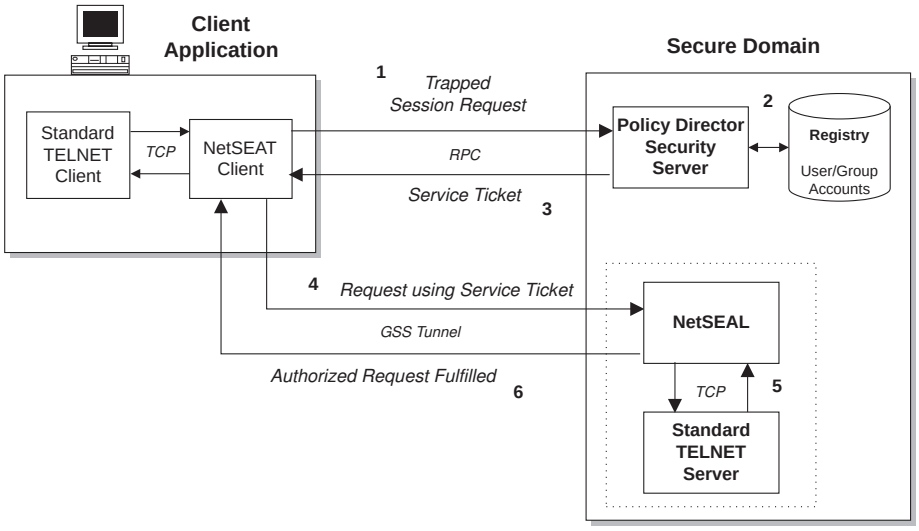
This chapter describes a number of network scenarios that use NetSEAL. Use the Telnet remote login application as an example of a TCP application in the diagrams. In each scenario, NetSEAL responds to a request that is based on:

- The source of the request
- The permissions on affected objects (such as destination ports and NetSEAL junctions)
- The principals involved in the connection

NetSEAT Client to NetSEAL over a GSS tunnel

When NetSEAT is configured to use a GSS tunnel, NetSEAT traps the outgoing request and, using RPC, authenticates the client to the Policy Director Security server. In addition, an authorization check is performed for the specific port associated with the requested application.

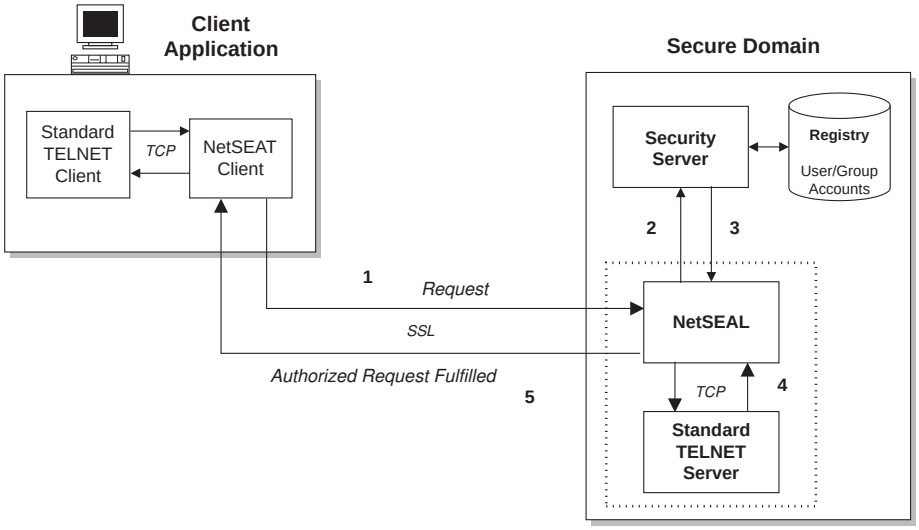
If the authentication and authorization process proves favorable, a GSS tunnel will be established between NetSEAT and the NetSEAL server. Access to the requested TCP application from NetSEAL is made using TCP.



NetSEAT client to NetSEAL over SSL

When you configure NetSEAT to use SSL, authentication and authorization are handled between NetSEAL and the Policy Director security services. NetSEAL can accept a username and password for the client identity.

If the authentication and authorization process proves to be favorable, Policy Director will process the request. Use TCP to access the requested TCP application from NetSEAL.

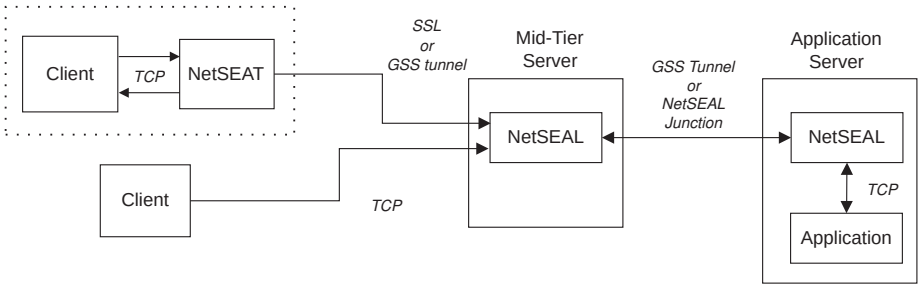


NetSEAL network segments

The connection request in a NetSEAL transaction experiences different levels of protection along the network route. As described in “NetSEAT client to NetSEAL over SSL” on page 224, the NetSEAT client supports an SSL or GSS connection to the NetSEAL server.

Note: By design, NetSEAL server-to-NetSEAL server connectivity is *always* GSS. Do not use an SSL connection between two NetSEAL servers.

The final route from NetSEAL to the local or remote TCP application port is always using TCP.



The following table summarizes the level of protection for each connection segment:

Connection Segment	Protection
NetSEAT client-to-NetSEAL server	SSL or GSS tunnel
TCP client-to-NetSEAL server	None
NetSEAL server-to-NetSEAL server	GSS tunnel
NetSEAL junctions	GSS tunnel
NetSEAL server-to-TCP application port	None

NetSEAL follows the same connection-decision process for both local and remote applications:

- Is the requested port protected (with an ACL)?
- For a protected port, does the user have the proper permission for access?
- For an unprotected port, allow the outgoing connection.

NetSEAL also separates the issues of incoming information that is required by the Security Manager and of outgoing connection processing. In other words, the Security Manager does not need to know whether the NetSEAT client traps the connection request locally or remotely.

Illustrating client-to-NetSEAL services

The following scenarios illustrate the types of interactions possible between a client and a NetSEAL-protected TCP application. Clients can include NetSEAT clients and non-NetSEAT clients.

Scenarios include:

- “Incoming tunneled connection to Policy Director server” on page 226.
- “Incoming tunneled connection to protected host” on page 226.
- “Incoming TCP connection to Policy Director server” on page 227.

Incoming tunneled connection to Policy Director server

In this first basic scenario, you configure a NetSEAT client to trap outgoing connections that are destined for an application on the Policy Director server. After the NetSEAT client traps the communication, a secure tunnel is established with NetSEAL on the Policy Director server. The request in this example that is destined for port 23 is forwarded over this tunnel.

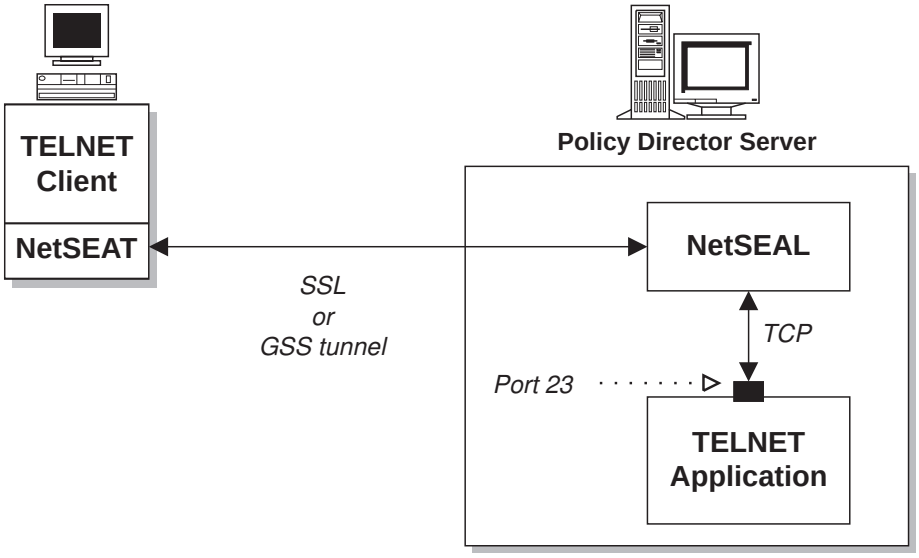
The authentication process is transparent to the user.

The NetSEAL server completes the transaction in the following manner:

1. Based on port's ACL, can the user connect to the requested port?

   Yes—Establish a TCP connection to the requested port.

   No—Reject the connection request.



Incoming tunneled connection to protected host

This scenario considers an application that resides on a protected remote server. The application server can be a Policy Director or a third-party server. Policy Director always uses GSS for communication between two NetSEAL servers.

In this scenario, NetSEAL can protect TCP applications that run on platforms that Policy Director does not support.

The NetSEAL server completes the transaction in the following manner:

1. Can the user connect to the requested port on the destination server (based on its ACL)?

   Yes—Continue.

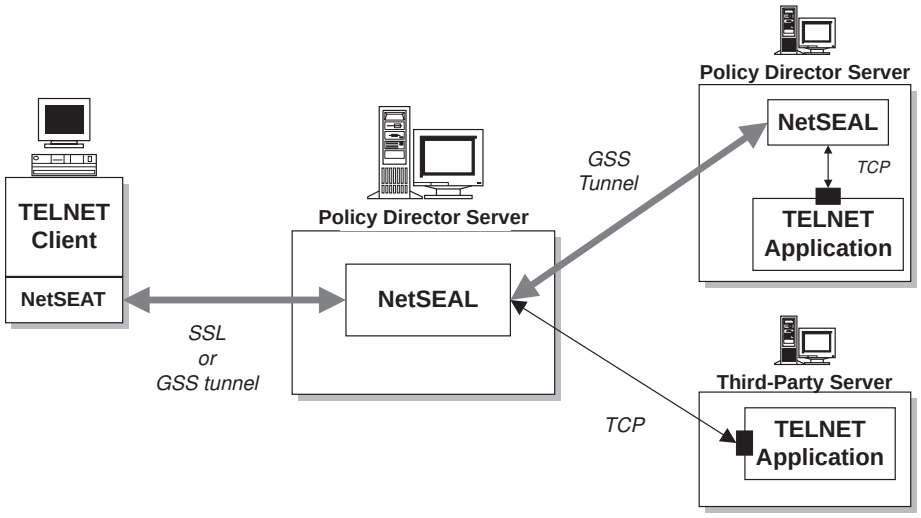
   No—Reject the connection request.

2. Is the destination a Policy Director server?

   Yes—Establish a secure tunnel to the server. Establish a TCP connection to the requested port.

   No—Establish a TCP connection (nonsecure) to the requested port.

A third-party application server always has an unprotected TCP connection. Use such a configuration within a trusted network environment.



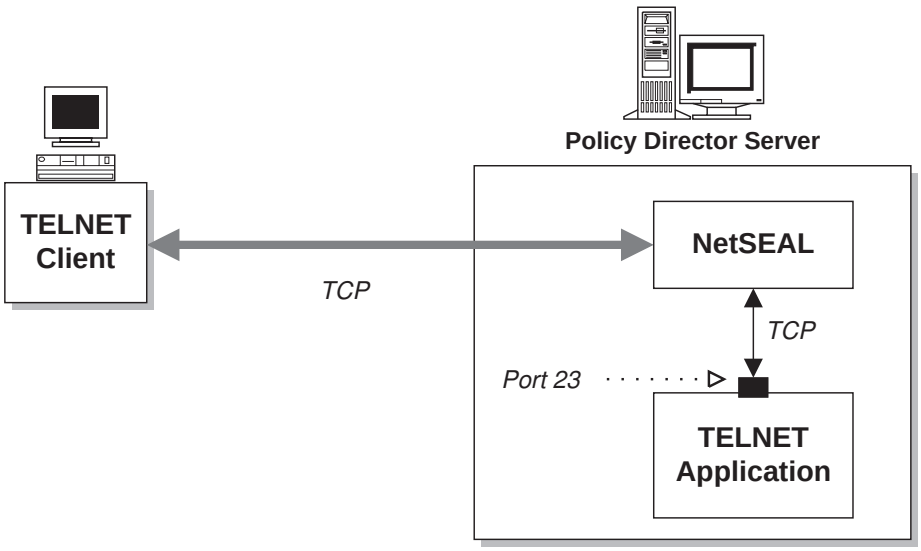
Incoming TCP connection to Policy Director server

This scenario considers the situation for a non-NetSEAT TCP client user. Policy Director recognizes such a client as unauthenticated. If the requested port is unprotected (no ACL), Policy Director will allow access to the port. If an ACL protects the port, the Security Manager will check the ACL for unauthenticated access.

This configuration protects direct access to network services. An external authorization service could use the client's IP address to determine access rights.

The NetSEAL server completes the transaction in the following manner:

1. Is the request trapped by Policy Director (ACL on port)?
   Yes—Pass request to Security Manager (secmgrd).
   No—Allow incoming connection.
2. Are unauthenticated requests on the port permitted?
   Yes—Establish a TCP connection to the requested port.
   No—Reject the connection request.



Illustrating NetSEAL-to-NetSEAL services

The following scenarios illustrate the types of interactions possible between two servers. The first NetSEAL server (with a local client) initiates the connections in these scenarios rather than initiating them from a remote NetSEAT client.

The first NetSEAL server (with a local client) initiates the connections in these scenarios rather than initiating from a remote NetSEAT client. This local client could operate from the server's console or Telnet in to this server from a remote location. A backend NetSEAL server can protect the TCP application.

Scenarios include:

- "Outgoing connection to Policy Director server".
- "Outgoing connection to protected host" on page 229.

Outgoing connection to Policy Director server

Use a GSS tunnel to make a connection between two Policy Director NetSEAL servers. The first NetSEAL server (with a local client) initiates the connection rather than initiating from a remote NetSEAT client. Policy Director can permit or deny a connection and protect any permitted communication.

The Policy Director server completes the transaction in the following manner:

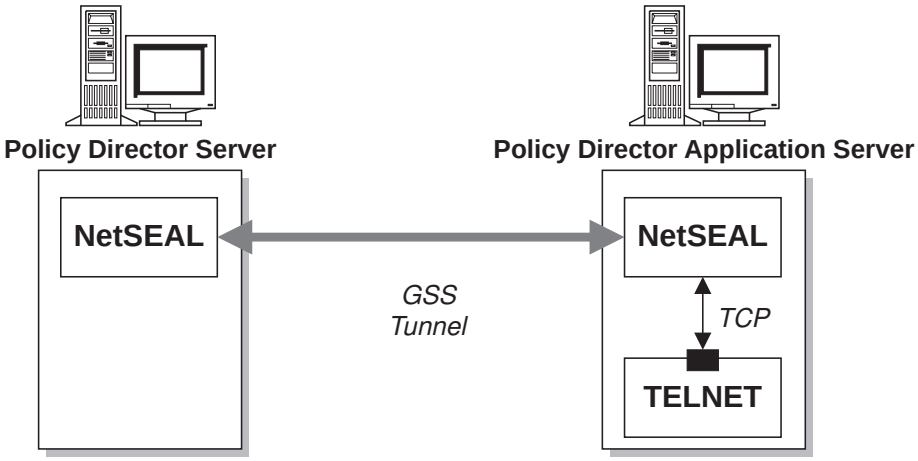
1. Is the requested port on the destination machine protected (ACL)?

   Yes—Pass request to Security Manager (secmgrd).

   No—Allow outgoing connection.
2. Are unauthenticated requests on the port permitted?

   Yes—Establish a secure tunnel to the server. Establish a TCP connection to the requested port.

   No—Reject the connection request.



Outgoing connection to protected host

You can use TCP to make a connection between a Policy Director NetSEAL server and a third-party server. However, any connection made over TCP is not a secure connection. There is no NetSEAL server on the backend third-party server. Policy Director can only permit or deny a connection to the back-end server; Policy Director cannot secure the communication across the connection.

The Policy Director server completes the transaction in the following manner:

1. Is the requested port on the destination machine protected (ACL)?

   Yes—Pass request to Security Manager (secmgrd).

   No—Allow outgoing connection.
2. Does the user have permission to access the requested port on the destination server?

   Yes—Continue.

   No—Reject the connection request.
3. Are integrity and privacy set on the port?

   Yes—Reject the connection request.

   No—Establish a TCP connection to the requested port.

Introducing NetSEAL junctions

NetSEAL junctions provide a mechanism for securely forwarding communications through a network of Policy Director servers to a destination server or network. NetSEAL junctions determine the direction of packet forwarding through a Policy Director server.

NetSEAL junctions are unidirectional static routes. Unidirectional junctions permit the manager of each Policy Director server to better control access to their networks. Each direction of travel requires a separate NetSEAL junction. Data flow across a NetSEAL junction, however, is always bidirectional.

A GSS tunnel secures communications across a NetSEAL junction. The final Policy Director server in the communications path then uses a TCP connection to the destination port. This destination port might be on the Policy Director server itself.

NetSEAL junctions provide data protection and security for communications through an organization. You can partition an organization on a geographical or functional basis. For example, NetSEAL junctions are useful in situations where an untrusted network exists between two geographically separated Policy Director servers. Also, these two Policy Director servers are members of the same secure domain.

Each junction has a source Policy Director server, a destination, and a routing direction. The destination can be another Policy Director server or a network specification. Junctions allow easy firewall configuration because all traffic across the junction requires only one route through the firewall.

Configuring NetSEAL junctions

An administrator can create NetSEAL junctions by using the **ivadmin** utility. The **ivadmin** utility includes commands to add, delete, and list NetSEAL junctions. You can create junctions between Policy Director servers or between a Policy Director server and a network.

Refer to “Appendix A. Policy Director administration using ivadmin” on page 267.

NetSEAL junctions and access control

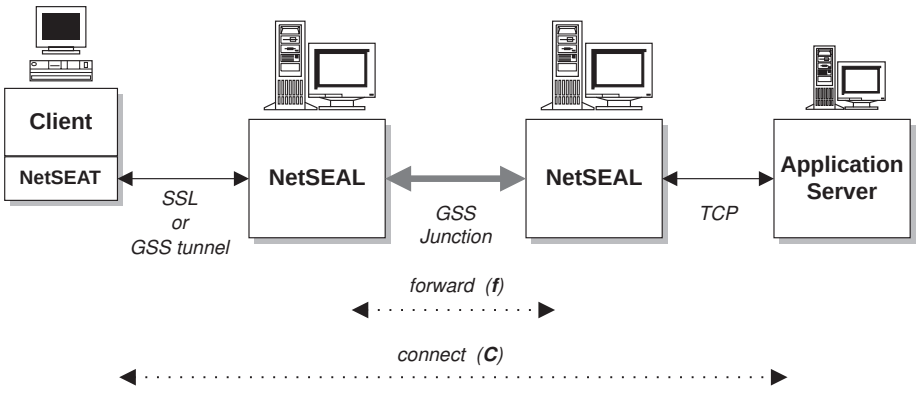
NetSEAL recognizes two ACL permissions for controlling access to a port on an application server.

The ACL on a destination port object controls access to that port. An ACL entry must contain the connect (C) permission to allow the user or group access to the port.

An ACL on the Policy Director server, responsible for the outgoing connection, controls traversal across a NetSEAL junction. An ACL entry must contain the forward (f) permission to allow the user or group access across the junction.

Use and check the forward (f) permission on each intermediary Policy Director server object in a chain of junctioned servers.

NetSEAL Protected Object Permissions:	Access	Description
C	connect	Connect through a NetSEAL server to a local or remote protected service
f	forward	Permit outgoing connection across a NetSEAL junction; traverse the junction



Illustrating services controlled by NetSEAL junctions

NetSEAL junctions provide data protection and security for communications through an organization. You can partition an organization on a geographical or functional basis. For example, NetSEAL junctions are useful in situations where an untrusted network exists between two geographically separated Policy Director servers. Also, these two Policy Director servers are members of the same secure domain.

Scenarios include:

- "Incoming junctioned connection to Policy Director server".
- "Incoming junctioned connection to protected host" on page 232.
- "Outgoing connection to junctioned Policy Director server" on page 232.
- "Outgoing connection to junctioned protected host" on page 233.

Incoming junctioned connection to Policy Director server

This scenario considers an application that resides on a protected remote Policy Director server. This scenario explicitly establishes the GSS tunnel between the Policy Director servers as a NetSEAL junction. Access control to the destination port now includes consideration of the forward permission.

The Policy Director server completes the transaction in the following manner:

1. Can the user connect to the requested port on the destination server (based on its ACL)?

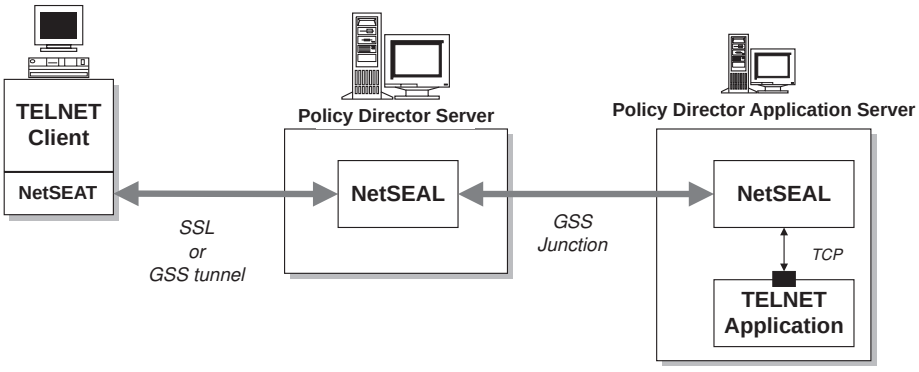
   Yes—Continue.

   No—Reject the connection request.

2. Can the user forward across the junction?

   Yes—Forward request across the GSS junction. Establish a TCP connection to the requested port.

   No—Reject the connection request.



Incoming junctioned connection to protected host

This scenario considers an application that resides on a protected remote third-party server. This scenario explicitly establishes the GSS tunnel between the Policy Director servers as a NetSEAL junction. Access control to the destination port now includes consideration of the forward permission.

The Policy Director server completes the transaction in the following manner:

1. Can the user connect to the requested port on the destination server (based on its ACL)?

   Yes—Continue.

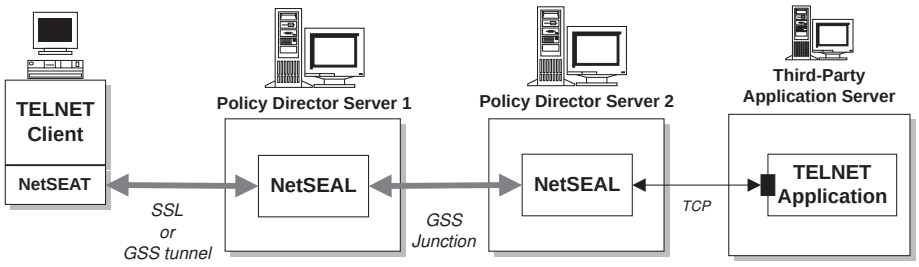
   No—Reject the connection request.

2. Can the user forward across the junction?

   Yes—Forward request across the junction. Establish a TCP connection to the requested port.

   No—Reject the connection request.

A third-party application server always has an unprotected TCP connection. Use such a configuration within a trusted network environment.



Outgoing connection to junctioned Policy Director server

This scenario considers an application that resides on a protected remote Policy Director server. This scenario explicitly establishes the GSS tunnel between the Policy Director servers as a NetSEAL junction. Access control to the destination port now includes consideration of the forward permission, which provides good mid-tier server protection.

The Policy Director server completes the transaction in the following manner:

1. Is the requested port on the destination machine protected (ACL)?

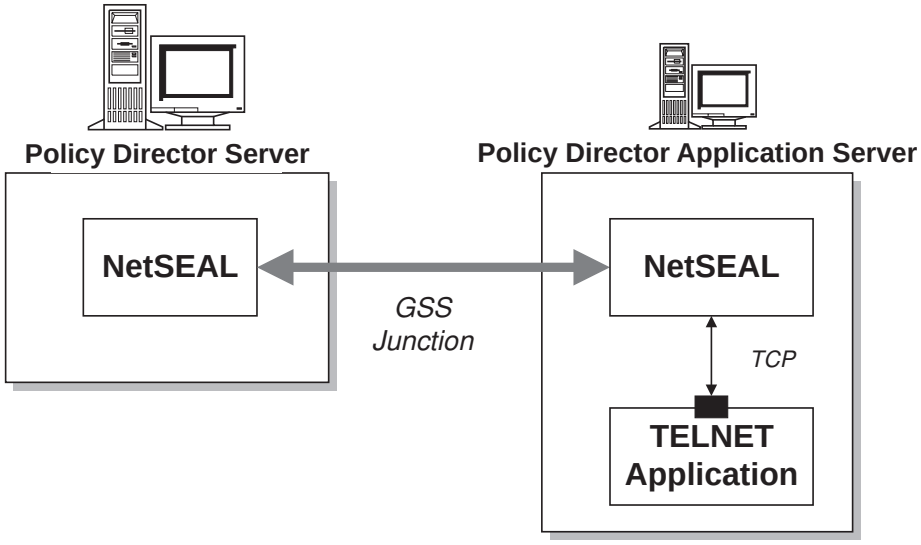
   Yes—Pass request to Security Manager (secmgrd).

   No—Allow outgoing connection.

2. Can the user forward across the junction?

   Yes—Forward request across the junction. Establish a TCP connection to the requested port.

   No—Reject the connection request.



Outgoing connection to junctioned protected host

This scenario considers an application that resides on a protected remote third-party server. This scenario explicitly establishes the GSS tunnel between the Policy Director servers as a NetSEAL junction. Access control to the destination port now includes consideration of the forward permission.

The Policy Director server completes the transaction in the following manner:

1. Is the requested port on the destination machine protected (ACL)?

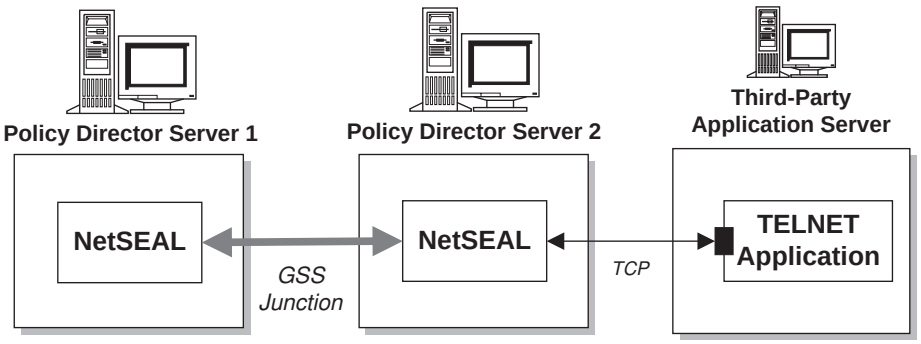
   Yes—Pass request to Security Manager (secmgrd).

   No—Allow outgoing connection.

2. Can the user forward across the junction?

   Yes—Forward request across the junction. Establish a TCP connection to the requested port.

   No—Reject the connection request.



Protecting TCP services

ACLs on the TCP-service ports control access to those ports. A NetSEAL client user can access a specific TCP service when the ACL on the port object contains the connect (C) permission for the user.

An administrator can deny access to a TCP service by simply removing the connect permission from the appropriate entry in the ACL.

ACL permissions can also control the NetSEAL communication quality of protection. Some combination of data integrity and privacy permissions determines the quality of protection. You can control the outgoing communication quality of protection through a NetSEAL junction. To control the quality of protection, entries in the ACL on the destination port object must contain the integrity permission and privacy permission.

The ACL tested is the most specific according to the network address and network mask of the object. The presence or absence of an ACL on a destination port is not taken into account when locating the requested object on which to perform the ACL check.

For example, a client Telneting to 10.0.0.1 where the following ACLs are configured is granted access according to ACL3. Policy Director grants access even though port 23 on the matching network contains an explicit ACL.

`10.0.0.0:255.255.255.0`	ACL1
`10.0.0.0:255.255.255.0/Port 23`	ACL2
`10.0.0.1:255.255.255.255`	ACL3

Chapter 18. NetSEAL: General administration tasks

Policy Director NetSEAL is a virtual private network (VPN) solution for securing incoming TCP/IP communication. As a resource manager, NetSEAL controls a user's ability to connect to a specific TCP application. This chapter contains information that describes general administration tasks you can perform to customize NetSEAL for your network.

This chapter includes:

Enabling and disabling NetSEAL security

Use the **ivadmin** utility to enable and disable NetSEAL.

Enable NetSEAL

To enable NetSEAL on a specific Policy Director server:

```
ivadmin> server enable /NetSEAL/hostname
```

Where *hostname* is the server's host name, excluding the domain name.

Policy Director returns errors when the service is already enabled or when the service specification is not valid.

Policy Director disables NetSEAL by default unless you install the NetSEAL trap (IVTrap) component of the Policy Director distribution.

Disable NetSEAL

To disable NetSEAL on a specific Policy Director server:

```
ivadmin> server disable /NetSEAL/hostname
```

NetSEAL status

To check NetSEAL server status, use the **server status** command:

```
ivadmin> server status /NetSEAL/hostname
```

The status report displays the following information:

- NetSEAL server is enabled or disabled.
- NetSEAL server is available.
- Replica NetSEAL configuration databases have, or have not, been updated.

Using NetSEAL access controls

Policy Director ACL permissions are available to NetSEAL connections for the following security concerns:

- Permitting access to TCP services, such as destination ports
- Permitting packet forwarding across NetSEAL junctions
- Ensuring data integrity and privacy

The ACL on a destination-port object controls access to that port. An ACL entry must contain the connect (C) permission to allow the user or group access to the port. The connect permission can also govern access to an application server on a protected network.

An ACL on the Policy Director server responsible for the outgoing connection controls traversal across a NetSEAL junction. An ACL entry must contain the forward (f) permission to allow the user or group access across the junction.

Use and check the forward (f) permission on each intermediary Policy Director server object in a chain of junctioned servers.

	Access	Description
C	connect	Connect through a NetSEAL server to a local or remote protected service
f	forward	Permit outgoing connection across a NetSEAL junction; traverse the junction

ACL permissions can also control the NetSEAL communication quality of protection. Some combination of the data integrity permission and the privacy permission determine quality of protection.

You can control the outgoing communication quality of protection through a NetSEAL junction. The entries in the ACL on the destination-port object must contain the integrity (I) and privacy (P) permissions. You cannot extend data integrity and privacy to a third-party (non-Policy Director) application server on a trusted network.

Managing protected networks

You can think of networks as non-Policy Director servers that are protected by NetSEAL. Use the **ivadmin** utility to define and manage networks. Commands include add, delete, and list protected networks.

Command	Description
netseal network add *network netmask* **[***network-alias***]**	
	Create a new network to be protected by NetSEAL. The network/netmask pair are standard IP network address numbers and netmasks. The optional network alias name can be used to identity this network. If no alias is specified, the network must be identified by the network and netmask pair. Errors are returned if the network already exists.
netseal network delete *network-id*	
	Delete the specified network from the system where *network-id* can correspond to one of the following: • *network/netmask* pair • *network-alias* The network-id argument can correspond to either a network/netmask pair, or a network alias. All references to this network within the system will be removed, including NetSEAL junctions. An error will be returned if the network is not found within the database.
netseal network list	
	Display all of the networks in the database, which includes the network and netmask pair as well as any defined aliases.

Example:

```
ivadmin> netseal network add 10.125.0.0 255.255.255.0 west
```

This command adds the network nodes 10.125.0.0 through 10.125.0.255 to a network specification that NetSEAL protects. Also, this command assigns to this network specification the alias name `west`.

Managing NetSEAL junctions

NetSEAL junctions determine the direction communication should travel through a Policy Director server. You can create junctions between two Policy Director servers or between a Policy Director server and a network. Use of a GSS tunnel secures communication across a junction between two Policy Director servers.

Use the **ivadmin** utility to define and manage NetSEAL junctions. Commands include add, delete, and list NetSEAL junctions.

Command	Description
netseal junction add *hostname destination*	
	Creates a junction from a NetSEAL server to a specified destination where *hostname* is the name of the NetSEAL server minus the domain name, and *destination* can correspond to one of the following: • *pd-server* • *network/netmask* pair • *network-alias* Errors are returned if the junction already exists, the host server does not exist, or the destination does not exist.
netseal junction delete *hostnamedestination*	
	Deletes the junction from a NetSEAL server to the specified destination where *hostname* is the name of the NetSEAL server minus the domain name, and *destination* can correspond to one of the following: • *pd-server* • *network/netmask* pair • *network-alias* An error is returned if the junction does not currently exist. The command has no effect on current connections.
netseal junction list *hostname*	
	Display all of the NetSEAL junctions for the specified NetSEAL server.

Example:

```
ivadmin> netseal junction add clipper west
```

This command creates a junction from the NetSEAL server `clipper` to the network that is defined by the alias `west`. The command also defines the routing direction (`clipper` to `west`); NetSEAL junctions are unidirectional.

Managing protected ports

The Policy Director NetSEAL server provides security services for specific ports, hosts, and networks. For example, you can configure the NetSEAL server to secure Telnet traffic on a specific port.

Use the **ivadmin netseal** utility to define the list of ports that you want NetSEAL-protected. Commands include add, delete, and list protected ports. You can specify ports for Policy Director servers or networks.

Command	Description
netseal port add *destination port-id*	
	Protects connections to the specified destination on the specified *port-id* where *destination* can correspond to one of the following: • *pd-server* • *network netmask* • *network-alias* And *port-id* can correspond to one of the following: • *port* • *port-range* • *port-alias* An error is returned if the port is already being protected, if the server does not exit, or the port alias does not exist.
netseal port delete *destination port-id*	
	Stops protecting connections to the specified destination on the specified port where *destination* can correspond to one of the following: • *pd-server* • *network netmask* • *network-alias* And *port-id* can correspond to one of the following: • *port* • *port-range* • *port-alias* An error is returned if the port is not already being protected, if the server does not exist, or the port alias does not exist.
netseal port list *destination*	
	Displays a list of all ports for the specified destination where *destination* can correspond to one of the following: • *pd-server* • *network netmask* • *network-alias* An error is returned if the server does not exist.

Note: Express the port range as two port numbers separated by a dash (for example: 22–88).

Example:

```
ivadmin> port add west 23
```

This command defines the port number 23 on the network that is specified by the network alias “west” as a NetSEAL-protected port.

Managing protected port aliases

Use the **ivadmin** utility to define and manage port aliases. Commands include add, delete, and list port aliases. Use port aliases to identify trapped port ranges in a more meaningful manner.

Command	Description
netseal port-alias add *port-spec port-alias*	
	Create a new port alias for the specified port specification where *port-spec* can correspond to one of the following: • *port* • *port-range* An error will be returned if the port range is already aliased by another name.
netseal port-alias delete *port-id*	
	Delete the port alias for the specified port-id from the system Where *port-id* can correspond to one of the following: • *port* • *port-range* • *port-alias* An error will be returned if the port alias is not found within the database.
netseal port-alias list	
	Display all of the port aliases found within the database.

Examples:

```
ivadmin> netseal port-alias add 23 telnet
```

This command creates the port alias `telnet` for the port number `23`.

```
ivadmin> netseal port-alias add 5000-5010 pilot
```

This command creates the port alias “pilot” for the port range 5000 through 5010.

Configuring trusted hosts and trusted networks

The Policy Director NetSEAL server provides security services for specific ports, hosts, and networks. For example, you can configure the NetSEAL server to secure Telnet traffic on a specific port on the Policy Director server. In addition, the NetSEAL server can trust certain systems (trusted hosts) and certain collections of hosts (trusted networks).

The `[trusted_hosts]` or `[trusted_networks]` stanzas of the secmgrd.conf configuration file contain parameters for identifying trusted hosts and trusted networks.

Trusted hosts

Your NetSEAL server communicates often with specific highly trusted servers (hosts). You can optimize performance by allowing NetSEAL to exempt incoming requests from these servers from access control.

Note: Identifying trusted hosts makes your system vulnerable to IP-spoofing attacks. Make sure that you protect any trusted hosts from such attacks.

By default, Policy Director does not identify any trusted hosts.

To identify a trusted host, go to the `[trusted_hosts]` stanza, and list the IP address and server name.

For example, to trust all requests from a server that is named “typhoon” with an IP address of 220.12.35.102, type:

```
[trusted_hosts]
220.12.35.102 = typhoon
```

When using the NetSEAL trap to secure a machine, it is often necessary to trust the local machine. Trusting the local machine allows services that run on this machine to continue functioning. These services continue even when unauthenticated access is turned off to a port or range of ports.

Policy Director requires these entries in order to *trust* the local machine:

- One entry for the local host
- Entries for each IP address associated with that machine

For example, the NetSEAL server typhoon might have an IP address of 220.12.35.102. Another process on the same server might have the local host IP address of 127.0.0.1. To trust all local requests on NetSEAL server typhoon from other processes on the same server, you would type:

```
[trusted_hosts]
220.12.35.102 = typhoon
127.0.0.1 = localhost
```

There will typically be only one IP address per machine, however some machines are connected to more than one network and thus have more than one address.

Trusted networks

If your network includes entire subnets or local area networks of trusted systems, you can specify the entire subnet without having to list each host.

By default, Policy Director does not define any trusted networks.

To identify a trusted network, go to the `[trusted_networks]` stanza and list the subnet IP address and netmask.

For example, to trust all requests from subnet 192.96.32.0, type:

```
[trusted_networks]
192.96.32.0 = 255.255.255.0
```

Setting SSL time-out parameters

The SSL time-out parameters you can set include:

- “Setting SSL session cache timeout”.
- “Setting SSL connection timeout”.

Setting SSL session cache timeout

The [ssl] stanza of the secmgrd.conf configuration file contains the parameter for setting the static SSL session cache timeout.

NetSEAL caches credential information internally. The session cache time-out parameter dictates the length of time authorization-credential information remains in memory on NetSEAL.

The parameter is not an inactivity timeout. The value maps into a *credential lifetime* rather than a *credential timeout.* Its purpose is to enhance security by forcing the user to re-authenticate when the specified time-out limit is reached.

The default cache timeout (in seconds) is:

```
[ssl]
ssl-cache-timeout = 3600
```

Adjust this value to balance server performance against user convenience, depending on the volume of SSL requests that the server must handle.

Setting SSL connection timeout

The [ssl] stanza of the secmgrd.conf configuration file contains the parameter for setting the SSL connection timeout.

When NetSEAL accepts an SSL connection from a NetSEAT client over a NetSEAL SSL tunnel, an SSL protocol handshake needs to take place. This parameter controls how long the Security Manager will wait for NetSEAT to initiate an SSL handshake at the start of an SSL connection. After this time, the Security Manager shuts down the connection.

The default SSL connection timeout (in seconds) is:

```
[ssl]
ssl-init-connect-timeout = 120
```

Allocating NetSEAL connections

Parameters for allocating NetSEAL connections are located in the [netseal] stanza of the secmgrd.conf configuration file.

Use max-connections to specify the maximum number of simultaneous connections that NetSEAL will allow.

Policy Director installation sets the following default value:

```
[netseal]
max-connections = 32
```

You might want to increase this value to best suit the traffic conditions within your network. If you set the max-connections too low, connections will be rejected under heavy load conditions. Setting it too high results in wasted resources and a degradation in server performance.

Note: The lower bound for this configuration parameter is 20. If you attempt to designate a setting lower than 20, the value simply defaults to 20.

Chapter 19. NetSEAT: Overview

The Policy Director NetSEAT client enables Windows clients to participate in a Policy Director secure domain. NetSEAT provides secure tunneling between Windows clients and Policy Director servers. NetSEAT encrypts communications by using either SSL or GSS tunneling.

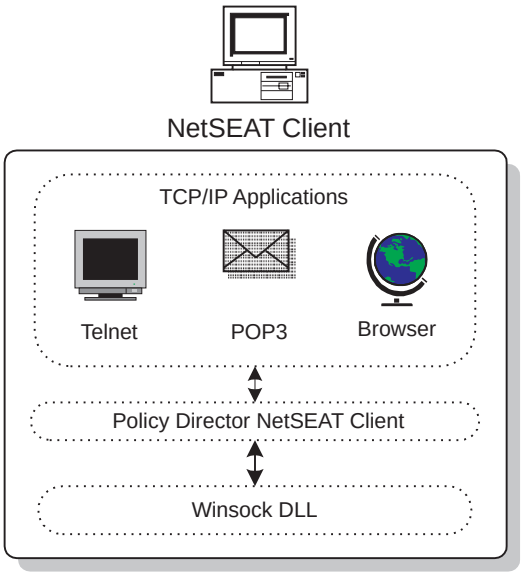
This chapter includes:

NetSEAT client overview

The Policy Director NetSEAT client enables Windows clients to communicate securely with Policy Director WebSEAL and NetSEAL servers. Adding NetSEAT to a Windows workstation configures the workstation into a Policy Director secure domain. A secure domain is where NetSEAT can use the Policy Director authentication and authorization security services.

NetSEAT provides authentication and message protection at the network-protocol level. An application does not have to recompile or relink in order to use NetSEAT.

NetSEAT secures network traffic between the Windows client and a Policy Director server by intercepting network requests before they go through the Winsock layer. NetSEAT uses its configuration information to recognize requests that are made from generic TCP/IP applications. Generic TCP/IP application include Telnet, POP3, or HTTP.



When a client request is made to an application on a Policy Director server, NetSEAT transparently establishes a secure tunnel to the Policy Director server. Then, NetSEAT redirects the client requests through the tunnel.

NetSEAT secures and encrypts communications to a Policy Director server by using one of two methods: an SSL tunnel or a GSS tunnel. An SSL tunnel uses SSL to encrypt communication. A GSS tunnel uses the GSS API to encrypt communication.

You can deploy the NetSEAT client for several different purposes, as described in the sections describing the:

- virtual private network (VPN) client
- "Support module for Policy Director on Windows NT" on this page.
- "Support module for Policy Director Management Console".

Virtual Private Network client

You can configure NetSEAT as a virtual private network (VPN) client that uses a secure tunnel to provide a secure communications link with a Policy Director NetSEAL server.

As a VPN client, NetSEAT can encrypt communication using the following types of tunnels:

- SSL
- GSS

Support module for Policy Director on Windows NT

Policy Director installs NetSEAT as a support module for each installation of the Policy Director for Windows NT server components. Policy Director for Solaris and AIX do not require this support from NetSEAT client.

In this role, the NetSEAT client provides a kernel trap for internal use by the Policy Director security services.

As a Policy Director for Windows NT support module, NetSEAT requires the following services:

- Directory Services Broker
- GSS tunneling

Use GSS tunneling when installing the NetSEAT client as a support module either to Policy Director for Windows NT or Policy Director Management Console on Windows.

Support module for Policy Director Management Console

When running the Policy Director Management Console on a Windows client, you must install NetSEAT as a support module. In this configuration, NetSEAT enables an administrator to use the Management Console to perform administration tasks from a Windows system. The Windows system does not have to have any of the Policy Director server components installed.

As a support module for the Management Console, NetSEAT requires the following services:

- Directory Services Broker
- GSS tunneling

Use GSS tunneling when installing the NetSEAT client as a support module either to Policy Director for Windows NT or Policy Director Management Console on Windows.

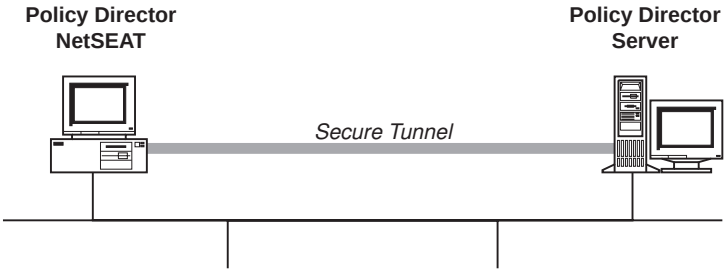
Secure tunneling

Before forwarding a client request, NetSEAT uses a secure tunnel to contact the Policy Director Security server (for the appropriate secure domain). Also, NetSEAT uses a secure tunnel to establish the client's identity and credentials. On successful authentication of the client, NetSEAT encapsulates the requested transaction in another secure tunnel. Also, NetSEAT completes the requested transaction according to the applicable security settings.

For example, when a Web browser requests access to a service or resource that is secured by WebSEAL, NetSEAT transparently intercepts the request. If this is the first request to Policy Director and you require authentication, NetSEAT prompts the user with a login dialog box.

After Policy Director authenticates a user, NetSEAT transparently allows subsequent requests and secure tunnels that are based on these initial credentials. NetSEAT uses these credentials to make authorization decisions on each request.

NetSEAT can establish two different types of secure tunnels, as described in "Using SSL tunneling" and "Using GSS tunneling" on page 248.



Using SSL tunneling

The term *SSL tunneling* refers to a secure tunnel that uses the SSL protocol. NetSEAT uses SSL tunneling when deployed as a VPN client to Policy Director NetSEAL. Use of the SSL tunnel simplifies the NetSEAT configuration. In this scenario, the administrator does not need to specify the configuration of the DCE services within the Policy Director secure domain.

An SSL tunnel is also useful when a client wants to access data that is protected by a firewall. In this mode, NetSEAT uses a single port for all communications with the Policy Director server behind the firewall. Policy Director encapsulates all communications through the port in a secure tunnel.

During NetSEAT configuration, you can specify the port number you want to use when traversing the firewall. The port number must match the port that is configured on the NetSEAL server.

SSL tunnelling can be used by users who are authenticated through eityher an LDAP user registry or a DCE user registry.

Using GSS tunneling

The term *GSS tunnelling* refers to a secure tunnel based on the use of the Generic Security Service API (GSS API). NetSEAT uses a GSS tunnel when communicating with DCE cells.

In this mode, NetSEAT is a client that uses Security Managers (secd) and Time Servers (dtsd) located on other servers within the DCE cell. NetSEAT also uses an Policy Director Directory Services Broker to proxy namespace lookup requests to Cell Directory Services (cdsd) located elsewhere in the DCE cell.

A secure GSS tunnel is used when integrating a userÆs PKI login, through an Entrust client, with the userÆs identity in the Policy Director secure domain. In this scenario, users log in to their workstation using an Entrust PKI client. When the user attempts to use Policy Director authentication and authorization services, a secure tunnel is established between NetSEAT and the Policy Director servers. The Policy Director servers automatically map the PKI login identity to the userÆs Policy Director identity in order to make authentication and authorization decisions.

GSS tunnelling cannot be used for users who are authenticated by an LDAP user registry.

Accessing protected servers

A NetSEAT client can use its secure tunnel to a Policy Director NetSEAL server to send requests to any remote application servers. These remote application servers are protected by the NetSEAL server. During NetSEAT configuration, the administrator can specify:

- The name of the application server.
- The NetSEAL server that protects it.
- The type of secure tunnel (SSL or GSS tunnel) that is used between NetSEAT and NetSEAL.

When configured with this information, NetSEAT intercepts the Windows client's requests to the application server. Then, NetSEAT routes them through a secure tunnel to a Policy Director NetSEAL server.

You can also configure NetSEAT to access entire protected subnets when Policy Director NetSEAL protects a subnet. An example would be an intranet that is protected from the Internet by a Policy Director NetSEAL server.

Directory Services Broker

When using as the support module for Policy Director servers and Management Console on Windows (GSS tunneling), NetSEAT requires the services of a Directory Services Broker (DSB). A DSB is needed to communicate with the Policy Director servers. The DSB proxies namespace lookups to a Cell Directory Services server that is located in the same secure domain.

Policy Director automatically installs the DSB as part of the Policy Director Management Server (IVMgr) package. You must configure the NetSEAT client to know the name of the server system that hosts the Directory Services Broker.

Chapter 20. NetSEAT: General administration tasks

This chapter describes system administration tasks for configuring the NetSEAT client, managing security contexts, and finding and correcting problems (*troubleshooting*).

This chapter includes:

Configuring NetSEAT client

Configure each NetSEAT client into a Policy Director secure domain. You can configure NetSEAT clients at installation time or at any time after installation. The NetSEAT configuration tool (utility) provides a graphical user interface that enables administrators to easily configure the client.

Perform all configuration tasks within the context of a secure domain in which you configured NetSEAT to participate.

Some configuration tasks apply only to configurations for one of the secure tunnel types (GSS or SSL). NetSEAT uses these secure tunnels to communicate with Policy Director servers. NetSEAT clients, which you have configured solely to be SSL clients to NetSEAL servers, require only a limited subset of the configuration tasks.

The following table shows the configuration tasks that correspond to each secure tunnel type:

Tunnel types	Configuration tasks
Both GSS and SSL	• "Adding NetSEAT into a secure domain" on page 250 • "Configuring NetSEAL servers" on page 252 • "Denying access to network resources" on page 259
GSS only	• "Adding DCE servers" on page 251 • "Setting DCE server properties" on page 252 • "Configuring integrated login" on page 255 • "Configuring advanced login (PKI integration)" on page 257. • "Setting the maximum time delta" on page 259
SSL only	"Configuring an SSL proxy" on page 259

NetSEAT provides extensions to several DCE security utilities and also provides a utility for checking on the availability of DCE services.

You use these utilities only when you configure NetSEAT to use GSS tunneling.

The following sections describe these utilities:

- Using NetSEAT security utilities (refer to “Using the NetSEAT security utilities” on page 260).
- Finding and correcting problems by using **netseat_ping** (refer to “Troubleshooting by using netseat_ping” on page 262.)

Starting the NetSEAT configuration tool

To reconfigure NetSEAT after you complete initial installation and configuration, use the NetSEAT configuration tool. Or, to configure NetSEAT when you postponed configuration during initial installation, use the NetSEAT configuration tool.

There are two ways to start the NetSEAT configuration tool:

- To start the NetSEAT configuration tool from the Windows desktop, click **Start → Programs → Policy Director → NetSEAT → NetSEAT Configuration**
- To start the NetSEAT configuration tool from the **NetSEAT** icon in the System Tray, right-click the **NetSEAT** icon and select **Properties**.

Adding NetSEAT into a secure domain

To add NetSEAT into a secure domain, complete the following steps:

1. Start the NetSEAT configuration tool. On startup, the NetSEAT configuration tool displays the **Secure Domains** tab in the NetSEAT configuration window.
2. Click **Add**.

   The New Secure Domain dialog box appears.
3. Type the name of the secure domain to which NetSEAT belongs.
4. Select the supported protocols for this domain, and click **OK**.
   - If you selected **Enable GSS**, go to “Adding DCE servers” on page 251.
   - If you selected **Enable SSL only**, go to “Configuring NetSEAL servers” on page 252.
5. **GSS tunneling only**—If you have configured more than one domain, return to the **Secure Domains** tab. Highlight the default domain for user logins. Then, click **Set as Default**.

   If you configured only one domain, the domain automatically becomes the default.
6. Click **OK**.

Adding DCE servers

If you configure a domain to enable GSS secure tunnel use, there are other configuration options that you must set.

For the secure domain (or *cell*) that NetSEAT joins, obtain the name of Policy Director security server systems in the cell. Identify the basic security (DCE) services each system provides. For each server, determine if it provides:

- **Security** — Security Services (secd)
- **Time** — Time Services (dtsd)
- **DSB** — Directory Services Brokers (dsb)
- **CDS** — Cell Directory Services (cdsd)

  NetSEAT needs to know the location of the CDS only if the Directory Services Broker (DSB) is running on the NetSEAT client. You usually configure the DSB to run on the same server as the Policy Director Management Server (IVMgr).

After creating an entry for a New Secure Domain and after clicking **OK**, the Secure Domain Properties dialog box appears.

1. Click **Add**.

   The Add a DCE Server dialog box appears.
2. Type the name of a server that provides DCE services in this secure domain.
3. Select one or more of the following services for each of the servers: **Security**, **Time**, **DSB**, or **CDS**.
4. Optionally, you can click **Advanced** if you want to set any of the advanced properties for this DCE server.

   Refer to “Setting DCE server properties” on page 252.
5. Click **OK** to complete the setting of supported services on the specified DCE server.

   The Secure Domain Properties dialog box returns.
6. Accept the following defaults:
   - Integrated Login Support: **Disabled**
   - Advanced Login: **DCE Login only**

   If you want to optionally configure integrated login support and advanced login for the secure domain, see the following sections:
   - Configuring integrated login (refer to “Configuring integrated login” on page 255).
   - Configuring advanced login (refer to “Configuring advanced login (PKI integration)” on page 257).
7. When you have finished configuring this DCE server, click **OK**.

   The Secure Domain window reappears. You have now completed the required configuration tasks for adding a DCE server.

Setting DCE server properties

You can set the following values at the Advanced DCE Server Properties dialog box during configuration of a DCE server. This configuration task is optional.

Protocols and ports

For each DCE server, you can optionally specify the protocol (TCP or UDP) that is used by each security services. You can use this feature to configure operation through firewalls, such as the IBM SecureWay Firewall Version 4.1, a component of the IBM SecureWay Boundary Server product.

For example, you might want to deselect UDP access for the DCE services, and specify the TCP port numbers that the firewall administrator has configured.

Priority levels

When defining domains that have multiple copies of one or more services available, you can specify the order in which NetSEAT accesses each service. You can assign each service a positive integer to set its priority. A larger number means a higher priority.

Administrators can use this feature to optimize performance by having NetSEAT first go to the service that is electronically closest. If that service is unavailable, NetSEAT will default to the copy of the service with the next highest priority.

1. To configure the advanced properties, go to the Add a DCE Server dialog box.
2. Select the **DCE server**.
3. Click **Advanced**.

   The Advanced DCE Server Properties dialog box appears.
4. To restrict the protocol that is used to contact a DCE service, deselect the check box next to the appropriate DCE service. Deselecting the check box removes the protocol that you do not want enabled.
5. Type port numbers, if necessary, in the field next to the appropriate DCE service.
6. Set the priority level for each DCE service.
7. Click **OK**.

Configuring NetSEAL servers

To configure NetSEAT to communicate with a NetSEAL server, complete the following steps:

1. Go to the **NetSEAL Servers** tab and select the secure domain from the drop-down list.
2. Click **Add**.

   The Add a NetSEAL Server dialog box appears.
3. Type the machine name for the NetSEAL server.
4. If NetSEAL uses nondefault ports for GSS or SSL tunneling, type those port numbers in the appropriate fields. Otherwise, accept the default values.
   - The protocols that were not enabled when the secure domain entry was created are grayed out.
   - If GSS tunneling is enabled, do not select the **Specify Principal Name** check box. Only use this feature for backward compatibility with previous versions.

5. If you are using SSL tunneling and your NetSEAT configuration has an SSL proxy server configured, Policy Director will select automatically the **Use Proxy Server** check box.

   When you have not enabled an SSL proxy server, the **Use Proxy Server** check box is inactive (deselected and grayed out). To enable an SSL proxy server in the NetSEAT configuration, refer to “Configuring an SSL proxy” on page 259.
6. Click **OK**.

   The **NetSEAL Server** tab reappears. You have now added the NetSEAL server to the NetSEAT configuration.

Adding a protected server

You can configure NetSEAT to specify a communication channel with application servers that are protected by a NetSEAL server.

This option is available when the NetSEAT client uses either GSS tunneling or SSL tunneling.

After you add an application server to NetSEAT’s configuration, NetSEAT intercepts the Windows client’s requests to the application server. NetSEAT then routes them through a secure tunnel to a NetSEAL server.

To add a protected server to the NetSEAT configuration, provide the following information:

Field	Definition
Machine name	The name by which the application server is known in the TCP/IP domain.
Tunnel destination	The name of the NetSEAL server that is protecting the application server.
Port range	The port number on the protected server that is secured by the NetSEAL server. This port, or range of ports, is specified on the NetSEAL server through the ivadmin command.
Selected Protocol	The tunneling protocol. Both GSS and SSL can be enabled for the secure domain. Specify SSL tunneling for Policy Director. GSS tunneling is used for NetSEAL-to-NetSEAL connections.

To configure NetSEAT to recognize outgoing requests to a protected server, complete the following steps:

1. Click the **Host Security** tab.
2. Click **Add**.

   The Add a Protected Server dialog box appears.
3. For Machine Name, type the machine name of a server that is protected by a NetSEAL Server.

   Only one NetSEAL server can protect an application server.
4. For Tunnel Destination, use the drop-down list to select the NetSEAL server that protects the server.
5. Use the drop-down list, if necessary, to select the appropriate tunneling protocol.
6. Click **Add**.

   The Add a Port Range dialog box appears.
7. Specify the port or range of ports on the NetSEAL server that NetSEAL uses to communicate with the protected application server.

8. Click **OK**.

   The Add a Protected Server dialog box returns.

   For example, the following dialog box entries set the following values:

   - A NetSEAL server that is named “sunshine” protects a protected server that is named “thunder.”
   - The server sunshine protects communication to thunder on ports 5000-5005.
   - Policy Director defines a secure tunnel between the NetSEAT client and the NetSEAL server “sunshine.”
   - The secure tunnel is of type SSL tunneling.

9. Click **OK**.

   The **Host Security** tab reappears.

   Policy Director adds the protected server to the NetSEAT configuration.

Adding a protected subnet

You can configure NetSEAT to specify a communication channel with a subnet that is protected by an NetSEAL server. This option is available when the NetSEAT client uses either GSS tunneling or SSL tunneling.

After you add an application subnet to NetSEAT’s configuration, NetSEAT intercepts the Windows client’s requests to the application subnet. NetSEAT then routes them through a secure tunnel to an NetSEAL server.

To add a protected subnet to the NetSEAT configuration, provide the following information:

Field	Definition
Name of Any Machine in the Subnet	The name of any server on the protected subnet.
Netmask	The netmask of the subnet (for example, 255.255.0.0).
Tunnel Destination	The name of the NetSEAL server that is protecting the subnet.
Select Protocol	The tunneling protocol. Both GSS and SSL can be enabled for the secure domain. Specify SSL tunneling for Policy Director. GSS tunneling is used for NetSEAL-to-NetSEAL connections.

To configure NetSEAT to recognize outgoing requests to a subnet that is protected by a NetSEAL server, complete the following steps:

1. Click the **Subnet Security** tab.
2. Select the secure domain that contains the NetSEAL server (that protects the subnet).
3. Click **Add**.

   The Add A Protected Subnet dialog box appears.

4. Type the name of any machine on the subnet that is protected by the NetSEAL server.

   Only one NetSEAL server can protect an application subnet.

5. Type the netmask for the subnet.
6. For Tunnel Destination, use the drop-down list to select the NetSEAL server that protects the subnet.
7. Use the drop-down list, if necessary, to select the appropriate tunneling protocol.

For example, the following entries enable NetSEAT to access a subnet with netmask 255.255.0.0. The system named “thunder” is on the subnet. The NetSEAL server “sunshine,” which is also the SSL tunnel destination, protects the subnet:

Name of Any Machine in the Subnet: `thunder`

Netmask: `255.255.0.0`

Tunnel Destination `sunshine`

Select Protocol `SSL`

8. Click **OK**.

   The **Subnet Security** tab reappears.

   Policy Director has configured the NetSEAT client to communicate with the protected subnet.

Configuring integrated login

Optionally, you can install Integrated login support during the NetSEAT installation. The NetSEAT installation changes the Windows NT registry to support integrated login.

After you install integrated login, the NetSEAT configuration tool can enable or disable integrated login for each secure domain.

You can configure integrated login during the initial installation and configuration of NetSEAT. Or, you can configure it later. You set the Integrated login configuration independently for each secure domain.

Before configuring integrated login, a NetSEAT user should complete the following tasks:

- Obtain an account in each secure domain where an automatic login is needed.
- Configure the NetSEAT client to be a member of each secure domain.
- Synchronize the username and password for the secure domain login with those that are used in the Windows NT domain.

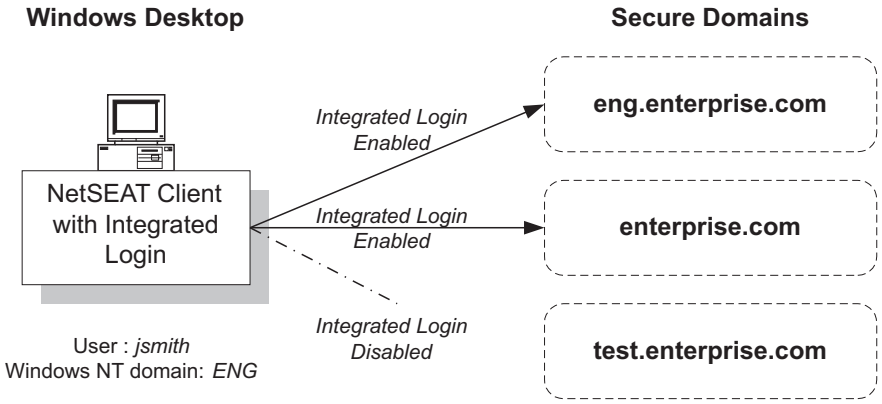
Reviewing an example integrated login configuration

An engineer named John Smith, with the Windows user name of jsmith, typically logs in to the Windows NT domain ENG. He logs in to several secure domains, as follows:

Secure Domain Name	Secure Domain User Account	Secure Domain Description
eng.enterprise.com	jsmith	The engineering division secure domain
enterprise.com	ENG/jsmith	The company-wide secure domain
test.enterprise.com	test_user	A small secure domain used only for testing purposes. The test cell does not maintain a full user name registry. Instead, users log in as test_user.

John Smith uses the NetSEAT configuration tool to configure integrated login for each secure domain as follows:

Secure Domain	Integrated Login Configuration
eng.enterprise.com	Integrated login enabled and configured to map Microsoft Windows NT user jsmith to secure domain user jsmith, and to automatically log jsmith into eng.enterprise.com.
enterprise.com	Integrated login enabled and configured to map Microsoft Windows NT user jsmith to secure domain user ENG/jsmith, and to automatically log ENG/jsmith into enterprise.com.
test.enterprise.com	Integrated login disabled, because jsmith must manually log in to this cell as user test_user.



Configuring integrated login

To configure integrated login:

1. Start the NetSEAT configuration tool.

   The **Secure Domains** tab appears.
2. Select the secure domain for which you want to configure integrated login.
3. Click **Edit**.

   The Secure Domain Properties window appears.
4. Select one of the **Integrated Login Support** menu choices.

   If **Integrated Login Support** is grayed out, you cannot enable integrated login for this secure domain because you did not install integrated login support.

   - Select **Disabled** to disable integrated login support for this secure domain.
   - Select **Enabled—DCE user name is Windows user name** if the user name in this secure domain matches the Windows user name.
   - Select **Enabled—DCE user name is Windows domain name/Windows user name** if the user name in this secure domain includes the Windows domain name.
5. Click **OK**.

   The **Secure Domains** tab appears.
6. Click **OK**.

Configuring integrated login notification mode

A secure domain password might not match the Windows NT domain password. When it does not match, the Windows registry entry determines whether secure domain logins fail silently (*silent mode*). Or, it prompts the user for the current password for the secure domain *(interactive mode).* In silent mode, Policy Director logs in the user into the Windows NT domain but not into the secure domain. In interactive mode, you can synchronize the secure domain password with the Windows password.

To change the notification mode, use the Registry Editor to edit this Windows registry entry:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetSEAT\Parameters
```

Change the following value:

```
InfoLevel:
0x00000001 (1)
```

InfoLevel can be one of the following values:

Registry Value	Mode	Description
0	Silent	After a login attempt, neither successful nor unsuccessful logins are reported to the user.
1	Interactive	After a login attempt, unsuccessful logins are reported to the user, and the user is prompted for actions.
2	Verbose	After a login attempt, both successful and unsuccessful logins are reported to the user.

Configuring advanced login (PKI integration)

This configuration task is optional, and applies only to NetSEAT clients that have enabled GSS tunneling.

You can configure NetSEAT to integrate a user's Public Key Infrastructure (PKI) login with the user's NetSEAT (Kerberos) login.

By default, integration with PKI login is disabled. Run the NetSEAT configuration tool to enable PKI Login. You enable or disable PKI login separately for each secure domain in which the NetSEAT client participates.

Supported PKI releases

NetSEAT supports integration of a user login with Entrust Version 4.0 only.

Note: The Entrust client must be installed prior to configuring NetSEAT PKI login.

Using the NetSEAT login utility

The **NetSEAT login** utility, when called from the Windows **Start** menu, displays a **PKI Login** check box at login time. This check box displays the advanced login setting that is configured for the current secure domain.

Using the PKI login check box

During NetSEAT login, a user can use the **PKI Login** check box to override the PKI Login configuration on a per log-in basis. Overriding is useful when the setting **PKI Login with fallback to DCE Login** was selected during configuration. Users can override when they want to skip the PKI login attempt and perform a DCE login. In this case, the user deselects the **PKI Login** check box to force a DCE Login.

If a user configured **PKI Login only** during configuration, deselecting the **PKI Login** check box during login has no effect.

If a user configured **DCE Login only** during configuration, selecting the **PKI Login** check box during login produces an error message.

Using the system tray login

You can use the **NetSEAT** icon in the System Tray to perform a NetSEAT Login. Use the drop-down list to select the domain to which you want to log in. If PKI login has been configured, the Entrust login prompt will appear.

If the PKI login fails and you have configured **PKI Login with fallback to DCE Login**, Policy Director prompts you to complete a DCE login.

Configuring advanced login

To integrate PKI Login with NetSEAT login:

1. Start the NetSEAT configuration tool.

   The **Secure Domains** tab appears.
2. Select the secure domain for which you want to configure PKI login.
3. Click **Edit**.

   The New Secure Domain dialog box appears.
4. Click **Configure**.

   The Secure Domain Properties window appears.
5. In the Advanced Login area, select one of the advanced login drop-down list choices.
   - Select **DCE Login only** to enable the user to log in to the Policy Director secure domain by using a username and password (Kerberos login).
   - Select **PKI Login with fallback to DCE Login** to enable a user to attempt a PKI Login. If the login is unsuccessful, NetSEAT will prompt the user to enter a username and password for a DCE login.
   - Select **PKI Login only** to require the user to log in using an X.509 certificate.
6. Click **OK**.

   The New Secure Domain dialog box appears.
7. Click **OK**.

   The **Secure Domains** tab reappears.

Setting the maximum time delta

This configuration task is optional, and applies only to NetSEAT clients on the Policy Director Management Console and the Windows NT servers for Policy Director.

NetSEAT uses remote time services when configured to use GSS tunneling. You can configure the maximum time delta allowed between the secure domain time and the NetSEAT system's clock. Or, use the default value of 15 minutes.

To set the maximum time delta:

1. Click the **General** tab.
2. Type a setting in the **Maximum Time Delta** field, if necessary.
3. Click **OK** or **Apply**.

Denying access to network resources

This configuration task is optional, and applies to NetSEAT clients that have enabled either GSS or SSL tunneling.

You can prevent a user from making nonsecure Telnet, RLOGIN, or HTTP requests from a workstation. Configure the NetSEAT client to deny or limit access to these network services. Denying or limiting access forces the client workstation to use only secure, encrypted Policy Director communication channels when engaging in network conversations.

This feature is off by default. This feature should only be turned on for highly secure, special-purpose networks.

To deny access to unprotected network services:

1. Select the **General** tab.
2. If desired, select the **Deny access to unprotected network services** check box.
3. Click **OK** or **Apply**.

Configuring an SSL proxy

This configuration task is optional, and applies only to NetSEAT clients that have enabled SSL tunneling.

When requests from the NetSEAT client must go through your network's SSL proxy server, you can configure NetSEAT client to use the proxy server. On the **General** tab, the **Enable Proxy Server** check box enables the proxy server for use by all NetSEAL servers.

If you deselect this check box, none of the NetSEAL servers that were added into the NetSEAT configuration will be able to access the proxy server.

To specify an SSL proxy server, complete the following steps:

1. Click the **General** tab
2. If necessary, change the number of minutes in the **Maximum Time Delta** field. The default is 15 minutes.

3. Make sure you do not select the **Deny access to unprotected network services** check box.
4. Select the **Enable Proxy Server** check box.
5. Type the name in the **Proxy Server Machine Name** field.
6. Type the port number in the **Proxy Server Port** field.
7. Click **OK** or **Apply**.

Using the NetSEAT security utilities

NetSEAT client provides the DCE security service utilities that are found in traditional DCE implementations. NetSEAT's unique architecture allows extension of the standard functionality for each of these utilities: **klist**, **kdestroy**, and **dce_login**.

klist

The **klist** command lists the primary user (*principal*) and the tickets that are held in the default credentials cache. Or, if you use the **-c** option, this command will list the user and tickets that are held in the cache that is identified by cache name.

NetSEAT carries out the standard **klist** command options and adds extended options.

The standard options are:

Option	Description
-c *cachename*	Specifies that the contents of the cache identified by cache name should be displayed instead of the contents of the default cache.
-e	Includes expired tickets in the display. Without this option, only current tickets are displayed.
-f	Displays option settings on the tickets.

The extended options are:

Option	Description
-C *cellname*	Describes the primary user and tickets held for the user in the DCE cell identified by *cellname*.
-m	Describes the primary user and tickets held for the user in all DCE cells for which the user has a DCE login context.
-s	Displays a short summary of all the cells the user is logged into and the user's login name for each of those cells.

kdestroy

The **kdestroy** command destroys a user's login context and the user's credentials. Until Policy Director re-establishes the credentials, the user and any processes that are created by the user are limited to unauthenticated access.

NetSEAT client supports the standard **kdestroy** option and adds extended options.

The standard option is:

Option	Description
-c *cachename*	Specifies that the login context and associated credentials for cache name should be destroyed instead of those in the default cache.

The extended options are:

Option	Description
-C *cellname*	Destroys the login context and associated credentials for the specified cell *cellname*.
-m	Destroys the user's login context and associated credentials for all cells in which the user has a login context.

dce_login

The **dce_login** command validates a user's identity, obtains the user's network credentials, and establishes a DCE login context.

The user must supply a principal_name (user name) and password. If you do not supply these values as command-line arguments, **dce_login** will prompt for them.

NetSEAT client supports the following standard **dce_login** options:

Option	Description
-exec *command_string*	Executes the command specified by *command_string* after login. If *command_string* is specified without a full path name, the path prefix is obtained by searching the directories according to the PATH variable.
-k *keytab_file_name*	Instructs **dce_login** to obtain user (principal) name and password from the keytab file *keytab_file_name*.
-r	Refreshes the user's DCE login context before the user's ticket expires.

Policy Director supports the following extended dce_login option:

Option	Description
-C *cellname*	Specifies a user login into the cell *cellname* instead of logging into the default cell.

Note: NetSEAT client does not support the **dce_login -c** option.

Troubleshooting by using netseat_ping

NetSEAT client provides the **netseat_ping** utility to enable a user to obtain status information on DCE services in one or more cells. Use **netseat_ping** to determine if the following services are available:

- Security Services
- Time Services
- Cell Directory Services
- Directory Services Broker

To obtain status on services in all cells in which the user maintains a login context, type:

```
netseat_ping
```

For example, if you configure a NetSEAT client to participate in the cell `redback`, the following output will appear:

```
/.../redback:
       SecurityServers:
                    ncacn_ip_tcp:redback[ ] is available
                    ncadg_ip_udp:redback[ ] is available
       CdsServers:
                    ncacn_ip_tcp:redback[ ] is available
                    ncadg_ip_udp:redback[ ] is available
       TimeServers:
                    ncacn_ip_tcp:redback[ ] is available
                    ncadg_ip_udp:redback[ ] is available
       DsbServers:
                    ncacn_ip_tcp:redback[ ] is available (v3.0)
                    ncacn_ip_udp:redback[ ] is available (v3.0)
```

Policy Director supports the following **netseat_ping** options:

Option	Description
-C *cellname*	Generates a list of bindings for all servers that the user has configured for the cell *cellname*.
-t -C *cellname*	Displays the bindings for the Time Server in the cell *cellname*.
-s -C *cellname*	Displays the bindings for the Security Server in the *cellname*.
-c -C *cellname*	Displays the bindings for the CDS server in the cell *cellname*.
-d -C *cellname*	Displays the bindings for the DSB server in the cell *cellname*.

If there is more than one Security Server, Time Server, Cell Directory Server, or DSB in the cell, **netseat_ping** will attempt to ping all of them.

Chapter 21. NetSEAT: Directory Services Broker

This chapter provides an overview of the Directory Services Broker (DSB) and describes how to customize the DSB configuration for your environment.

This chapter includes:

Overview of the Directory Services Broker

Policy Director NetSEAT client off-loads RPC Name Service Interface functionality to the Directory Services Broker (DSB). The DSB acts as a Cell Directory Services (CDS) mid-tier server.

The NetSEAT client directs requests for the location of resources and services to the DSB. The DSB, in turn, contacts the secure domain's CDS to resolve the request. Then, the DSB returns the requested information to the system that runs the NetSEAT client.

Policy Director automatically installs and configures the DSB during Policy Director installation. Policy Director provides the DSB as part of the Management Server (IVMgr) package. No additional steps are necessary to use a DSB.

- NetSEAT clients, which serve as the support module for Policy Director servers and the Management Console, use the DSB.
- NetSEAT clients that use SSL tunnelling do not use the DSB.

The DSB can support a large number of NetSEAT clients. For larger secure domains, you can optimize performance by running the DSB on a server that provides Cell Directory Services.

To provide high availability or to balance the load in very large networks, an administrator might want to deploy more than one DSB in a secure domain. You can manually install and configure additional DSBs.

Directory Services Broker configuration options

The DSB runs as a daemon or as a service. You can configure the DSB to start automatically on system reboot. In most situations, the DSB does not require any administration.

Refer to the following sections an explanation of how the administrator can adjust DSB configuration parameters:

Setting the DSB port

The DSB listens on a port for requests. By default, the DSB randomly chooses a port.

Optionally, an administrator can specify the port number by typing a value in one of the following files:

UNIX: `/etc/services`

Windows: *`install-path`*`\system32\drivers\etc\services`

For example, to tell the DSB on a UNIX system to listen on port 5000, place the following entry in /etc/services:

```
dsb                5000/tcp              # Directory Services Broker
```

Specifying the DSB log-file location

The DSB uses DCE serviceability log files for logging notices, errors, warnings, and irrecoverable (fatal) errors. If your DCE installation uses DCE serviceability, DSB will write the output to one or more serviceability log files. The DCE serviceability log files are found in the directory *DCELOCAL*/var/svc. *DCELOCAL* represents the DCE installation directory.

There can be multiple files that reflect the status of the message, such as notice, error, or warning. You can specify the location of these files by placing entries in the routing file *DCELOCAL*/var/svc/routing.

For example:

`NOTICE:FILE:`*`DCELOCAL`*`/var/svc/notice`

Alternatively, you can specify the location by using the environment variable *SVC_NOTICE*. The environment-variable definition overrides the routing-file specification. For example, you might define a UNIX environment variable for notice-file specification, as follows:

`export SVC_NOTICE=FILE:`*`DCELOCAL`*`/var/svc/notice`

Example command line:

If you define NOTICE in a routing file, and you start the DSB from a command line, as follows:

`SVC_NOTICE=FILE:`*`DCELOCAL`*`/var/dsb/dsb.log dsb -q -f -U cell_admin -P *****`

The DSB will perform the following tasks:

- Silently configure and start itself.
- Log all notices to *DCELOCAL*/var/dsb/dsb.log.
- Log all errors, warnings, and irrecoverable (fatal) errors according to specification in the routing file.
- Do not display any logging output.

For complete information on the format of routing files and the use of the SVC_* group of environment variables, see the DCE serviceability information. You can find the following installation and configuration documentation about DCE on the IBM Policy Director Security Services CD in the /doc directory:

- DCE22_QuickBeginnings_AIX.pdf
- DCE22_QuickBeginnings_NT.pdf
- DCE20_InstallGuide_Solaris.pdf
- DCE20_ReleaseNotes_Solaris.pdf

Directory Services Broker command-line options

The following table describes the DSB command-line options:

Option	Description
-d	Run in the foreground, instead of as a UNIX daemon or a Microsoft Windows NT service. This option is used primarily for debugging.
-f	Force reconfiguration of the DSB's DCE security entries. This flag creates the Policy Director/dsb-servers group (unless the group has already been created), the key table file, and the intraverse/dsb/default/*fully_qualified_DNS_name* principal (user), where *fully_qualified_DNS_name* is the system on which the DSB is running. This option is used at the initial DSB startup. It can be called more than once without first removing the principal (user) entry, group entry, and key table files.
-h	Command line usage message.
-q	Send standard output to a log file instead of to stdout or stderr. No notices will go to the screen.
-r	Unconfigure the DSB. This option removes the DSB's DCE security entries (as described above in the **-f** option). The **-r** flag is used either alone or with **-q**. No other command line flags can be used with **-r**.
-t	Specify the number of server threads for the DSB to use. The number of threads defines the number of client requests that can be processed concurrently.
-P *password*	Specify the password for the DCE login principal (user). Used with the **-U** flag only.
-U *principal_name*	The DCE login principal (user) used to run the configuration of the DSB. Use this option to specify any user that has been delegated the authority to create the DSB security entries (as described above in the **-f** option).
-b	*Microsoft Windows only*—Specify the file name of the executable DSB binary. Use this option if the DSB is installed in a non-default location. This option stores the DSB location in the Microsoft Windows NT registry. Microsoft Windows NT uses the registry setting when configuring the DSB as a Microsoft Windows NT service.
-v	*Microsoft Windows only*—Display DSB version and DCE vendor message.

Appendix A. Policy Director administration using ivadmin

The **ivadmin** utility is a command-line equivalent of the Management Console that you can use to perform Policy Director administration tasks.

This chapter includes:

Introducing the ivadmin utility

The **ivadmin** utility is a command-line alternative to the Management Console. Administrators who want to automate certain management functions can do so by writing scripts that use **ivadmin**.

Many **ivadmin** commands duplicate functions that are provided by the Management Console. In addition, **ivadmin** provides several advanced management functions that are not available through the Management Console. The IVBase package that is installed on any system that runs Policy Director also installs this utility as part of the same package.

Starting the ivadmin utility

To start the **ivadmin** utility, log in to a secure domain by using **dce_login**. Then, type:

UNIX: `# ivadmin`

Windows: `ivadmin`

The **ivadmin** prompt appears:

```
ivadmin>
```

At this prompt, type appropriate commands, options, and arguments. Refer to the command tables in "Using the ivadmin commands" on page 268.

For example, to see the **ivadmin** help messages, type:

```
ivadmin> help
```

Exiting the ivadmin utility

To exit the utility and return to the command prompt, type the **ivadmin exit** command:

```
ivadmin> exit
```

Using the ivadmin commands

The **ivadmin** commands include:

- "Server commands"
- "Object commands" on page 270
- "Action commands" on page 271
- "ACL commands" on page 272
- "NetSEAL commands" on page 274
- "Configuration management commands" on page 277
- "User management commands" on page 278
- "Group management commands" on page 282
- "Resource management commands" on page 285
- "Registry policy management commands" on page 290

Note: All **ivadmin** commands must be entered on one line as a single command. Some of the examples used in this book are long and are shown as wrapping to the next line.

Server commands

The **ivadmin server** commands provide functionality that is not currently handled by the Management Console:

Command	Description
server delete *server-name*	
	Deletes a Policy Director server. Usually this command is used internally by an uninstall program, except for external authorization servers. Use the following command syntax for an external authorization server: `server delete /ExternAuthzn/`*`server-name`*
server disable*server-name*	
	Disables the specified server..
server enable*server-name*	
	Enables the specified server..
server flush_logs *server-name*	
	Writes the WebSEAL server logs from memory to hard disk. Enables immediate tracking of server events.
server list	
	Lists all configured servers.
server modify *server-name* **baseurl** *mount-point*	
	Specifies the branch of the ACL space to be used by this server. For use with replicated WebSEAL servers. Designates the specified branch *mount-point* to be the master branch used by the Management Console for ACL administration. ACLs in this branch are applied to all replicated servers junctioned to this mount point (junction point); the replicated servers immediately reflect all ACL changes. Note that *mount-point* is relative to the /WebSEAL container object and must be located in the WebSEAL directory (not in any subdirectories).

server register dbreplica *server-name ns-location server-principal server-host*	
	Registers a new authorization policy database replica.
server register externauth *server-name ns-location server-group character description*	
	Registers an external authorization server with Policy Director. Use this command to inform the Policy Director Authorization Service that the external authorization server exists and must be consulted when resolving authorization privileges on protected objects. See “Registering an external authorization service” on page 138 for more information.
server register http *server-name ns-location server-principal http-port https-port*	
	Registers a new third-party HTTP server with Policy Director.
server register netseal *server-name ns-location server-principal server-host*	
	Registers a new NetSEAL server. **Warning:** Do not use this command manually. It is reserved for use by the IVTrap package installation.
server register password-mgmt *server-name ns-location server-principal*	
	Registers a new external password management server with Policy Director.
server register webseal *server-name ns-location server-principal server-host*	
	Registers a new WebSEAL server. **Warning:** Do not use this command manually. It is reserved for use by the IVWeb package installation.
server reload *server-name*	
	Forces the reload of the specified serverÆs authorization policy database.
server resume *server-name*	
	Resumes a suspended WebSEAL server.
server show *server-name*	
	Displays the specified server’s properties, such as name, description, hostname, NS location, principal, and root URL.
server start *server-name*	
	Starts the specified server. Starts secmgrd (NetSEAL and WebSEAL) and ivacld.
server status *server-name*	
	Determines if the server is running or stopped, and if the server’s ACL database replica has been updated with the latest changes.
server stop *server-name*	
	Stops the specified server. Stops secmgrd (NetSEAL and WebSEAL) and ivacld.
server suspend *server-name*	
	Suspends the specified WebSEAL server. Useful for performing server maintenance.

Technical Notes:

To display the properties of the WebSEAL server on the machine `chevelle`, type:

```
ivadmin> server show /WebSEAL/chevelle
Type: WebSEAL Server
Name: /WebSEAL/chevelle
Description: chevelle
Hostname: chevelle
NS Location: /.:/subsys/intraverse/secmgr/server/chevelle
Principal: secmgr/chevelle Root URL: /chevelle
```

Note that you must type the *server-name* argument in the exact format as displayed in the output of the **ivadmin server list** command.

For example:

```
ivadmin> server list
         /WebSEAL/chevelle
         /NetSEAL/chevelle
         /ExternAuthzn/timechecker
```

Object commands

The following **ivadmin object** commands provide the same functionality as the equivalent Object Space management commands on the Management Console:

Command	Description
object list *directory-name*	
	Lists the objects grouped under the indicated directory and displays the name of any ACL associated with each object. Note that this command does not expand the tree beyond this directory.
object show *object-name*	
	Displays the *object-name* information and the name of any ACL associated with it. If there is no associated ACL, the phrase `No ACL` appears.

Action commands

Use the following **ivadmin action** commands to define additional Policy Director authorization actions (permissions) on the Management Console.

For example, use the **ivadmin action** commands to add an external authorization mechanism to the list of available ACL permissions.

The **ivadmin action** commands provide functionality that is not currently handled by the Management Console:

Command	Description
action create *name description action-type*	
	Defines a new Policy Director authorization action (permission). Creates a new ACL permission code representing this action on the Management Console. The *name* argument designates the new single-letter permission code. The *description* argument provides the label for the new check box that will appear on the Management Console. The *action-type* argument provides a label for the action category as displayed on the Management Console. Example: `ivadmin> action create k time Ext-Authzn`
action delete *name*	
	Deletes an existing authorization action (permission) created by the **action create** command. Example: `ivadmin> action delete k`
action list	

	Lists all existing ACL actions (permissions) in the following format:
	permission name permission description action type
	
	Example:
	`ivadmin> action list`
	
	Would display information similar to:
	`r read WebSEAL`
	`...`

ACL commands

The following **ivadmin acl**commands provide the same functionality as the equivalent **ACLs** management commands on the Management Console:

Command	Description
acl attach *obj-name acl-name*	
	Attaches an ACL template to an object.
acl create *acl-name*	
	Creates a new ACL template in the ACL template database. Note that this command does not create ACL entries.
acl delete *acl-name*	
	Deletes an ACL template from the ACL template database.
acl detach *obj-name*	
	Detaches the current ACL template from the indicated object. Note that this command does not delete the ACL template from the ACL template database.
acl find *acl-name*	
	Finds and lists all objects that have the indicated ACL template attached.
acl list	
	Lists all ACL templates in the ACL template database.
acl modify *acl-name* **description** *desc*	
	Allows you to create or edit the description field associated with the indicated ACL template. One place this description appears is in the ACL Definition area of the Management Console's ACL management task panel.
acl modify *acl-name* **remove user** *user-name*	
	Allows you to remove an existing user ACL entry from the indicated ACL template definition.
acl modify *acl-name* **remove group** *group-name*	
	Allows you to remove an existing group ACL entry from the indicated ACL template definition.
acl modify *acl-name* **remove any-other**	
	Allows you to remove the any-authenticated ACL entry from the indicated ACL template definition.
acl modify *acl-name* **remove unauthenticated**	
	Allows you to remove the unauthenticated ACL entry from the indicated ACL template definition.

acl modify *acl-name* **set user** *user-name perms*	
	Allows you to create or edit a user ACL entry (`pubs`) in the indicated ACL template definition, where the permissions (*perms*) are `bPTr`. Example: `ivadmin> acl modify pubs set user peter bPTr`
acl modify *acl-name* **set group** *group-name perms*	
	Allows you to create or edit a group ACL entry in the indicated ACL template definition. Example: `ivadmin> acl modify pubs set group sales Tr`
acl modify *acl-name* **set any-other** *perms*	
	Allows you to create or edit the any-authenticated ACL entry (`pubs`) in the indicated ACL template definition. Example: `ivadmin> acl modify pubs set any-other r`
acl modify *acl-name* **set unauthenticated** *perms*	
	Allows you to create or edit the unauthenticated ACL entry in the indicated ACL template definition. Example: `ivadmin> acl modify pubs set unauthenticated r`
acl show *acl-name*	
	Lists the complete set of entries that make up the definition of the indicated ACL template. `ivadmin> acl show pubs`

NetSEAL commands

You can perform the following NetSEAL administration tasks by using the **ivadmin** utility:

- "Managing protected networks".
- "Managing NetSEAL junctions" on page 275.
- "Managing protected ports" on page 276.
- "Managing protected port aliases" on page 277.

Managing protected networks

You can think of *networks* as non-Policy Director servers that are protected by NetSEAL. Use the **ivadmin netseal** commands to add, delete, and list protected networks.

Command	Description
netseal network add *network netmask* **[***network-alias***]**	
	Creates a new network to be protected by NetSEAL. The network and netmask pair are standard IP network address numbers and netmasks. The optional network alias name can be used to identity this network. If no alias is specified, the network must be identified by the network and netmask pair. Errors are returned if the network already exists.
netseal network delete *network-id*	
	Deletes the specified network from the system where *network-id* can correspond to one of the following: • *network/netmask* pair • *network-alias* An error will be returned if the network is not found within the database.
netseal network list	
	Displays all of the networks in the database, which includes the network and netmask pair as well as any defined aliases.

Managing NetSEAL junctions

NetSEAL junctions determine the direction communication should travel through a Policy Director server. You can create junctions between two Policy Director servers or between a Policy Director server and a network. Use a GSS tunnel to secure communication across a junction between two Policy Director servers.

Use the **ivadmin netseal junction** commands to add, delete, and list NetSEAL junctions.

Command	Description
netseal junction add *hostname destination*	
	Creates a junction from a NetSEAL server to a specified destination where *hostname* is the name of the NetSEAL server minus the domain name, and *destination* can correspond to one of the following: • *pd-server* • *network/netmask* pair • *network-alias* Errors are returned if the junction already exists, the host server does not exist, or the destination does not exist.
netseal junction delete *hostname* **destination**	
	Deletes the junction from a NetSEAL server to the specified destination where *hostname* is the name of the NetSEAL server minus the domain name, and *destination* can correspond to one of the following: • *pd-server* • *network/netmask* pair • *network-alias* An error is returned if the junction does not currently exist. The command has no effect on current connections.
netseal junction list *hostname*	
	Displays all of the NetSEAL junctions for the specified NetSEAL server.

Managing protected ports

The Policy Director NetSEAL server provides security services for specific ports, hosts, and networks. For example, you can configure the NetSEAL server to secure Telnet traffic on a specific port.

Use the **ivadmin netseal port** commands to add, delete, and list protected ports. You can specify ports for Policy Director servers or networks.

Command	Description
netseal port add *destination port-id*	
	Protects connections to the specified destination on the specified port-id where *destination* can correspond to one of the following: • *pd-server* • *network/netmask* pair • *network-alias* And where *port-id* can correspond to one of the following: • *port* • *port-range* • *port-alias* An error is returned if the port is already being protected, if the server does not exit, or the port alias does not exist.
netseal port delete *destination port-id*	
	Stops protecting connections to the specified destination on the specified port where *destination* can correspond to one of the following: • *pd-server* • *network/netmask* pair • *network-alias* And where *port-id* can correspond to one of the following: • *port* • *port-range* • *port-alias* An error is returned if the port is not already being protected, if the server does not exist, or the port alias does not exist.
netseal port list *destination*	
	Displays a list of all ports for the specified destination where *destination* can correspond to one of the following: • *pd-server* • *network/netmask* pair • *network-alias* An error is returned if the server does not exist.

Note: The **port range** should be expressed as two port numbers separated by a dash (such as 22-88).

Managing protected port aliases

Use the **ivadmin port-alias** commands to add, delete, and list port aliases. Use *port aliases* to identify trapped port ranges in a more meaningful manner.

Command	Description
netseal port-alias add *port-spec port-alias*	
	Creates a new port alias for the specified port specification where *port-spec* can correspond to one of the following: • *port* • *port-range* An error will be returned if the port range is already aliased by another name.
netseal port-alias delete *port-id*	
	Deletes the port alias for the specified port-id from the system where *port-id* can correspond to one of the following: • *port* • *port-range* • *port-alias* The *port-id* can correspond to either a port, a port range, or a port alias name. An error will be returned if the port alias is not found within the database.
netseal port-alias list	
	Displays all of the port aliases found within the database.

Configuration management commands

The **ivadmin admin** configuration management commands display information about the server.

Command	Description
admin show configuration	

	Displays information about whether the user registry is contained in LDAP or DCE.
	
	Example:
	`ivadmin> admin show configuration`
	
	Would produce output similar to:
	`LDAP: TRUE`
	`SECAUTHORITY: Default`
	`GSO: TRUE`

User management commands

The following **ivadmin user** commands provide the same functionality as the equivalent **Users** management commands on the Management Console. This set of management commands controls user entries in the default LDAP registry.

A *user* is a Policy Director user. A *GSO user* is a Policy Director user that additionally has the authority to work with Web resources, such as a Web server.

Command	Description
user create [-gsouser] ***user-name dn cn sn pwd***	

	Creates a new Policy Director user (secUser) account in the LDAP user registry whose distinguished name does not already exists in the default LDAP registry database.
	
	The **-gsouser** argument is optional. The dash (-) is required for optional commands. When the -gsouser argument is specified, the user is also made a GSO user (gsouser).
	
	The *user-name* argument is the name for the user being created. This name must be unique.
	
	The *dn* argument is the LDAP distinguished name assigned to the user being created (for example, `cn=Diana Lucas,ou=Austin,o=Wesley Inc,c=US`). The DN must be unique.
	
	The *cn* argument is the common name assigned to the user being created (for example, `Diana Lucas`).
	
	The *sn* argument is the surname of the user being created (for example, `Lucas`).
	
	The *pwd* argument is the password you set for this new user. Passwords must adhere to the password policies set by the Policy Director administrator (for example, `mypasswd`).
	
	Example:
	`ivadmin> user create -gsouser dlucas`
	`cn=Diana Lucas,ou=Austin,o=Wesley Inc,c=US`
	`"Diana Lucas" Lucas mypasswd`
	
	To make the user account valid, you must manually activate this user by modifying the user information. To change the information, you must set the account-valid flag to “yes.”
	
	To add a description about a user, you must use the **ivadmin modify user** command to change the user account information.

user import [-gsouser] ***user-name dn***	

	Allows an existing user whose distinguished name already exists in the default LDAP registry database to be updated with Policy Director information so the user can participate in the secure domain.
	
	Example:
	`ivadmin> user import -gsouser mlucaser`
	`cn=Mike Lucaser,ou=Austin,o=Wesley Inc,c=US`

user modify *user-name* description *description*	
	Adds a description that provides information that makes it easier for the administrator to identify this user. Example: `ivadmin> user modify dlucas description "Diana Lucas, Credit Dept HCUS`
user modify *user-name* password *password*	
	Changes the password of the user from the current password to a new password. You are not requested to confirm the password. Example: `ivadmin> user modify dlucas password newpasswd`
user modify *user-name* authentication-mechanism *mech*	
	Changes the mechanism used for authentication. Example: `ivadmin> user modify dlucas authentication-mechanism dce`
user modify *user-name* account-valid {yes | no}	
	Specifies whether an account is active or inactive. To activate the account, select “yes”; to deactivate the account, select “no.” Example: `ivadmin> user modify dlucas account-valid yes`
user modify *user-name* password-valid {yes | no}	
	Specifies whether a password is active or inactive. To activate the account, select “yes”; to deactivate the account, select “no.” Example: `ivadmin> user modify dlucas password-valid no`
user modify *user-name* gsouser {yes | no}	
	Specifies whether the Policy Director user specified is also a GSO user. To add the user as a GSO user, select “yes”; to remove the user as a GSO user, select “no.” Example: `ivadmin> user modify dlucas gsouser no`
user delete *user-name*	
	Deletes an existing user account from the LDAP user registry. Deleting a Policy Director user account deletes the GSO user account information from the default LDAP registry also. Example: `ivadmin> user delete dlucas` Any resource credentials associated with a user account are automatically removed at the same time the user account is deleted.

user show *user-name*	
	Displays the user account information for the user specified. Example: `ivadmin> user show dlucas`
user show-dn *dn*	

	Provides additional information about the user when you specify the distinguished name (DN).
	
	Example:
	`ivadmin> user show-dn`
	`cn=Diana Lucas,ou=Austin,o=Wesley Inc,c=US`

user show-groups *username*	

	Displays the groups of which the specified user is a member.
	
	Example:
	`ivadmin> user show-groups dlucas`
	
	Would produce a list similar to:
	`sales`
	`credit`
	`engineering`

user list *pattern max-return*	

	Generates a list of all configured user accounts, listed by user names, for the pattern you specify. The list displays in the order the user accounts were created.
	
	The *pattern* argument allows you to specify a pattern for what you want to list. The wildcard matches against the user name. The pattern can include a mixture of wildcards and string constants, and is case sensitive (for example, `*luca*`).
	
	The *max-return* argument limits how many entries are found and returned for a single request (for example, `2`). This number is also governed by the configuration on the LDAP server. where you can specifies the maximum number of results that can be returned as part of a search operation. Policy Director returns the minimum of *max-return* and the configured value in the LDAP server.
	
	Example:
	`ivadmin> user list *luca* 2`
	
	Would produce a list similar to:
	`dlucas`
	`mlucaser`

user list-dn *pattern max-return*	

	If only a portion of the distinguished name is known, generates a list of all configured user accounts, listed by distinguished names. The list displays in the order the user names were created.
	
	The wildcard matches against the CN portion of the user's distinguished name, excluding the `cn=` portion.
	
	The *max-return* number is also governed by the configuration on the LDAP server. where you can specifies the maximum number of results that can be returned as part of a search operation. Policy Director returns the minimum of *max-return* and the configured value in the LDAP server.
	
	Example:
	`ivadmin> user list-dn *luca* 2`
	
	Would produce a list similar to:
	`Diana Lucas,ou=Austin,o=Wesley Inc,c=US`
	`Mike Lucaser,ou=Austin,o=Wesley Inc,c=US`

Technical Notes:

Note that you must enter the *dn* argument for the **user show-dn** command and **user show-groups-dn** command in the exact format. Use double quotes (″) when the *dn* argument contains spaces.

Example:

```
cn=Diana Lucas,ou=Austin,o=Wesley Inc,c=US
```

The **user show** and the **user show-dn** commands display information similar to the following for the user `Diana Lucas`:

```
Login ID: dlucas
LDAP dn: cn=Diana Lucas,ou=Austin,o=Wesley Inc,c=US
LDAP cn: Diana Lucas
LDAP sn: Lucas
Description: Diana Lucas, Credit Dept HCUS
IS SecUser: true
IS GSO user: false
Account valid: true
Password valid: true
Authorization mechanism:
Default: LDAP
```

Group management commands

The following **ivadmin group** commands are equivalent to the **Groups** management commands on the Management Console. This set of management commands control group entries in the LDAP directory registry.

A *group* is a set of Policy Director user accounts that have similar attributes. Groups allow the administrator to use a group name in an access control list (ACL) instead of listing all users individually.

You can delete or change any group entries. In addition, you can display information about a group or group membership. As the administrator, you can also list all configured groups.

Command	Description
group create *groupname dn cn*	
	Creates a new Policy Director group (SecGroup) in the LDAP user registry. The *groupname* argument is the name for the group being created. This name must be unique. The *dn* argument is the LDAP distinguished name assigned to the access group being created (for example `cn=credit,ou=Austin,o=Wesley Inc,c=US`). The *cn* argument is the common name assigned to the group (for example, `Credit`. Example: `ivadmin> group create credit cn=credit,ou=Austin,o=Wesley Inc,c=US Credit`
group import *groupname dn*	
	Imports the information about an existing LDAP registry group to create a Policy Director group. The group must already exist in the LDAP registry before a Policy Director group can import the information and create the group. The name of the group being created must be unique. Example: `ivadmin> group import engineering cn=engineering,ou=Austin,o=Wesley Inc,c=US`
group modify *groupname* description *description*	
	Adds a description for the specified group that makes it more easily identifiable by the Policy Director administrator. Example: `ivadmin> group modify credit description "Credit, Dept HCUS`

group modify *groupname* add *user-name*	
	Adds a new user to the specified group. Example: `ivadmin> group modify engineering add dlucas`
group modify *groupname* remove *user-name*	
	Deleting an existing user from the specified group. Example: `ivadmin> group modify engineering remove dlucas`
group delete *groupname*	
	Deletes an existing group and any entries associated with the group. Example: `ivadmin> group delete engineering`
group show *groupname*	
	Displays the details about a specified group. Example: `ivadmin> group show credit`
group show-dn *dn*	
	Provides the group name for the distinguished name specified. Example: `ivadmin> group show-dn cn=credit,ou=Austin,o=Wesley Inc,c=US`
group show-members *groupname*	

	Displays the members of the group specified, listed by distinguished names.
	Example:
	`ivadmin> group show-members credit`
	Would show information similar to:
	`dlucas`
	`mlucaser`

group list *pattern max-return*	

	Generates a list of all configured groups whose names match the specified pattern, listed by group names.
	
	The *pattern* argument allows you to specify a pattern for what you want to list. The wildcard matches against the group name. The pattern can include a mixture of wildcards and string constants, and is case sensitive (for example, `*Austin*`).
	
	The *max-return* argument limits how many entries are found and returned for a single request (for example, `2`). This number is also governed by the configuration on the LDAP server. where you can specifies the maximum number of results that can be returned as part of a search operation. Policy Director returns the minimum of *max-return* and the configured value in the LDAP server.
	
	Would show information similar to:
	
	`credit`
	`marketing`

group list-dn *pattern max-return*	

	If a portion of the distinguished name is known, generates a list of all configured groups, listed by distinguished names for the pattern specified.
	
	The wildcard matches against the CN portion of the group's distinguished name, excluding the `cn=` portion.
	
	The *max-return* number is also governed by the configuration on the LDAP server. where you can specifies the maximum number of results that can be returned as part of a search operation. Policy Director returns the minimum of *max-return* and the configured value in the LDAP server.
	
	Example:
	
	`ivadmin> group list-dn *t* 2`
	
	Would show information similar to:
	
	`cn=credit,ou=Austin,o=Wesley Inc,c=US`
	`cn=marketing,ou=Boston,o=Austin Sale,c=US marketing`

Technical Notes:

Note that you must enter the *dn* argument in the exact format as displayed in the output of the **group show-dn** command. Use double quotes (") when the *dn* includes a space.

Example:

```
cn=credit,ou=Austin,o=Wesley Inc,c=US
```

The **group show** and the **group show-dn** commands that display information similar to the following for the group `credit`:

```
Group ID: credit
LDAP dn: cn=credit,ou=Austin,o=Wesley Inc,c=US
Description: Credit, Dept HCUS
LDAP cn: credit
Is SecGroup: true
```

Resource management commands

The following Policy Director **ivadmin** commands are a set of management commands that control resource-related information.

Resource-related information includes:

- “Managing resources”
- “Managing resource groups” on page 286
- “Managing resource credentials” on page 288

Managing resources

The following **ivadmin rsrc** commands allow the administrator to manage different resources, such as Web servers for GSO users.

A *resource* is a Web server. The **-T** identifier in a smart junction definition identifies the Web server.

An **ivadmin rsrc** command identifies the name of the Web resource.

The following **ivadmin rsrc** commands provide the same functionality as the equivalent **GSO Resources** management commands on the Management Console.

Command	Description
rsrc create *resource-name* [-desc *description*]	
	Creates and names a Web server as a resource. The *resource-name* argument is the name given to the Web resource to identify it (for example, `engwebs01`). The *description* argument is an optional description that can be added to more easily identify this resource to the Policy Director administrator. Any optional parameter must be preceded with a dash (-). Descriptions containing a space must be enclosed in double quotes marks (″). `ivadmin> rsrc create engwebs01 -desc "Engineering Web server - Room 4807"`
rsrc delete *resource-name*	
	Deletes the named resource, including the description information. The resource must exist or an error is displayed. Example: `ivadmin> rsrc delete engwebs01`
rsrc list	

	Displays the names of all Web resources defined in the LDAP directory, listed by resource name.
	
	Example:
	
	`ivadmin> rsrc list`
	
	Would provides information similar to:
	
	`engwebs01`
	`engwebs02`
	`engwebs03`

rsrc show *resource-name*	

	Displays the Web resource information for the resource named.
	
	The resource must exist or an error message displays.
	
	Example:
	`ivadmin> rsrc show engwebs01`
	
	Would provide information similar to:
	`Web Resource Name: engwebs01`
	`  Description: Engineering Web server - Room 4807`

Managing resource groups

The following **ivadmin rsrcgroup** commands are equivalent to the **GSO Resource Groups** management commands that control GSO Resources information. These commands allow the administrator to manage different resource group-related attributes.

A *resource group* refers to a group of Web servers, where all the servers in the group have the same sets of user IDs (userids) and password. You can create a single resource credential for all the resources in the resource group. Policy Director uses a single resource credential for a resource group instead of a resource-credential for each resource in the resource group.

The following **ivadmin rsrcgroup** commands provide the same functionality as the equivalent **GSO Resource Groups** management commands on the Management Console.

Command	**Description**
rsrcgroup create *resource-group-name* [-desc *description*]	
	Creates and names a Web resource group. The *resource-group-name* argument is the name of the resource group. The *description* argument is an optional description that can be added to identify this resource group. The optional **-desc** parameter must be preceded with a dash (-). Descriptions that have spaces need to be enclosed in double quotes. Example: `ivadmin> rsrcgroup create webs4807 -desc "Web servers, Room 4807"`
rsrcgroup delete *resource-group-name*	
	Deletes the named resource group, including description information. The resource group must exist. Example: `ivadmin> rsrcgroup delete webs4807`

rsrcgroup modify *resource-group-name* add rsrcname *resource-name*	
	Adds a Web resource to an existing resource-group. The resource group must exist. Example: `ivadmin> rsrcgroup modify webs4807 add rsrcname engwebs02`
rsrcgroup modify *resource-group-name* remove rsrcname *resource-name*	
	Deletes a Web resource name from an existing resource-group. Example: `ivadmin> rsrcgroup modify webs4807 remove rsrcname engwebs02`
rsrcgroup list	

	Displays the names of all Web resource groups defined to the LDAP directory. Information following “list” is ignored.
	
	Example:
	`ivadmin> rsrcgroup list`
	
	Would provide information similar to:
	`webs4807`
	`websbld3`
	`websbld4`

rsrcgroup show *resource-group-name*	

	Displays the Web resource group information for the specified resource group.
	
	The resource group must exist or an error message displays.
	
	Example:
	`ivadmin> rsrcgroup show webs4807`
	
	Would provide information similar to:
	`Resource Group Name: webs4807`
	`Description: Web servers, Room 4807`
	`Resource Members:`
	`engwebs01`
	`engwebs02`
	`engwebs03`

Managing resource credentials

The following **ivadmin rsrccred** commands allow the administrator to manage different resource credential-related attributes.

A *resource credential* provides a user identification and password for a GSO user-specific resource, such as a Web server or a group of Web servers.

You can only specify “web” or “group” as types of resource when using the **ivadmin rsrccred** commands.

Note: The resource, or resource group, must exist before you can apply the resource-credential commands to it.

Command	Description
rsrccred create *resource-name* rsrcuser *resource-userid* rsrcpwd *resource-password* rsrctype {web | group} user *user-name*	
	Creates and names a resource credential. Both the user and the resource (or resource group) must already exist in order to create the resource-credential. If the user, resource, or resource group does not exist or is not specified, an error message is displayed. The types of resource include only “web” or “group” resources when referring to resource-credential management commands. The *resource-name* argument is the name given to the resource when the resource was created (for example, `engwebs01`). The *resource-userid* argument is the unique user identification (userid) for the user at the Web server (for example, `4807ws01`. The *resource-password* argument is the password for a user at the Web server (for example, *`rsrcpwd`*). The *user-name* argument is the name of the user for whom the resource-credential information applies (for example, `dlucas`). Example: `ivadmin> rsrccred create engwebs01 rsrcuser 4807ws01 rsrcpwd rsrcpwd rsrctype web user dlucas`
rsrccred delete *resource-name* rsrctype {web | group} user *user-name*	
	Deletes only the resource-credential information for an existing user. The type of resource must match the resource type assigned when the resource was first created, such as “web” or “group.” Example (entered as one line): `ivadmin> rsrccred delete engwebs01 rsrctype group user dlucas`

rsrccred list user *user-name*	

	Displays the names of all defined resources and their type for the specified user.
	
	Example:
	`ivadmin> rsrccred list user dlucas`
	
	Would provide information similar to:
	`Resource name: engwebs01`
	`  Resource Type: group`
	`Resource name: engwebs02`
	`  Resource Type: web`

rsrccred modify *resource-name* rsrctype {web | group} set [-rsrcuser *resource-userid*] [-rsrcpwd *resource-password*] user *user-name*	

	Changes the user ID and password resource credential information for the named resource.
	
	To change or reset the resource userid of the user or password information, these optional commands must be preceded by a dash (-). Before the resource-credential information can be changed, the resource or resource group, and the user must already exist.
	
	The type of resource you specify must match the resource type assigned when it was first created, such as “web” or “group.”
	
	Example (entered as one line):
	`ivadmin> rsrccred modify engwebs01 rsrctype group`
	`  set -rsrcuser 4807ws01 -rsrcpwd newrsrpw`
	`  user dlucas`

rsrccred show *resource-name* rsrctype {web | group} user *user-name*	

	Displays the resource-credential information for a specified user. The resource-credential and the user must both exist or an error message displays.
	
	Example:
	`ivadmin> rsrccred show webs4807 rsrctype group user dlucas`
	
	Would provide information similar to:
	`Resource Name: engwebs01`
	`Resource Type: group`
	`Resource User Id: dlucas`

Registry policy management commands

The **ivadmin policy** commands are a set of management commands that control general policy information for Policy Director users. The administrator can manage the following policy attributes:

- “Managing login policies”.
- “Managing password policies” on page 291

A *policy* defines the set of constraints placed on user accounts and passwords in order to improve the overall security of the system. These constraints can be imposed generally (globally across every user in the system) or specifically (only to a specified user). If the user has a specific policy applied, this specific policy takes precedence over any general policy that also might be defined. The precedence applies, regardless of whether the specific policy is more or less restrictive than the general policy.

Managing login policies

The following **ivadmin policy** commands allow the administrator to manage login-related policies.

Use the login-related **policy** management commands to create new login policies or copy existing login policies. In addition, you can display information about a user account’s login policy.

For login-related policies, Policy Director defines relative time as DDD-hh:mm:ss, and defines absolute time as YYYY-MM-DD-hh:mm:ss when referring to **policy** management commands.

Command	Description
policy set max-account-age [*relative-time*] [-user *user-name*]	
policy get max-account-age [-user *user-name*]	

	Manages the policy controlling the time period, such as a maximum number of days and hours, before the account belonging to the user expires.
	
	Examples:
	`ivadmin> policy set max-account-age 031-12:30:00 -user dlucas`
	
	Or:
	`ivadmin> policy get max-account-age -user dlucas`

policy set account-expiry-date [*absolute-time*] [-user *user-name*]	
policy get account-expiry-date [-user *user-name*]	

	Manages the policy controlling the absolute date and time an individual user account is to expire. Can also be used to specify when all user accounts are to expire at the same time.
	
	Examples (entered on one line):
	`ivadmin> policy set account-expiry-date 1999-12-30-23:30:00 -user dlucas`
	
	Or:
	`ivadmin> policy get account-expiry-date -user dlucas`

Managing password policies

The following **ivadmin policy** commands allow the administrator to manage different password-related policy attributes.

For password-related policies, Policy Director defines relative time as DDD-hh:mm:ss when referring to **policy** management commands.

Command	Description
policy set max-password-age [*relative-time*] [-user *user-name*]	
policy get max-password-age [-user *user-name*]	

	Manages the policy controlling the maximum time before a password must be changed. The time specified can be a *relative* time expressed in days, hours, and minutes.
	
	As the administrator, you can specify a specific user name or apply the policy globally to all users listed in the registry. The relative-time argument is the maximum time, expressed in days, hours, and minutes in this format: DDD-hh:mm:ss
	
	Examples (entered on one line):
	`ivadmin> policy set max-password-age 031-08:30:00 -user dlucas`
	
	Or:
	`ivadmin> policy get max-password-age -user dlucas`

Appendix B. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department LZKS
11400 Burnet Road
Austin, TX 78758
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written.

These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form

without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *enter the year or years.* All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AIX
IBM
FirstSecure
Policy Director
SecureWay

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark in the United States, other countries, or both and is licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

AuthAPI	DASCOM, Inc.
DASCOM	DASCOM, Inc.
IntraVerse	DASCOM, Inc.
Internet Information Server (IIS)	Microsoft Corporation
Internet Explorer	Microsoft Corporation
Netscape	Netscape Communications Corporation
Netscape Communicator	Netscape Communications Corporation
Netscape logos	Netscape Communications Corporation
Netscape FastTrack	Netscape Communications Corporation
Netscape Navigator	Netscape Communications Corporation
NetSEAL	DASCOM, Inc.
NetSEAT	DASCOM, Inc.
Smart Junctions	DASCOM, Inc.
Solaris	Sun Microsystems, Inc
WebSEAL	DASCOM, Inc.

Glossary

This glossary defines the terms and abbreviations in this book that may be new or unfamiliar and terms that may be of interest. It includes terms and definitions from:

- The IBM Dictionary of Computing, New York: McGraw-Hill, 1994.
- The American National Standard Dictionary for Information Systems, ANSI X3.172–1990, American National Standards Institute (ANSI), 1990.
- The Answers to Frequently Asked Questions, Version 3.0, California: RSA Data Security, Inc., 1998.

A

access control list (ACL). A mechanism for limiting the use of a specific resource to authorized users.

ACL. Access control list.

action history. Accumulated events in the life cycle of a credential.

American National Standard Code for Information Interchange (ASCII). The standard code that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set uses a coded character set that consists of 7-bit coded characters (8 bits including a bit for parity checking). The character set consists of control characters and graphic characters.

American National Standards Institute (ANSI). An organization that establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States. It consists of producers, consumers, and general interest groups.

ANSI. American National Standards Institute.

API. Application program interface.

applet. A computer program that is written in Java and runs inside a Java-compatible Web browser. Also known as a Java applet.

application program interface (API). In Policy Director, a functional interface that allows an application program that is written in a high-level language to use specific Policy Director functions. The standards-based Policy Director Authorization API allows applications to make calls to the centralized Policy Director authorization service. By using these calls, you eliminate the necessity for developers to write authorization code for each new application. The Policy Director Authorization API allow businesses to standardize all applications on a trusted authorization framework. Using the Policy Director Authorization API, businesses can provide more control over access to resources on their networks. See the *Policy Director Programmer's Guide and Reference* for complete information and descriptions of the Policy Director Authorization API.

ASCII. American National Standard Code for Information Interchange.

audit log. A table in a database that stores one record per audit event.

audit server. A server that receives audit events from audit clients and writes them to an audit log.

audit trail. Data, in the form of a logical path, that links a sequence of events. An audit trail enables tracing of transactions or the history of a given activity.

authentication. The process of reliably determining the identity of a communicating party.

authorization. Permission to access a resource.

B

base64 encoding. A common means of conveying binary data with MIME.

browser. *See* Web browser.

browser certificate. A digital certificate is also known as a client-side certificate. It is issued by a CA through an SSL-enabled Web server. Keys in an encrypted file enable the holder of the certificate to encrypt, decrypt, and sign data. Typically, the Web browser stores these keys. Some applications permit storage of the keys on smart cards or other media. *See also* digital certificate.

C

CA. Certificate authority.

CA certificate. A certificate your Web browser accepts, at your request, from a CA it does not recognize. The browser can then use this certificate to authenticate communications with servers that hold certificates issued by that CA.

CAS. Credentials Acquisition Service.

CAS server. The server for the Policy Director Credentials Acquisition Service (CAS) component.

certificate authority (CA). The software responsible for following an organization's security policies and assigning secure electronic identities in the form of

certificates. The CA can process requests from Registration Authorities (RAs) to issue, renew, and revoke certificates. The CA interacts with the IBM SecureWay Trust Authority product's RA to publish certificates and CRLs in the Directory. *See also* digital certificate.

certificate extension. An optional feature of the X.509v3 certificate format that provides for the inclusion of additional fields in the certificate. There are standard extensions and user-defined extensions. Standard extensions exist for various purposes, including key and policy information, subject and issuer attributes, and certification path constraints.

certificate policy. A named set of rules that indicates the applicability of a certificate to a particular class of applications that have common security requirements. For example, a certificate policy might indicate whether a particular certification type allows a user to conduct transactions for goods within a given price range.

certificate profile. A set of characteristics that define the type of certificate wanted (such as SSL certificates or IPSec certificates). The profile aids in managing certificate specification and registration. The issuer can change the names of the profiles and specify characteristics of the desired certificate, such as the validity period, key usage, distinguished name (DN) constraints, and so forth.

certificate revocation list (CRL). A digitally signed, time-stamped list of certificates that the certificate authority has revoked. The certificates in this list should be considered unacceptable. *See also* digital certificate.

certification. The process during which a trusted third party issues an electronic credential that vouches for an individual, business, or organizational identity.

CGI. Common Gateway Interface.

chain validation. The validation of all CA signatures in the trust hierarchy through which a given certificate was issued. For example, if a CA was issued its signing certificate by another CA, both signatures are validated during validation of the certificate that the user presents.

class. In object-oriented design or programming, a group of objects that share a common definition and therefore share common properties, operations, and behavior.

cleartext. Data that is not encrypted. *Synonym for* plaintext.

client. (1) A functional unit that receives shared services from a server. (2) A computer or program that requests a service of another computer or program.

client/server. A model in distributed processing in which a program at one site sends a request to a program at another site and waits for a response. The requesting program is called a client; the answering one is called a server.

code signing. A technique for signing executable programs with digital signatures. Code signing is designed to improve the reliability of software that is distributed over the Internet.

Common Cryptographic Architecture (CCA). IBM software that enables a consistent approach to cryptography on major IBM computing platforms. It supports application software that is written in a variety of programming languages. Application software can call on CCA services to perform a broad range of cryptographic functions, including DES and RSA encryption.

Common Gateway Interface (CGI). Standard method of transmitting information between Web pages and Web servers.

confidentiality. The property of not being divulged to unauthorized parties.

credential. Confidential information used to prove one's identity in an authentication exchange. In environments for network computing, the most common type of credential is a certificate that a CA has created and signed.

Credentials Acquisition Service (CAS). The Policy Director Credentials Acquisition Service (CAS) component.

CRL. Certificate revocation list.

CRL publication interval. Set in the CA configuration file, the interval of time between periodic publications of the CRL to the Directory.

cross-certification. A trust model whereby one CA issues to another CA a certificate that contains the public key associated with its private signature key. A cross-certified certificate allows client systems or end entities in one administrative domain to communicate securely with client systems or end entities in another domain.

cryptographic. Pertaining to the transformation of data to conceal its meaning.

cryptography. In computer security, the principles, means, and methods for encrypting plaintext and decrypting encrypted text.

D

daemon. A program that carries out tasks in the background. It is implicitly called when a condition occurs that requires its help. A user need not be aware of a daemon, because the system usually spawns it

automatically. A daemon might live forever or the system might regenerate it at intervals.

The term (pronounced *demon*) comes from mythology. Later, it was rationalized as the acronym DAEMON: Disk And Execution MONitor.

Data Encryption Standard (DES). An encryption block cipher, defined and endorsed by the U.S. government in 1977 as an official standard. IBM developed it originally. DES has been extensively studied since its publication and is a well-known and widely used cryptographic system.

DES is a symmetric cryptographic system. When it is used for communication, both the sender and receiver must know the same secret key. This key is used to encrypt and decrypt the message. DES can also be used for single-user encryption, such as to store files on a hard disk in encrypted form. DES has a 64-bit block size and uses a 56-bit key during encryption. It is was originally designed for implementation in hardware. NIST has recertified DES as an official U.S. government encryption standard every five years.

data storage library. A module that provides access to persistent data stores of certificates, CRLs, keys, policies, and other security-related objects.

decrypt. To undo the encryption process.

DER. Distinguished Encoding Rules.

DES. Data Encryption Standard.

digital certificate. An electronic credential that is issued by a trusted third party to a person or entity. Each certificate is signed with the private key of the CA. It vouches for an individual, business, or organizational identity.

Depending on the role of the CA, the certificate can attest to the authority of the bearer to conduct e-business over the Internet. In a sense, a digital certificate performs a similar role to a driver's license or a medical diploma. It certifies that the bearer of the corresponding private key has authority to conduct certain e-business activities.

A certificate contains information about the entity it certifies, whether person, machine, or computer program. It includes the certified public key of that entity.

digital certification. *See* certification.

digital signature. A coded message added to a document or data that guarantees the identity of the sender.

A digital signature can provide a greater level of security than a physical signature. The reason for this is that a digital signature is not an encrypted name or series of simple identification codes. Instead, it is an encrypted summary of the message that is being signed. Thus, affixing a digital signature to a message provides solid identification of the sender. (Only the sender's key can create the signature.) It also fixes the content of the message that is being signed (the encrypted message summary must match the message content or the signature is not valid). Thus, a digital signature cannot be copied from one message and applied to another because the summary, or hash, would not match. Any alterations to the signed message would also invalidate the signature.

digital signature algorithm. A public key algorithm that is used as part of the Digital Signature Standard. It cannot be used for encryption, only for digital signatures.

Directory. A hierarchical structure intended as a global repository for information related to communications (such as e-mail or cryptographic exchanges). The Directory stores specific items that are essential to the PKI structure, including public keys, certificates, and certificate revocation lists.

Data in the Directory is organized hierarchically in the form of a tree, with the root at the top of the tree. Often, higher level organizations represent individual countries, governments, or companies. Users and devices are typically represented as leaves of each tree. These users, organizations, localities, countries, and devices each have their own entry. Each entry consists of typed attributes. These provide information about the object that the entry represents.

Each entry in the Directory is bound with an associated distinguished name (DN). This is unique when the entry includes an attribute that is known to be unique to the real world object. Consider the following example DN. In it, the country (C) is US, the organization (O) is IBM, the organizational unit (OU) is Trust, and the common name (CN) is CA1.

```
C=US/O=vnet/OU=Trust/CN=CA1
```

Directory server. The IBM SecureWay Directory. This Directory supports LDAP standards and uses DB2 as its base.

Distinguished Encoding Rules (DER). DER selects just one type of encoding from those that the encoding rules allow, eliminating all of the sender's options.

distinguished name (DN). The unique name of a data entry that is stored in the Directory. The DN uniquely identifies the position of an entry in the hierarchical structure of the Directory.

DN. Distinguished name.

document encrypting key. Typically, a symmetric encryption/decryption key, such as DES.

domain. *See* security domain *and* registration domain.

E

e-business. Business transactions over networks and through computers. It includes buying and selling goods and services. It also includes transferring funds through digital communications.

e-commerce. Business-to-business transactions. It includes buying and selling goods and services (with customers, suppliers, vendors, and others) on the Internet. It is a primary element of e-business.

end-entity. The subject of a certificate that is not a CA.

encrypt. To scramble information so that only someone who has the appropriate decryption code can obtain the original information through decryption.

encryption/decryption. Using the public key of the intended recipient to encipher data for that person, who then uses the private key of the pair to decipher the data.

extranet. A derivative of the Internet that uses similar technology. Companies are beginning to apply Web publishing, electronic commerce, message transmission, and groupware to multiple communities of customers, partners, and internal staff.

F

File Transfer Protocol (FTP). An Internet client/server protocol for use in transferring files between computers.

firewall. A gateway between networks that restricts the flow of information between networks. Typically, the purpose of a firewall is to protect internal networks from unauthorized use from the outside.

FTP. File Transfer Protocol.

G

gateway. A functional unit that allows incompatible networks or applications to communicate with each other.

H

HTML. Hypertext Markup Language.

HTTP. Hypertext Transaction Protocol.

HTTP server. A server that handles Web-based communications with browsers and other programs in a network.

hypertext. Text that contains words, phrases, or graphics that the reader can click with the mouse to retrieve and display another document. These words, phrases, or graphics are known as hyperlinks. Retrieving them is known as linking to them.

Hypertext Markup Language (HTML). A markup language for coding Web pages. It is based on SGML.

Hypertext Transaction Protocol (HTTP). An Internet client/server protocol for transferring hypertext files across the Web.

I

ICL. Issued certificate list.

instance. In DB2, an instance is a logical database management environment for storing data and running applications. It allows definition of a common set of configuration parameters for multiple databases.

integrity. A system protects the integrity of data if it prevents unauthorized modification (as opposed to protecting the confidentiality of data, which prevents unauthorized disclosure).

integrity checking. The checking of audit records that result from transactions with external components.

internal structure. *See* schema.

Internet. A worldwide collection of networks that provide electronic connection between computers. This enables them to communicate with each other via software devices such as electronic mail or Web browsers. For example, some universities are on a network that in turn links with other similar networks to form the Internet.

intranet. A network within an enterprise that usually resides behind firewalls. It is a derivative of the Internet and uses similar technology. Technically, intranet is a mere extension of the Internet. HTML and HTTP are some of the commonalties.

IPSec. An Internet Protocol Security standard, developed by the IETF. IPSec is a network layer protocol, designed to provide cryptographic security services that flexibly support combinations of authentication, integrity, access control, and confidentiality. Because of its strong authentication features, it has been adopted by many VPN product vendors as the protocol for establishing secure point-to-point connections over the Internet.

issued certificate list (ICL). A complete list of the certificates that have been issued and their current status. Certificates are indexed by serial number and state. This list is maintained by the CA and stored in the CA database.

J

Java. A set of network-aware, non-platform-specific computer technologies developed by Sun Microsystems, Incorporated. The Java environment consists of the Java OS, the virtual machines for various platforms, the object-oriented Java programming language, and several class libraries.

Java applet. *See* applet. *Contrast with* Java application.

Java application. A standalone program that is written in the Java language. It runs outside the context of a Web browser.

Java language. A programming language, developed by Sun Microsystems, designed specifically for use in applet and agent applications.

Java Virtual Machine (JVM). The part of the Java run-time environment responsible for interpreting bytecodes.

K

key. A quantity used in cryptography to encipher or decipher information.

key pair. Corresponding keys that are used in asymmetric cryptography. One key is used to encrypt and the other to decrypt.

L

LDAP. Lightweight Directory Access Protocol.

Lightweight Directory Access Protocol (LDAP). A protocol used to access the Directory.

M

MIME (Multipurpose Internet Mail Extensions). A freely available set of specifications that allows the interchange of text in languages with different character sets. it also allows multimedia e-mail among many different computer systems that use Internet mail standards. For example, the e-mail messages may contain character sets other than US-ASCII, enriched text, images, and sounds.

N

National Language Support (NLS). Support within a product for differences in locales, including language, currency, date and time format, and numeric presentation.

National Security Agency (NSA). The official security body of the U.S. government.

NLS. National language support.

non-repudiation. The use of a digital private key to prevent the signer of a document from falsely denying having signed it.

O

object. In object-oriented design or programming, an abstraction encapsulating data and the operations associated with that data. *See also* class.

object type. The kind of object that can be stored in the IBM SecureWay Directory. For example, an organization, meeting room, device, person, program, or process.

P

PEM. Privacy-enhanced mail.

PKCS. Public Key Cryptography Standards.

PKCS#10. *See* Public Key Cryptography Standards.

PKCS#12. *See* Public Key Cryptography Standards.

PKI. Public key infrastructure.

PKIX. An X.509v3-based PKI.

PKIX certificate management protocol (CMP). A protocol that enables connections with PKIX-compliant applications. PKIX CMP uses TCP/IP as its primary transport mechanism, but an abstraction layer over sockets exists. This enables support for additional polling transports.

PKIX listener. The public HTTP server that a particular registration domain uses to listen for requests from the Trust Authority client application.

plaintext. Unencrypted data. *Synonym for* cleartext.

preregistration. In IBM SecureWay Trust Authority, a process that allows one user, typically an administrator, to enroll other users. If the request is approved, the Registration Authority (RA) provides information that allows the user to obtain the certificate at a later time using the Trust Authority client application.

privacy. Protection from disclosing unauthorized data.

privacy-enhanced mail (PEM). The Internet privacy-enhanced mail standard, that the Internet Architect Board (IAB) adopted to provide secure electronic mail over the Internet. The PEM protocols provide for encryption, authentication, message integrity, and key management.

private key. The key in a public/private key pair that is available only to its owner. It enables the owner to receive a private transaction or make a digital signature.

Data signed with a private key can be verified only with the corresponding public key. *Contrast with* public key. *See also* public/private key pair.

protocol. An agreed-on convention for inter-computer communication.

proxy server. An intermediary between the computer that is requesting access (computer A) and the computer that is being accessed (computer B). Thus, if an end user makes a request for a resource from computer A, this request is directed to a proxy server. The proxy server makes the request, gets the response from computer B, and then forwards the response to the end user. Proxy servers are useful for accessing World Wide Web resources from inside a firewall.

public key. The key in a public/private key pair that is made available to others. It enables them to direct a transaction to the owner of the key or verify a digital signature. Data encrypted with the public key can be decrypted only with the corresponding private key. *Contrast with* private key. *See also* public/private key pair.

Public Key Cryptography Standards (PKCS). Informal inter-vendor standards developed in 1991 by RSA Laboratories with representatives from various computer vendors. These standards cover RSA encryption, the Diffie-Hellman agreement, password-based encryption, extended-certificate syntax, cryptographic message syntax, private-key information syntax, and certification syntax.

- PKCS#10 specifies a standard syntax for certification requests.
- PKCS#12 specifies a portable format for storing or transporting a user's private keys, certificates, miscellaneous secrets, and so forth.

public key infrastructure (PKI). A standard for security software that is based on public key cryptography. The PKI is a system of digital certificates, certificate authorities, registration authorities, certificate management services, and distributed directory services. It is used to verify the identity and authority of each party involved in any transaction over the Internet. These transactions might involve operations where identity verification is required. For example, they might confirm the origin of proposal bids, authors of e-mail messages, or financial transactions.

The PKI achieves this by making the public encryption keys and certificates of users available for authentication by a valid individual or organization. It provides online directories that contain the public encryption keys and certificates that are used in verifying digital certificates, credentials, and digital signatures.

The PKI provides a means for swift and efficient responses to verification queries and requests for public encryption keys. It also identifies potential security threats to the system and maintains resources to deal with security breaches. Lastly, the PKI provides a digital timestamping service for important business transactions.

public/private key pair. A public/private key pair is part of the concept of key pair cryptography (introduced in 1976 by Diffie and Hellman to solve the key management problem). In their concept, each person obtains a pair of keys, one called the public key and the other called the private key. Each person's public key is made public while the private key is kept secret. The sender and receiver do not need to share secret information: all communications involve only public keys, and no private key is ever transmitted or shared. It is no longer necessary to trust some communications channel to be secure against eavesdropping or betrayal. The only requirement is that public keys must be associated with their users in a trusted (authenticated) manner (for instance, in a trusted directory). Anyone can send a confidential message by using public information. However, the message can be decrypted only with a private key, which is in the sole possession of the intended recipient. Furthermore, key pair cryptography can be used not only for privacy (encryption), but also for authentication (digital signatures).

R

RA. Registration authority.

RC2. A variable key-size block cipher, designed by Ron Rivest for RSA Data Security. *RC* stands for *Ron's Code* or *Rivest's Cipher*. It is faster than DES and is designed as a drop-in replacement for DES. It can be made more secure or less secure against exhaustive key search than DES by using appropriate key sizes. It has a block size of 64 bits and is about two to three times faster than DES in software. RC2 can be used in the same modes as DES.

An agreement between the Software Publishers Association (SPA) and the United States government gives RC2 special status. This makes the export approval process simpler and quicker than the usual cryptographic export process. However, to qualify for quick export approval a product must limit the RC2 key size to 40 bits with some exceptions. An additional string can be used to thwart attackers who try to pre-compute a large look-up table of possible encryptions.

RC4. A variable key-size block cipher, designed by Ron Rivest for RSA Data Security. *RC* stands for *Ron's Code* or *Rivest's Cipher*. It is similar to RC2, but it has a block size of 128 bits.

registration authority (RA). IBM SecureWay Trust Authority software that administers digital certificates to ensure that an organization's business policies are applied from the initial receipt of an enrollment request through certificate revocation.

registration database. Contains information about certificate requests and issued certificates. The database stores enrollment data and all changes to the certificate data throughout its life cycle.

registration domain. A set of resources, policies, and configuration options related to specific certificate registration processes. The domain name is a subset of the URL that is used to run the registration application.

repudiate. To reject as untrue; for example, to deny that you sent a specific message or submitted a specific request.

request ID. A 24- to 32-character ASCII value that uniquely identifies a certificate request to the RA. This value can be used on the certificate request transaction to retrieve the status of the request or the certificate that is associated with it.

S

schema. As relates to the IBM SecureWay Directory, the internal structure that defines the relationships between different object types.

Secure Sockets Layer (SSL). An IETF standard communications protocol with built-in security services that are as transparent as possible to the end user. It provides a digitally secure communications channel.

An SSL-capable server usually accepts SSL connection requests on a different port than requests for standard HTTP requests. SSL creates a session during which the exchange signals to set up communications between two modems need to occur only once. After that, communication is encrypted. Message integrity checking continues until the SSL session expires.

security domain. A group (a company, work group or team, educational or governmental) whose certificates have been certified by the same CA. Users with certificates that are signed by a CA can trust the identity of another user that has a certificate signed by the same CA.

server. (1) In a network, a data station that provides functions to other stations; for example, a file server. (2) In TCP/IP, a system in a network that handles the requests of a system at another site, called a client/server.

server certificate. A digital certificate, issued by a CA to enable a Web server to conduct SSL-based transactions. When a browser connects to the server by using the SSL protocol, the server sends the browser its public key. This enables authentication of the identity of the server. It also enables encrypted information to be sent to the server. *See also* CA certificate, digital certificate, *and* browser certificate.

servlet. A server-side program that gives Java-enabled servers additional functionality.

SGML. Standard Generalized Markup Language.

sign. To use your private key to generate a signature. The signature is a means of proving that you are responsible for and approve of the message you are signing.

signing and verifying. To sign is to use a private digital key to generate a signature. To verify is to use the corresponding public key to verify the signature.

Simple Mail Transfer Protocol (SMTP). A protocol that transfers electronic mail over the Internet.

site certificate. Similar to a CA certificate, but valid only for a specific Web site. *See also* CA certificate.

S/MIME. A standard that supports the signing and encryption of e-mail transmitted across the Internet. *See* MIME.

SMTP. Simple Mail Transfer Protocol.

SSL. Secure Sockets Layer.

Standard Generalized Markup Language (SGML). A standard for describing markup languages. HTML is based on SGML.

T

TCP/IP. Transmission Control Protocol/Internet Protocol.

top CA. The CA at the top of a PKI CA hierarchy.

transaction ID. An identifier provided by the RA in response to a preregistration enrollment request. It enables a user running the Trust Authority Client application to obtain the pre-approved certificate.

Transmission Control Protocol/Internet Protocol (TCP/IP). A set of communication protocols that support peer-to-peer connectivity functions for local and wide area networks.

triple DES. A symmetric algorithm that encrypts the plaintext three times. Although many ways exist to do this, the most secure form of multiple encryption is triple-DES with three distinct keys.

Trust Authority. An integrated IBM SecureWay security solution that supports the issuance, renewal, and revocation of digital certificates. These certificates can be used in a wide range of Internet applications, providing a means to authenticate users and ensure trusted communications.

trust domain. A set of entities whose certificates have been certified by the same CA.

trust model. A structuring convention that governs how certificate authorities certify other certificate authorities.

tunnel. In VPN technology, an on-demand virtual point-to-point connection made through the Internet. While connected, remote users can use the tunnel to exchange secure, encrypted, and encapsulated information with servers on the corporate private network.

type. *See* object type.

U

Unicode. A 16-bit character set that is defined by ISO 10646. The Unicode character encoding standard is an international character code for information processing. The Unicode standard encompasses the principal scripts of the world and provides the foundation for the internationalization and localization of software. All source code in the Java programming environment is written in Unicode.

Uniform Resource Indicator (URI). Absolute URL indicates a URI position in relation to both a HOST name or IP address, and a network port.

Uniform Resource Locator (URL). A scheme for addressing resources on the Internet. The URL specifies the protocol, host name or IP address. It also includes the port number, path, and resource details needed to access a resource from a particular machine.

URI. Uniform Resource Indicator.

URL. Uniform Resource Locator.

user authentication. The process of validating that the originator of a message is the identifiable and allowed owner of the message. It also validates that you are communicating with the end user or system you expected to.

UTF-8. A transformation format. It enables information processing systems that handle only 8-bit character sets to convert 16-bit Unicode to an 8-bit equivalent and back again without loss of information.

V

VPN. Virtual Private Network.

Virtual Private Network (VPN). A private data network that uses the Internet rather than telephone lines to establish remote connections. Because users access corporate network resources through an Internet Service Provider (ISP) rather than a telephone company, organizations can significantly reduce remote access costs. A VPN also enhances the security of data exchanges. In traditional firewall technology, message content can be encrypted, but the source and destination addresses are not. In VPN technology, users can establish a tunnel connection in which the entire information packet (content and header) is encrypted and encapsulated.

W

Web browser. Client software that runs on a desktop PC and enables the user to browse the World Wide Web or local HTML pages. It is a retrieval tool that provides universal access to the large collection of hypermedia material available in the Web and Internet. Some browsers can display text and graphics, and some can display only text. Most browsers can handle the major forms of Internet communication, such as FTP transactions.

Web server. A server program that responds to requests for information resources from browser programs. *See also* server.

World Wide Web (WWW). That part of the Internet where a network of connections is established between computers that contain hypermedia materials. These materials provide information and can provide links to other materials in the WWW and Internet. WWW resources are accessed through a Web browser program.

X

X.509 certificate. A widely-accepted certificate standard designed to support secure management and distribution of digitally signed certificates across secure Internet networks. The X.509 certificate defines data structures that accommodate procedures for distributing public keys that are digitally signed by trusted third parties.

X.509 Version 3 certificate. The X.509v3 certificate has extended data structures for storing and retrieving certificate application information, certificate distribution information, certificate revocation information, policy information, and digital signatures.

X.509v3 processes create time-stamped CRLs for all certificates. Each time a certificate is used, X.509v3 capabilities allow the application to check the validity of the certificate. It also allows the application to determine whether the certificate is on the CRL. X.509v3 CRLs can be constructed for a specific validity period. They can also be based on other circumstances that might invalidate a certificate. For example, if an employee leaves an organization, their certificate would be put on the CRL.

Index

Special Characters

A

B

C

D

E

F

G

H

I

N

O

P

Q

R

S

T

U

V

W

X

Y

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.