HP OpenView Select Access

For the Windows®, HP-UX®, Linux® and Solaris® Operating Systems

Software Version: 6.2

Concepts Guide

Document Release Date: September 2006
Software Release Date: September 2006

®

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

© Copyright 2004-2006 Hewlett-Packard Development Company, L.P.

Trademark Notices

HP OpenView Select Access includes software developed by third parties. The software HP OpenView Select Access uses includes:

- Software developed by the Apache Software Foundation.
- Software developed by Claymore Systems, Inc.
- Cryptographic software written by Eric Young.
- Cryptographic software developed by The Cryptix Foundation Limited.
- cURL, Copyright 2000 Daniel Stenberg.
- JavaBeans Activation Framework version 1.0.1 Sun Microsystems, Inc.
- JavaMail, version 1.2 Sun Microsystems, Inc.
- JavaService software from Alexandria Software Consulting.
- JClass LiveTable, Copyright 2002 Sitraka Inc.
- The OpenSSL Project for use in the OpenSSL Toolkit.
- Protomatter Syslog, Copyright 1998-2000 Nate Sammons.
- SoapRMI, Copyright 2001 Extreme! Lab, Indiana University.

For expanded copyright notices, see HP OpenView Select Access `<install_path>/3rd_party_license` directory.

Documentation Updates

This manual's title page contains the following identifying information:

- Software version number, which indicates the software version
- Document release date, which changes each time the document is updated
- Software release date, which indicates the release date of this version of the software

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

http://ovweb.external.hp.com/lpe/doc_serv/

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

You can visit the HP OpenView Support web site at:

www.hp.com/managementsoftware/support

HP OpenView online support provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest

- Submit and track support cases and enhancement requests

- Download software patches

- Manage support contracts

- Look up HP support contacts

- Review information about available services

- Enter into discussions with other software customers

- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels, go to:

www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

www.managementsoftware.hp.com/passport-registration.html

Contents

1 Introduction

Identity management touches upon almost every aspect of the HP Adaptive Enterprise vision, affecting access to information across hardware, software, network resources, application servers, enterprise applications, and web portals within an organization and across organizations via business-to-business transactions.

For managing an adaptive enterprise (one that can respond quickly to change) HP OpenView Select Access provides systematic and secure user access to third-party network services and the enterprise resources they deploy. Select Access provides integrated infrastructure management that drastically reduces corporate IT costs.

By using a highly scaleable and extensible architecture, Select Access integrates with most dynamic IT environments that include:

- Wireless, web, non-web and legacy applications support
- Native LDAP v3 directory serves as the repository for user, resource and policy data; it works with existing user data and directory schema
- Leading web J2EE-compliant application and portal servers
- Popular authentication schemes to allow flexibility in strength of user identification

Audience

This document is intended for anyone using or deploying Select Access either as a standalone access management tool or as part of a larger HP OpenView Identity Management solution. It helps you understand important concepts that are part of the Select Access feature set.

The Select Access Documentation Set

This manual refers to the following Select Access documents. These documents are installed with Select Access and are available in the `<install_path>/docs` folder.

- *HP OpenView Select Access 6.2 Installation Guide*, © Copyright 2000-2006 Hewlett-Packard Development Company, L.P. (`installation_guide.pdf`)
- *HP OpenView Select Access 6.2 Policy Builder Guide*, Copyright 2000-2006 Hewlett-Packard Development Company, L.P. (`policy_builder_guide.pdf`)
- *HP OpenView Select Access 6.2 Network Integration Guide*, © Copyright 2002-2006 Hewlett-Packard Development Company, L.P. (`integration_guide.pdf`)
- *HP OpenView Select Access 6.2 Concepts Guide*, © Copyright 2005 - 2006 Hewlett-Packard Development Company, L.P. (`concepts_guide.pdf`)

Integration Papers for Select Access and vendor-specific technologies are available on the product CDs in the `docs/solutions` folder.

Online help is available with both the Setup Tool and the Policy Builder components.

As part of the Select Access SDK, two other documents are also available with this product:

- *HP OpenView Select Access 6.2 Developer's Tutorial Guide*, © Copyright 2004-2006 Hewlett-Packard Development Company, L.P. (`dev_tut_guide.pdf`)

- *HP OpenView Select Access 6.2 Developer's Reference Guide*, © Copyright 2004-2006 Hewlett-Packard Development Company, L.P. (`dev_ref_guide.pdf`)

For details on how to obtain this SDK, visit HP's Partner Care site (`http://support.openview.hp.com/partner_care.jsp`).

Chapter Summary

This guide includes the chapters listed in Table 1.

Table 1   Chapters in This Guide

Chapter	Description
Chapter 2, Adaptive Access Management	This chapter describes the concept of access management and how it applies to the management of an adaptive enterprise.
Chapter 3, Related Products and Solutions	This chapter explains three related components of the HP OpenView Federated Identity Management solution: HP OpenView Select Audit, HP OpenView Select Federation and HP OpenView Select Identity.
Chapter 4, Profiles: Understanding Directory Entries and Attributes	This chapter describes LDAP elements of an identity entry. It further describes how these LDAP elements are used to identify identities to a Select Access-protected system.
Chapter 5, Enabling Single Sign-on	This chapter details how to configure and enable SSO in single and multidomain network environments. It also explains how Select Access uses nonces and cookies.
Chapter 6, Understanding Authentication	This chapter provides information on how Select Access uses password-based and certificate-based authentication.
Appendix A, Monitoring Select Access Operations	This appendix explains where alerts are configured and describes two types of alerts.

2 Adaptive Access Management

Access management is just one part of the HP OpenView Identity Management solution. Access management is the process by which you manage and adapt to the increasing number of user populations and network resources, as well as the rising complexity of corporate partnerships.

Select Access offers adaptive enterprises a simple, user-centric, policy-based access management model that is taking on an increasingly important role in today's portal-driven environments. Select Access addresses the need for an effective, centralized security management tool that secures content in a heterogeneous environment.

Chapter Overview

This chapter describes the concept of access management and how it applies to the management of an adaptive enterprise.

Topics in this chapter include:

- What is an Access Policy? on page 11
- How is Access Managed? on page 12
- How is Access Enforced? on page 12

What is an Access Policy?

Access management is defined by an access policy. An access policy is an explicit definition of whether an identity is allowed or denied access to a resource. If access is conditional, the policy then also defines the criteria for possible access. Figure 1 shows the basic components of a policy.

Access policy = Resouce definition + User/group/role definition + Allow/deny/conditional + Conditional rule
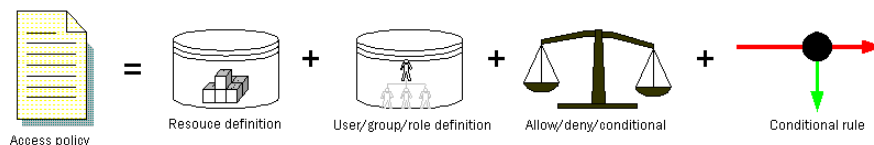
Figure 1   Basic Access Policy Structure

Policy can be manually assigned or automatically inherited across folders, groups, or even entire user locations. Inheritance is used by the Policy Builder to minimize the amount of time it takes to assign policy to thousands or millions of user and resource combinations. For more details on assigning policy, see Chapter 7, Controlling Network Access in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

Static Access Policies

A static access policy is one in which an identity is either allowed or denied access to a specific internal or external resource. For example, you might create a static access policy for an identity named John Smith where he is denied access to `ftp.mycompany.com/accounting`. You might also create a separate static access policy allowing John Smith access to `www.mycompany.com/public`.

Dynamic Access Policies

A dynamic access policy is one in which an identity is allowed access to a resource only if she meets certain criteria. These criteria are defined in rules that are created with the Rule Builder. For example, you might create a dynamic access policy for an identity named Mary Jones where she can access `ftp.mycompany.com/accounting` only if she is using 40-bit encryption. For details, see Chapter 8, Creating Conditional Access Rules with the Rule Builder, in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

How is Access Managed?

Select Access provides a Java-based user interface called the Policy Builder that simplifies the complicated and often time-consuming process of creating and applying access policies. As the administrative hub of Select Access, the Policy Builder gives administrators (full or delegated) a policy-driven approach for administering user privileges and transaction security. The interface is divided into three main areas:

- Identities Tree
- Resources Tree
- Policy Matrix

The Policy Matrix provides a grid-based representation of users and resources. With simple visual icons and inheritance rules, the Policy Builder allows you to quickly understand policy logic and thereby reduce the amount of errors typically incurred by other list-based access management systems.

For more information on using the Policy Builder, see the *HP OpenView Select Access 6.2 Policy Builder Guide*.

How is Access Enforced?

User validation and policy enforcement are the twin goals of the security model that Select Access provides. Select Access does this with two server components: the Policy Validator (the logic engine) and the Enforcer plugin (the web server agent).

▶ You can extend the Policy Validator's functionality via customized plugins that you create. For details, see the *HP OpenView Select Access 6.2 Developer's Tutorial Guide* and the *HP OpenView Select Access 6.2 Developer's Reference Guide*.

The process by which these two components evaluate a user access request is summarized below:

1 The Enforcer plugin gathers information about a user's incoming access request and sends what it discovers as an XML query to the Policy Validator.

2 When a query is received, the Policy Validator deciphers the query and determines where additional data lookups need to be performed for the user and resource pair in question.

3 The Policy Validator accesses the required policy information for the user and resource pair from the Policy Store and interprets the policy logic and conditions surrounding the request. This information is cached to improve performance of subsequent queries.

4 If it discovers a conditional policy, the Policy Validator checks and evaluates the rule criteria defined within one or more conditional rules, until the Policy Validator reaches a final decision.

5 The Policy Validator communicates the result of the policy logic to the Enforcer plugin making the request. The Enforcer plugin then enforces the policy decision.

Query and Response Structure

Select Access is the only solution in the identity management industry designed from the ground up as an XML-based extensible system.

The query

The Policy Validator receives queries from various Enforcer plugins as XML documents. The Enforcer plugin also encodes queries to the Policy Validator using XML. XML allows for complete extensibility with respect to modifying information in the query structure: you can arbitrarily add new attributes with values to a query.

► Select Access ignores any attributes for which it has no use.

As with the authorization rules, any type of data can be passed to the Policy Validator. This can include:

• Binary objects like X.509 digital certificates

• Complex objects like complete emails

• Simple objects like standard text

For example, an XML query from the Enforcer plugin on behalf of Lance Mountain is shown in the Sample Policy Validator query on page 14.

This query shows that the Enforcer plugin's **Query Details** parameter has been set to maximal. Additionally, it shows Lance's personalization details forwarded by the SAML partner, as well as a list of information that describes Steve Smith in the directory server. Notice how data is delimited by the following tag pairs:

• `<PolicyValidatorQuery>` and `</PolicyValidatorQuery>` delimit the entire query and all data contained in it.

• `<PROPERTY NAME="`*name*`"></PROPERTY>` delimit the value for the property with "*name*".

• `<PROPERTYLIST NAME="`*name*`">` delimit the value for the property list (in this case a nested property list) with *"name"*.

Sample Policy Validator query

The following XML sample is an example of an Enforcer plugin's query to a Policy Validator:

```
< PolicyValidatorQuery>
<PROPERTY NAME="service">http://mymenu.foodcompany.com:8070</PROPERTY>
  <PROPERTY NAME="path">/test/auth/submit.cgi</PROPERTY>
  <PROPERTY NAME="dstIP">10.10.10.30</PROPERTY>
  <PROPERTY NAME="srcIP">10.10.10.110</PROPERTY>
  <PROPERTY NAME="dstPort">8070</PROPERTY><PROPERTY NAME="srcPort">
  4001</PROPERTY>
<PROPERTY NAME="dstHost">mymenu.foodcompany.com</PROPERTY>
  <PROPERTY NAME="protocol">http</PROPERTY>
  <PROPERTY NAME="srcHost">user_isp.com</PROPERTY>
<PROPERTY NAME="nonce">AgAAAAAAPl5SEQAAAAA+XlRqc29sMDAxLmNhLmJhbHHR
pbW9yZS5jb206OTk5OABwYXNzAGNuPXN0ZXZlIGtvdHNvcG91bG9zLG91PXN0ZXZlLGRj
PWNhLGRjPWJhbHRpbW9yZSxkYz1jb20AAGp790LYZinTvdZ8UhpWv4CfkXtmu8885S0Ae
HA3vX7qrF353ASKBJ5RS2bJz1qCJXGfzK4vQqfVTT0mcE8yIgJQefoFX+GnVV092WaFYti
Oe7QjfpSqXtaQmShZC1QlRnAYJVwo5Gx5s4B/THU5BgPKZyvUEJnK/pcsb2hOqCOd
</PROPERTY>
    <PROPERTY NAME="fullname">Lance Mountain</PROPERTY>
    <PROPERTY NAME="arrived">10am</PROPERTY>  </PROPERTYLIST>
 <PROPERTY NAME="native_nonce">enable</PROPERTY>
  <PROPERTY NAME="http_query">hungry=yes&amp;lunch=pizza</PROPERTY>
 <PROPERTYLIST NAME="http_query_list">
    <PROPERTY NAME="hungry">yes</PROPERTY>
    <PROPERTY NAME="lunch">pizza</PROPERTY>
  </PROPERTYLIST>
<PROPERTY NAME="method">POST</PROPERTY>
<PROPERTYLIST NAME="http_header_list">
    <PROPERTY NAME="Host">dev01:8070</PROPERTY>
    <PROPERTY NAME="User-Agent">Mozilla/5.0 (Windows; U; en-US;
    rv:1.0.1)Gecko/20020823 Netscape/7.0</PROPERTY>
 <PROPERTY NAME="Accept">text/xml,application/xml,application/
xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-mng,image/
png,image/jpeg,image/gif;q=0.2,text/css,*/*;q=0.1</PROPERTY>
    <PROPERTY NAME="Accept-Language">en-us, en;q=0.50</PROPERTY>
    <PROPERTY NAME="Accept-Encoding">gzip, deflate, compress;q=0.9
    </PROPERTY>
    <PROPERTY NAME="Accept-Charset">ISO-8859-1, utf-8;q=0.66, *;q=0.66
    </PROPERTY>
    <PROPERTY NAME="Keep-Alive">300</PROPERTY>
    <PROPERTY NAME="Connection">keep-alive</PROPERTY>
```

```
        <PROPERTY NAME="Referer">http://dev01:8070/test/allow/postfix.html
</PROPERTY>

<PROPERTY NAME="Cookie">PolicyUser=AgAAAAAAPl5SEQAAAAA+XlRqc29sMDAx
LmNhLmJhdHRpbW9yZS5jb206OTk5OABwYXNzAGNuPXN0ZXZllIGtvdHNvcG91bG9zLG91PXN0
ZXZlLGRjPWNhLGRjPWJhdHRpbW9yZSxkY1jb20AAGp79OLYZinTvdZ8UhpWv4CfkXtmu
8885S0AeHA3vX7qrF353ASKBJ5RS2bJz1qCJXGfzK4vQqfVTT0mcE8yIgJQefoFX+GnVV092
WaFYtiOe7QjfpSqXtaQmShZC1QlRnAYJVwo5Gx5s4B/THU5BgPKZyvUEJnK/pcsb2hOqCOd
</PROPERTY>

    <PROPERTY NAME="Content-Type">application/x-www-form-urlencoded
    </PROPERTY>

    <PROPERTY NAME="Content-Length">54</PROPERTY>

  </PROPERTYLIST>

  <PROPERTY NAME="server">Apache/2.0.40 (Unix) DAV/2</PROPERTY>

  <PROPERTY NAME="queryID">2</PROPERTY>

</PolicyValidatorQuery>
```

The reply

Replies are also formatted as XML. They can contain three things:

- An action telling the Enforcer plugin what to do next. Typical actions are:
  — Allow or Deny
  — Get more information by displaying a form

- Data that must be communicated to an application on the web server. For example, a Java application might require information to personalize the experience for a user.

- Session information, namely a cookie or nonce that identifies the user on subsequent visits to one or more configured cookie domains. For details on cookies and nonces, see Understanding Nonces and Cookies on page 38.

For example, an XML response from the Policy Validator is shown in the Sample Policy Validator reply to the Enforcer plugin on page 15. Notice how data is delimited by the following tag pairs:

- `<PolicyValidatorReply>` and `</PolicyValidatorReply>` delimit the entire response and all data it contains.

- `<PROPERTY NAME="name"></PROPERTY>` delimit the value for the property with "*name*".

Sample Policy Validator reply to the Enforcer plugin

The following XML sample is an example of an Policy Validator's response to an Enforcer plugin's query:

```
<PolicyValidatorReply>

<PROPERTY NAME="queryID">2</PROPERTY>

  <PROPERTY NAME="authenticated_dn">cn=Lance Mountain,ou=customer,
  dc=com,dc=user_isp</PROPERTY>

  <PROPERTYLIST NAME="personalization">

    <PROPERTY NAME="User">lmountain</PROPERTY>
```

```
        <PROPERTY NAME="Phone">504%2D2325</PROPERTY>

        <PROPERTY NAME="Fax">504%2D2399</PROPERTY>

        <PROPERTY NAME="DName">cn%3Dlance%20mountain%2Cou%3Dlance%2Cdc
        %3Dcom%2Cdc%3Duser_isp%2Cdc</PROPERTY>

        <PROPERTY NAME="Groups">customer%20people</PROPERTY>

    </PROPERTYLIST>

    <PROPERTY NAME="login_time">Thu Feb 27 12:59:45 2003</PROPERTY>

    <PROPERTYLIST NAME="authentication_server_types">

      <PROPERTY NAME="authentication_method">password</PROPERTY>

    </PROPERTYLIST>

    <PROPERTY NAME="action">ALLOW</PROPERTY>

</PolicyValidatorReply>
```

What Is a Policy Validator Cache?

To enhance performance, the Policy Validator employs a cache to reduce the amount of network traffic as well as speed up response time. The Policy Validator cache contains any policy and user data that the Policy Validator has previously retrieved from the directory server:

- When the Policy Validator receives a policy query for a particular server, it typically caches that policy for future reference.

- For each authenticated user, the Policy Validator caches the user's information.

▶ Due to the fact that runtime conditions can affect the outcome of a given query, the Policy Validator does not cache responses to Enforcer plugin queries. For example, time of day changes can stop a resource from being accessible to a given user after several previous successful accesses.

A cache refresh interval is defined in the Policy Validator's configuration. The Policy Validator therefore checks the directory server and updates the information stored in the cache periodically, so it continually has new information on which to base allow or deny decisions.

⚑ The administrator can also flush the cache manually. If the cache expires or is flushed, and a new query comes in for that resource, the cache is again primed with the resource data.

3 Related Products and Solutions

HP OpenView Federated Identity Management puts control in the hands of the business user with a suite of secure, scalable applications focused on ease of use, expedited deployment, and the ability to embrace change.

Chapter Overview

This chapter explains three related components of the HP OpenView Federated Identity Management solution: HP OpenView Select Audit, HP OpenView Select Federation and HP OpenView Select Identity.

Topics in this chapter include:

- The HP OpenView Solution on page 17
- What is Federation? on page 19
- Federation Concepts on page 19
- Standards and Initiatives on page 21
- Business Benefits on page 22

The HP OpenView Solution

HP OpenView provides a holistic solution to Federated Identity Management that focuses on the way that business processes are defined, executed and managed. Such an approach led HP to Business Driven Service Oriented Identity Management, a solution that is founded on the principles of a service-oriented architecture. This allows the enterprise to manage its federated identity needs in the same way it manages its existing business processes.

Figure 2 illustrates the technological components of the HP OpenView solution: Select Access, Select Audit, Select Federation, and Select Identity.
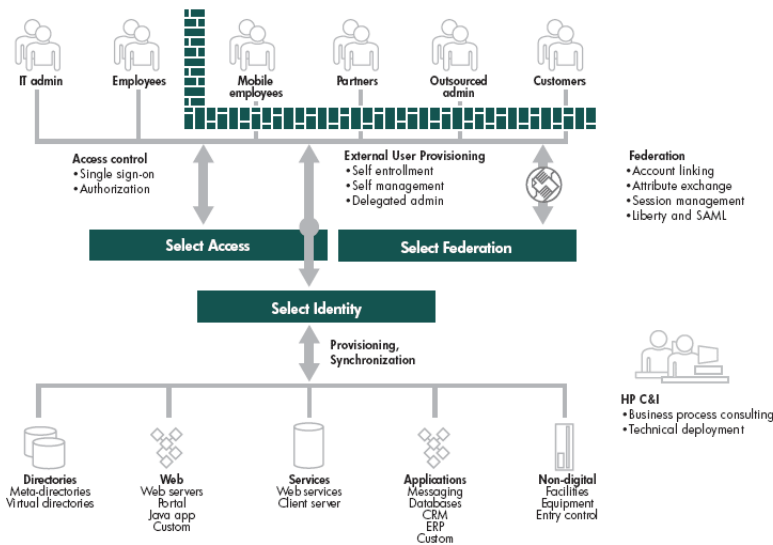
Figure 2   Business-driven Federated Identity Management

HP OpenView Select Audit

Select Audit provides a consolidated and tamper-aware identity audit trail. All HP OpenView Identity Management components report every administrative action, user change, access request and authorization decision to a centralized audit system. The entries are digitally signed to prevent the tampering of audit records. This allows the monitoring of events such as who accessed what resources, and who made what changes to the identities and their entitlements, and when.

Select Audit has a detailed reporting engine that enables organizations to define reporting procedures according to their operational and audit policies. Auditors and authorized administrators can create reports on identity and access administration and user actions that are sorted and/or filtered by audit data, including users, servers, administrators, dates, specific events, and custom data.

Select Audit outputs to multiple destinations including databases and files. It can trigger alerts based on any audit event, and can be configured with distinct alert handling instructions. Instructions can range from writing additional information to the audit records, generating email and SNMP notification, opening a trouble ticket in HP OpenView Service Desk, or reporting to your network operation centre via HP OpenView Operations.

Select Audit can consolidate all logs that contain personally identifiable information (PII) into a central storage location.

HP OpenView Select Federation

Select Federation effectively enables extranet identity management, web single sign-on, and cross-domain identity management without requiring a centralized data repository or synchronization between repositories. Using Select Federation, you easily achieve single sign-on and federated session management while leveraging your existing identity management deployments. Select Federation is designed to work with HP OpenView Identity Management solutions, as well as to integrate with other vendors' systems.

Select Federation is protocol-agnostic. It allows federation to be achieved via SAML (1.0/1.1), Liberty ID-FF (1.1/1.2), and Liberty ID-WSF (1.0) protocols. Additionally, Liberty Personal Profile, Liberty Employee Profile, and the LECP (Liberty Enabled Client Proxy) services are provided. Select Federation includes extensive administrative features that allow for management of all circle-of-trust relationships, as well as privacy preferences.

For more details on Federation, see What is Federation? on page 19.

HP OpenView Select Identity

Providing industry leading identity lifecycle management functionality, HP OpenView Select Identity provides a robust and well integrated framework based upon a pure J2EE implementation for managing your distributed identity environment.

Unlike the traditional roles/rules-based approaches to identity management, HP OpenView Select Identity is based on the concept of Service Oriented Identity Management. This vital architectural distinction allows Select Identity to mirror your IT/business services structure, tightly integrating the myriad of business processes associated with Federated Identity Management. Using the unique context-based approach, Select Identity automates change management in your business processes and needs. Its 100 percent all web-based functions and open standards-based architecture allows for the extreme delegation of administration functionality to your partners, user organizations, and customers that are part of your federated deployment.

What is Federation?

Federation is achieved with HP OpenView's Select Federation product. Traditionally, identity management was a core component of security infrastructures, where it maintained account information that allowed users to log in to a system or a limited set of applications. Recently, however, identity management evolved beyond the sole purview of information security professionals. It is now a key enabler for electronic business through a technological innovation called Federation. This evolution came about as a result of the increasing number and complexity of online distributed systems that both house and manage some portion of our identity.

Federation is the combination of business and technology practices to enable identities to span systems, networks, and domains in a secure and trustworthy fashion. This is analogous to how we use passports to assert our identity as we travel between countries.

Federation Concepts

There are several concepts related to federation that drive a federated identity solution's value when correctly implemented. This section describes the following key federation concepts:

- Single Sign-on on page 20
- Account Linking on page 20
- Attribute Sharing on page 20
- Privacy on page 21

Single Sign-on

Single sign-on (SSO) in the light of federation means that authentication can take place seamlessly across multiple, heterogenous, security domains both within and between businesses.

For details about SSO, see Chapter 5, Enabling Single Sign-on.

Account Linking

Account linking allows a user with multiple service and/or application accounts to link these accounts for future authentication and sign-in at these sites. Essentially, a mapping of distributed account information takes place. As an example (refer to Figure 3), Jane has an account (Jane123) with her credit union where she performs online banking operations. She also has an account at an online bookseller (JaneTheBookLover). In order for the online bookseller to accept authentication assertions from Jane's credit union, and for the bookseller to make payment requests to the credit union, the two organizations must map each others account identifiers (the bookseller needs to know that Jane123 is Jane TheBookLover, and vice-versa). For security and privacy reasons, most distributed account mappings are done using a pseudonym (alias) instead of the actual account name. This helps prevent collusion between multiple service providers.
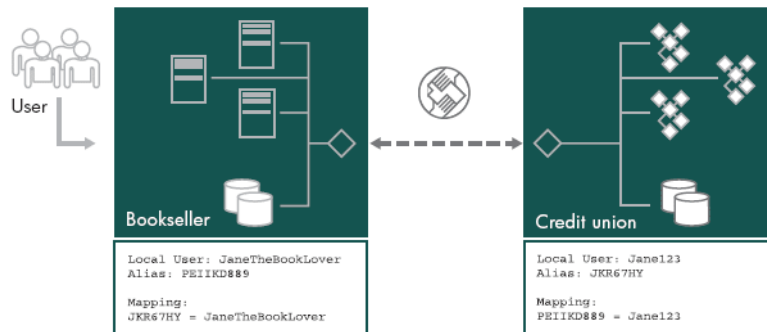

Figure 3   Account Linking

Attribute Sharing

Federation is more than just single sign-on. It also includes the secure sharing of identity attributes for the purposes of personalization, making fine-grained access control decisions or completing a transaction. Consider the earlier example of Jane's Credit Union and the online bookseller; attribute sharing could be used to allow the bookseller to retrieve your credit card number from the credit union to complete an online purchase without the need for Jane to supply it for each transaction or for the bookseller to store it locally.

Personalization allows an organization to tailor the user experience for a given individual, leading to a streamlined interface for the user and the ability to target information dissemination for a business.

Privacy

In a federated environment where accounts can be linked and attributes shared, it is vital to protect personal identifiable information. Besides employing strong security measures at the protocol and transport layers of the interactions, processes around end user consent/control, anonymity, and pseudo-anonymity need to be employed. Account linking should only occur with prior user consent. What attributes are shared, and with whom, should be based upon the policies designated by the end user. Additionally, there needs to be the capability to share pertinent attribute information in an anonymous fashion. It is not always pertinent to a service provider who the end user is, so why share that information needlessly? For example, a weather service provider only needs to know your location (your zip code); the fact that your name is John Doe need not be revealed in order to provide the local forecast.

Standards and Initiatives

Standards play an important role in federation by providing the common set of protocols, semantics, and processing rules that allow the various parties and their identity infrastructures to interoperate. Figure 4 depicts some of the relevant standards associated with federation and how they are interconnected.
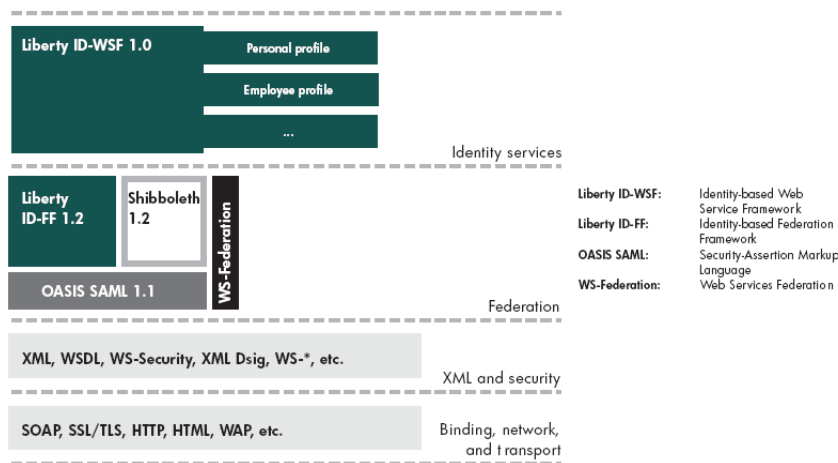

Figure 4   Federation Protocol Landscape

Looking at the federation protocol landscape diagram in Figure 4, there are several things to note.

First, there is a dependency among the layers of the protocol landscape. For example, each of the protocols in the federation layer rely on well-known and industry-tested standards at the transport, network, binding and XML layers. In many cases this dependency is mandated for interoperability, and in other cases it is optional (such as Digital Signatures and SSL).

Second, there is a dependency between some of the federation protocols themselves. This is a result of the evolutionary nature of the standards. As business needs changed or became more complex, new standards emerged to meet those needs. Those new federation standards leveraged and extended concepts of early standards. Here you can see that the Security Assertion Markup Language (SAML) created in OASIS has formed the basis for both the Liberty Identity Federation Framework (ID-FF) and Shibboleth.

The third item to note is a "new" layer in the federation stack called "Identity Services." Identity services are distributed network services that act on the behalf of an identity, with prior consent and according to policy, to share specific identity information or attributes to a relying party so that intelligent decisions can be made. This enables such things as personalization of services, and intelligent transactions based on identity information.

Business Benefits

An enterprise can realize numerous business benefits directly associated with federation. A solid federated identity management infrastructure can bring substantial cost savings, operational efficiencies, revenue growth through development of strategic offerings, and increased security and risk management.

The benefits of deploying a federated identity management infrastructure are summarized below:

- Regulatory compliance: Provides key capabilities for achieving and maintaining compliance with privacy and information integrity regulations.

- Business agility and reach: Enables the business to more quickly respond to change. Promotes extension of information resources to external organizations. Systems can be deployed quickly and easily since the components of the solution are based on commonly accepted standards and interfaces, eliminating the need to develop to a myriad of integration points.

- Cost reduction and revenue enhancement: Provides multiple, measurable avenues to cost reduction. Decreases time to productivity for employees, partners, and customers. Enables the business infrastructure required for the development of bundled customer offerings with strategic partners.

- User self sufficiency: Enables users to manage their own identities. Provides the tools for business managers to manage the access needs of their employees, partners, and customers.

- IT efficiency: Enables IT to do more with less by reducing user administration burden and deflecting calls to the help desk. Enables integration of legacy systems without re-engineering their authentication and authorization modules.

- Increased security: Provides systematic and auditable control over user accounts, entitlements, and access rights. Provides context-sensitive, gradient levels of authentication and risk management.

- Risk management: Decentralizes data management. Identity information is not centrally aggregated in a federated solution, but rather managed in place by the data owners. This decentralization of data management lessens the risk for any one enterprise by delegating a portion of the risk to partners that manage that distributed information.

- User experience and productivity: Provides a unified method for users to request and self-manage their accounts, access rights and passwords. Provides for a seamless experience through single sign-on and personalization.

4 Profiles: Understanding Directory Entries and Attributes

This chapter describes LDAP elements of an identity entry. It further describes how these LDAP elements are used to identify identities to a Select Access-protected system.

What is an Identity Profile?

Identities (as well as groups, dynamic groups, and folders) are stored as directory entries on a directory server. Entries are used to manage and describe the individual or even the entity they represent. An identity's entry consists of three main elements:

- The *object class* of the entry
- The *name* of the entry
- The *attributes* of the entry. Of all directory entry components, attributes are frequently used by important Select Access features like dynamic groups, personalization, and profile self-management.

About Object Classes and Entry Names

Object classes and names help to identify the entry.

- An object class: Defines the entry's required and optional attributes. For details, see About Object Classes on page 21.
- Entry names: Are a systematic way of creating unique IDs to avoid having two entries belonging to two separate entities. This is done by two names: a distinguished name (DN) and a relative distinguished name (RDN). For details, see About Directory Entry Names on page 22.

About Object Classes

A directory entry must belong to one or more object classes. An object class defines all the common attribute types a directory entry can have. For example, user entries typically belong to the `organizationalPerson` object class. This object class specifies that an entry must contain a CN attribute (common name) and a SN attribute (surname). This means that the entry must contain values for these attributes or the entry cannot be created in the directory server.

User, group and folder entries each belong to one or more standard LDAP object classes. The dynamic group's object class is not a standard LDAP object class. You can use the Policy Builder to view an entry's object class structure but you cannot change the object classes defined for an entry. For details, see To view the object class on page 29.

About Directory Entry Names

A directory entry is identified by:

- A relative distinguished name (RDN), or entry name, that refers to the actual user, group, dynamic group, or folder the entry is describing. The RDN is a component of a DN. For details, see Relative Distinguished Name (RDN) or Entry Name on page 22.

- A distinguished name (DN) that uniquely identifies the entry in the directory server. For details, see Distinguished Name (DN) on page 22 and DN Qualifiers on page 24.

The entry's distinguished name (DN) is created by adding the entry's RDN to the parent DN names. For example, if an identity's parent entries are:

```
ou=People,o=mycompany.com
```

and the identity's RDN is:

```
CN=Brian Jones
```

then the identity's DN is:

```
CN=Brian Jones,ou=People,o=mycompany.com
```

You can use the Policy Builder to view an entry's names. For details, see To view an entry's DN and RDN on page 28.

Relative Distinguished Name (RDN) or Entry Name

A directory entry's relative distinguished name (RDN) refers to the actual user, group, dynamic group or folder the entry is describing. One of the entry's attribute type/value pairs is the entry's RDN. When you create a directory entry using the Policy Builder, the attributes described in the table below are used.

Table 2   Attributes Used for Entry RDNs

For this User Tree entry...	The RDN is...
User	`CN` (common name) attribute is the alternate
Group	`CN` (group name) attribute
Dynamic Group	`nxRole` (dynamic group name) attribute
Folder	`ou` (folder name) attribute

In the Policy Builder, the Identities Tree lists entries according to their RDN. The value of an entry's RDN is commonly referred to as the entry name.

Distinguished Name (DN)

A directory entry's distinguished name (DN) is the unique name used to identify the entry in the directory server. An entry's DN consists of a series of attribute type/value pairs that trace the entry's path up the directory tree. The entry's RDN is shown in Figure 1.
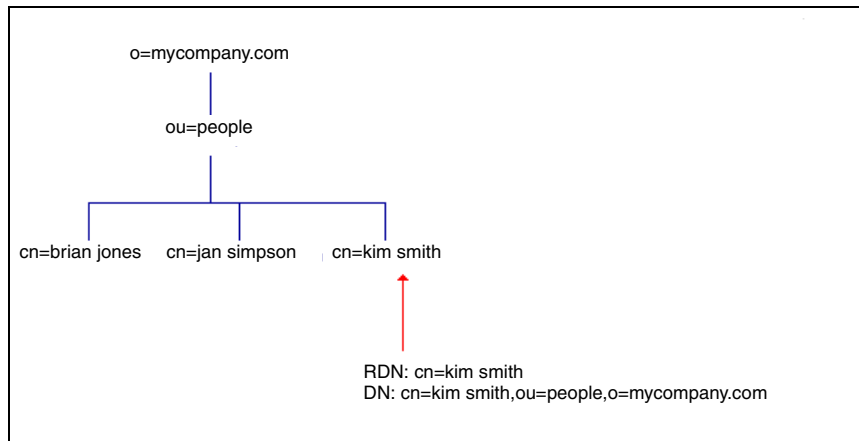
Figure 5   Entry RDN

Each entry's DN must be unique in the directory server. This means you cannot create two
identities with the same RDN value in the same folder. For example, the People folder cannot
contain two different identities with the RDN attribute `cn=brian jones`.

However, depending on how your directory server is configured, it is possible for two entries to
have the same RDN value, if they belong to different branches in the directory tree. This
means the following entries are valid because each entry has a different DN, as shown in
Figure 2.

➤ These entries are not valid in all directory servers. For example, if a directory server requires
that all user entries contain a unique value for the `cn` attribute, then two different identities
cannot have the same common name, even if they belong to different branches in the tree.

➤ Some directory servers do not support all characters, which makes them invalid when used in
the Policy Builder. Invalid characters can cause unpredictable behavior by your directory
server. A list of invalid characters and a list of corresponding directory servers is provided in
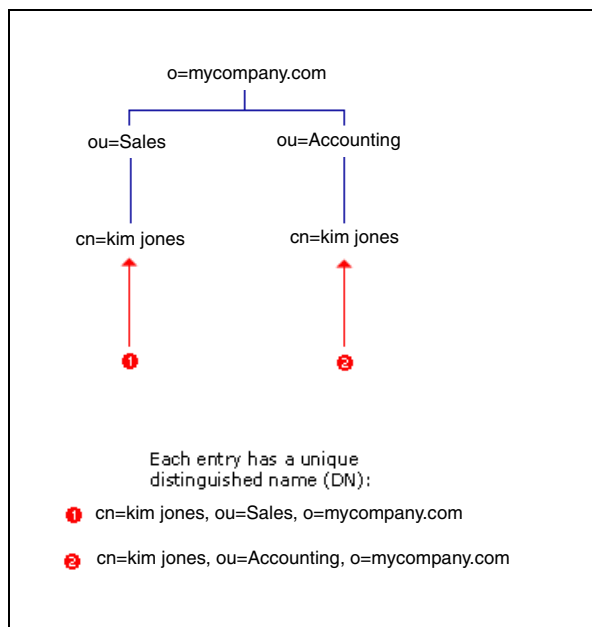Appendix A, Invalid Characters in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

Figure 6   Entry DN

DN qualifiers

When two or more identities share the same CN, assign a DN qualifier for those entries. This allows you to distinguish each of these identities, based on some other identifier. For example, if you have three identities with a CN of John Smith belonging to the same OU, you can use a DN qualifier of Junior to identify them based on seniority level.

About Directory Attributes

A directory entry contains attributes that describe the item. For example, a directory entry for an identity contains attributes such as the identity's name, ID, email address, and phone number. These attributes can be used to assign dynamic group membership, or activated to evaluate whether or not to deploy one personalized page or another. For more details on attributes, see To enable Select Auth in the *HP OpenView Select Access 6.2 Policy Builder Guide*. The table below outlines the three ways you use attributes.

Table 3   Attribute Purpose

Purpose	Details
Assign dynamic group membership.	Creating and Modifying a Dynamic Group in the *HP OpenView Select Access 6.2 Policy Builder Guide*
Deploy personalized network resources.	To enable personalization in the *HP OpenView Select Access 6.2 Policy Builder Guide*
Define what directory entry data an identity can self-manage via a profile.	The Profile Self-Management Terminal Point in the *HP OpenView Select Access 6.2 Policy Builder Guide*

Directory Attribute Structure

An attribute has a type and a value. The attribute type determines the type of information the attribute can contain. For example, in the attribute phone=555-1212, the attribute type (phone) specifies that the attribute value (555-1212) must be a telephone number. You can also specify multiple values for an attribute. For example, the directory entry for an identity named Kim Smith contains the attributes illustrated below.

```
cn=Kim Smith

sn=Smith

uid=ksmith

phone=555-1212

mail=ksmith@mycompany.com

mail=ksmith@abc.com
```

In the example, notice that the `mail` attribute has two different values (mail=ksmith@mycompany.com and mail=ksmith@abc.com), one for each of Kim's email addresses.

Creating a List of Preferred Attributes

Select Access allows you to select the attributes to do the following:

- Determine how resources are to be deployed for personalization. Policy Validator evaluates which network resource to display, depending on the attributes you select.

- Determine what attributes identities can self-manage in their identity profile.

- Test access entitlement for an identity and/or groups and dynamic groups the identity is a member of. For details, see The Directory Attributes Decision Point in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

⚠ Microsoft has placed a size limit on LDAP records for Active Directory servers. This limitation creates further issues with the activation of *all* available attributes. Because Microsoft will not be changing this size limitation, you cannot activate all available attributes. However, this is typically not a problem since most deployments do not require all user attributes activated.

🚩 Activate at least one attribute before you can use the Attribute Logic decision point in Rule Builder.

▶ For details on how to create your list of preferred attributes, see To select the preferred identity attributes in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

Configuring Advanced Identity Profile Properties

There are advanced properties you can view or configure for user, group, or folder entries, as outlined in Table 3.

Table 4   Advanced Properties Overview

Advanced Property	Details
Attributes	Viewing Attributes on page 26
Distinguished Name	To view an entry's DN and RDN on page 28
Object classes	To view the object class on page 29
Operational attributes	To view operational attributes on page 30

▶ These settings do not apply to dynamic groups.

Viewing Attributes

The **Attributes** tab shows all of the attributes for the identity, group, or folder. You can add, modify, and delete attributes.

To view an entry's attributes

1 Select an identity, group, or folder.

2 Click **Edit** → **Properties**.

3 Click **Advanced**.

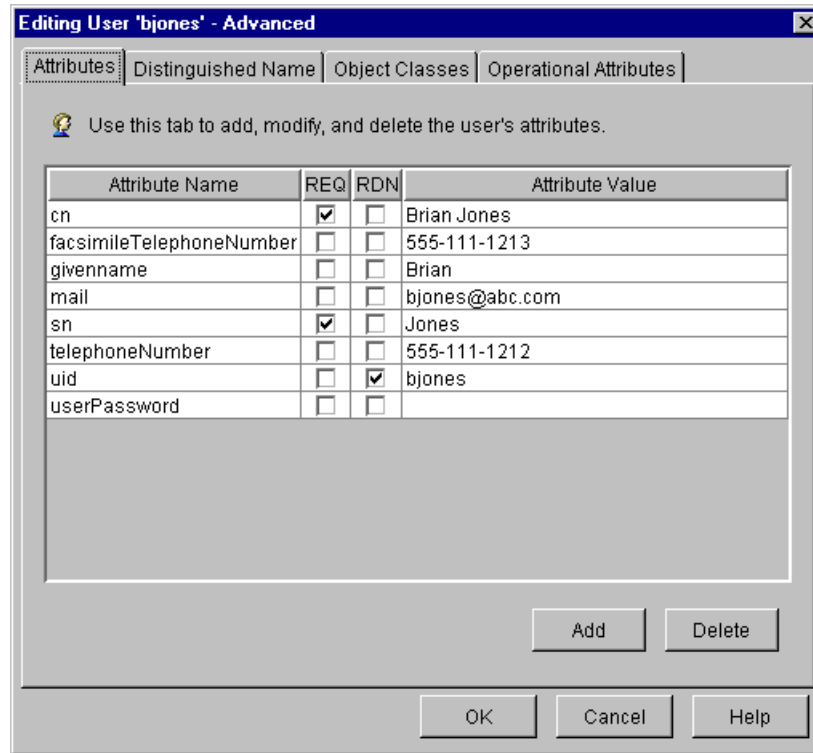4 Select the **Attributes** tab as shown in Figure 3.

Figure 7   Editing User Advanced Dialog Box

Required attributes are identified with a checkmark in the **Req** column. Required attributes are determined by the object class structure for the identity, group, or folder and cannot be changed.

5 The RDN attribute for the identity, group, or folder is shown in the **RDN** column. The value of this attribute is used as the directory entry name and is shown on the Identities Tree. You cannot change which attribute is used as the RDN attribute. You can do any of the tasks outlined in Table 4:

Table 5Attribute Overview

Option	Details
Add an attribute.	To add an attribute on page 27
Enter multiple values for an attribute.	To enter multiple values for an attribute on page 27
Change an attribute's value.	To change an attribute's value on page 28
Delete an attribute.	To delete an attribute on page 28

To add an attribute

You can store additional information for the identity, group, or folder by adding attributes.

1 Click **Add**. The **Add Attribute** dialog box appears, and lists the attributes available for the identity, group, or folder's object class, as shown in Figure 4.
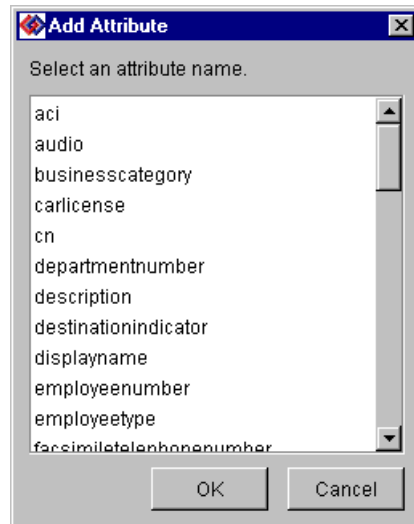

Figure 8   Add Attribute Dialog Box

2 Select an attribute and click **OK**. A new row is added and the attribute's name is entered automatically in the **Attribute Name** field.

3 In the **Attribute Value** field, enter a value for the attribute and click **Enter**.

To enter multiple values for an attribute

You can add several instances of the attribute and specify a different value for each instance, as shown in Figure 5. For example, you can record two different email addresses for an identity.
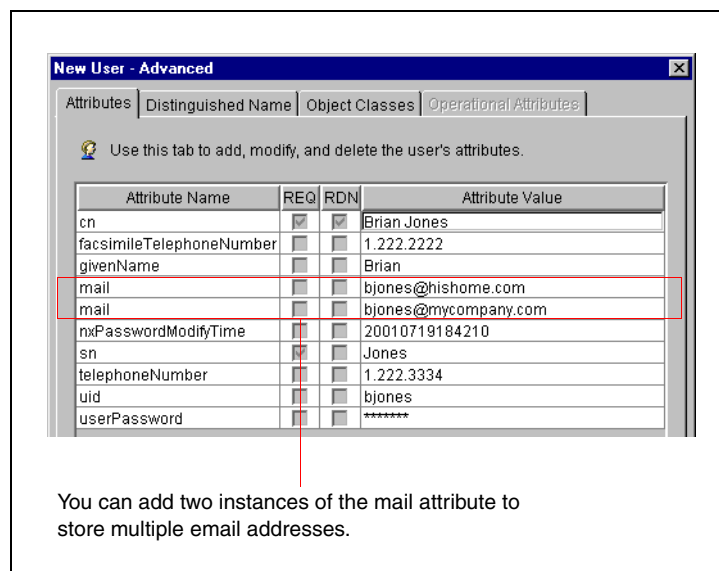

You can add two instances of the mail attribute to store multiple email addresses.

Figure 9   New Identity Advanced Dialog Box

To change an attribute's value

1   Click the **Attribute Value** field.

2   Type a new value and then click **Enter**.

To delete an attribute

Select the attribute and then click **Delete**.

➤   If you delete an attribute that also appears on the **Identity Information**, **Folder Information**, or **Group Information** tab, the attribute's value is deleted, but the attribute is not removed from the tab.

To view an entry's DN and RDN

1   Select an identity, group, or folder.

2   Click **Edit → Properties**.

3   Click **Advanced**. Select the **Distinguished Name** tab, as shown in Figure 6.

```
Editing User 'bjones' - Advanced                                    [X]

  Attributes | Distinguished Name | Object Classes | Operational Attributes

        This tab shows the name and location of the user's directory entry.

    Entry Name (RDN):        uid=bjones

    Location (Parent DNs):   ou=People,o= baltimore.com




                                        OK      Cancel      Help
```
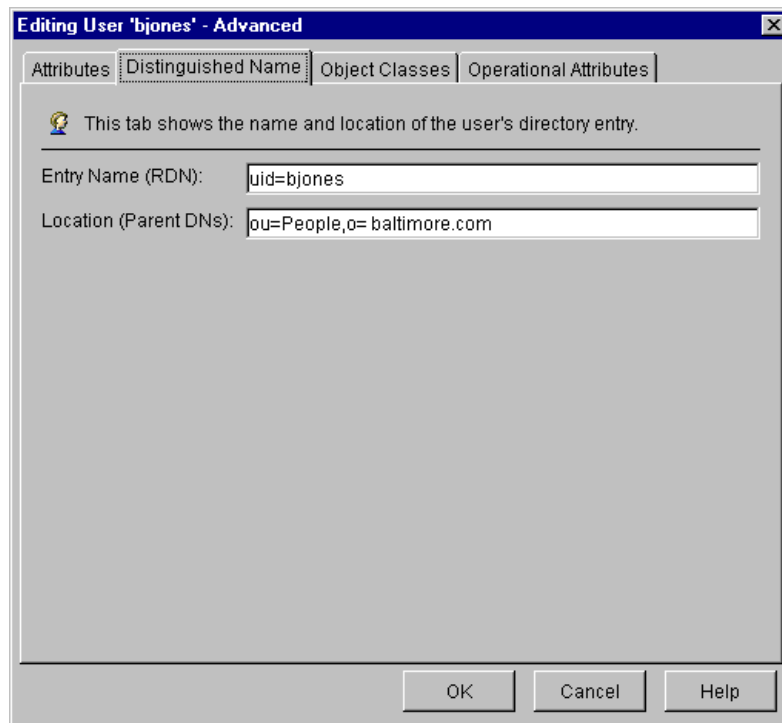
Figure 10 Editing User Advanced Dialog Box

• The **Entry Name (RDN)** field shows the entry's relative distinguished name (RDN).

• The **Location (Parent DNs)** field shows the entry's parent entries in the directory tree.

➤   Some directory servers do not support all characters, which makes them invalid when used in the Policy Builder. Invalid characters can cause unpredictable behavior by your directory server. A list of invalid characters and a list of corresponding directory servers is provided in Appendix A, Invalid Characters, in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

➤   You cannot change a DN once it has been saved to the directory server.

To view the object class

1 Select an identity, group, or folder.

2 Click **Edit → Properties**.

3 Click **Advanced**.

4 Select the **Object Classes** tab. The object class structure for the identity, group, or folder is shown. The object classes are listed in alphabetical order, as shown in Figure 7.
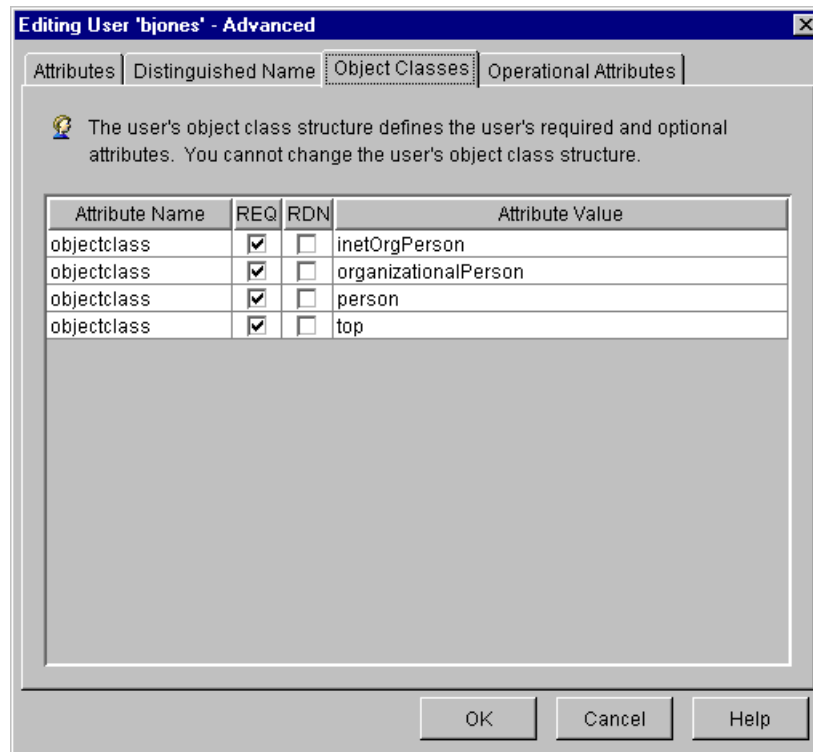

Figure 11 Editing User Advanced Dialog Box

To view operational attributes

1 Select the identity, group, or folder.

2 Click **Edit → Properties**.

3 Click **Advanced**.

4 Select the **Operational Attributes** tab. The operational attributes for the identity, group, or folder appear. The operational attributes for a directory entry describe when the entry was created and modified. You cannot change operational attributes.
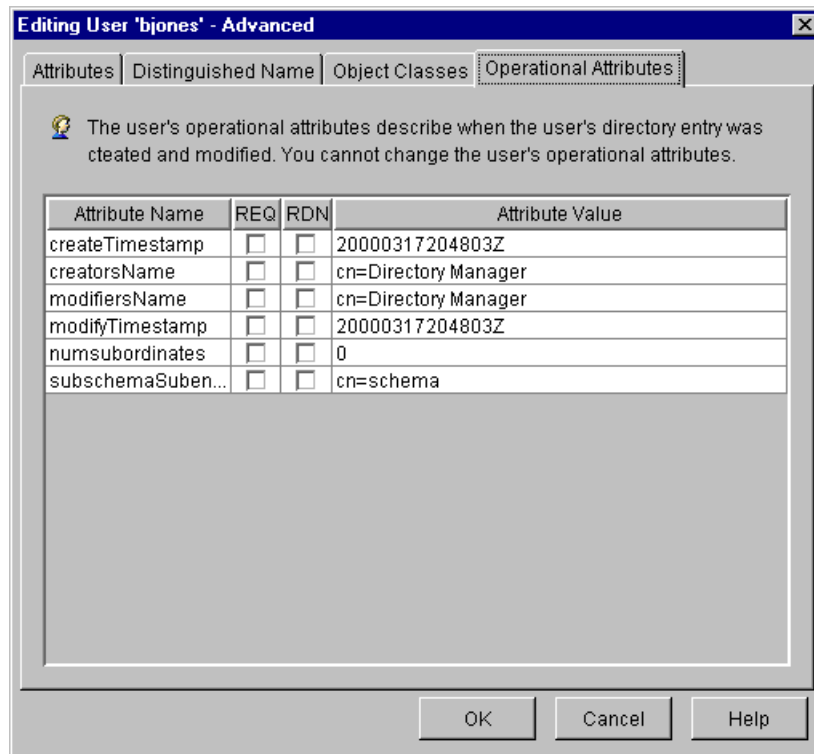
Figure 12 Editing User Advanced Dialog Box

5 Enabling Single Sign-on

Single sign-on (SSO) is a key technology required for distributed computing. For example, a single user name and password, once authenticated by the Policy Validator, acts like a passport, giving identities access to distributed web content, groupware, workflow or client and server applications. It allows the identity to seamlessly move from one site to another without requiring reauthentication with each new site.

▶ SSO requires that the web server's authentication methods and server names match. This is necessary because when the Policy Validator generates a cookie for the authenticated identity, it includes a delimited list of authentication service names. As the identity moves from one server to another, the authentication service name and the authentication method for that service must be of the same level. If not, SSO fails and the identity is required to reauthenticate with the new web server. For details on authentication services, see Chapter 7, Setting Up Authentication Services, in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

Select Access supports two different levels of SSO, each of which requires a different configuration to enable it on your Select Access-protected network:

- Single-domain SSO: A single-domain network environment is one where multiple web servers with unique resources exist on a single cookie domain. For details, see Configuring SSO on Single Internet Domains on page 34.

- Multiple domain SSO: A multidomain network environment is one where multiple web servers with unique resources exist on more than one cookie domain. For details, see Configuring SSO on Multiple Internet Domains on page 35.

Chapter Overview

This chapter details how to configure and enable SSO in single and multidomain network environments. It also explains how Select Access uses nonces and cookies.

Topics in this chapter include:

- Configuring SSO on Single Internet Domains on page 34
- Configuring SSO on Multiple Internet Domains on page 35
- Understanding Nonces and Cookies on page 38
- How Select Access Uses Nonces and Cookies to Authenticate Identities on page 40
- How Select Access Uses Nonces and Cookies in Session Management on page 40

Configuring SSO on Single Internet Domains

You can set up multiple Enforcer-protected web servers on a single cookie domain, so that identities can access resources on them, without being required to reauthenticate each time or to provide additional identity information.

For example, if an identity is authenticated as JDoe at `www.mycompany.com` (Server1), and a hyperlink sends her to `extranet.mycompany.com` (Server2), she is never challenged by any of the Enforcer-protected web servers because they share the same cookie domain. Figure 13 on page 34 illustrates this sample scenario.
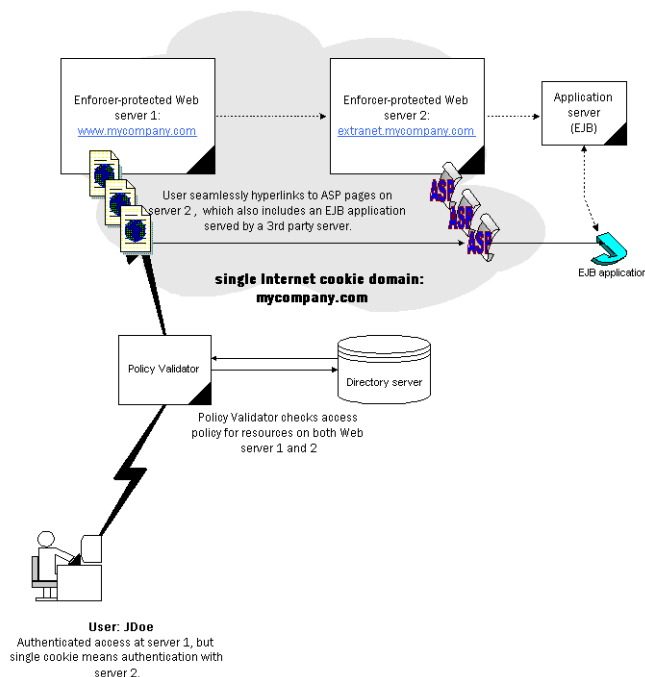
Figure 13 Single-domain SSO Scenario

To configure SSO on a single domain only

1 When configuring your Enforcer plugin, perform a **Custom** configuration.

2 On the Enforcer plugin's **Single DNS domain SSO** setup screen, enter the cookie domain.

3 Repeat these steps on all Enforcer-protected web servers.

For details on configuring the Enforcer plugin, see Chapter 8, Configuring the Enforcer Plugins in the *HP OpenView Select Access 6.2 Installation Guide*.

Configuring SSO on Multiple Internet Domains

Using multidomain SSO is valuable for corporate networks with many web servers with multiple domains because it makes authentication consistent and streamlined. multidomain SSO provides the following benefits:

- IDs and passwords across all domains are synchronized through Select Access-protected networks.

- Fewer ID and password pairs are needed.

- ID and password pairs are centrally located.

A typical multidomain network environment can be an expensive proposition in terms of the administration effort that needs to be maintained as an identity shifts across different domains. Credentials do not just need to be entered; they also need to be remembered. For each new set, there is an added security risk because identities have the tendency to:

- Keep passwords the same so they are easy to remember.

  OR

- Make passwords unique but record them in an unsecured way.

To alleviate the symptoms of administration overhead and security risk, you can set up multiple Enforcer-protected web servers on multiple cookie domains, so that identities can access resources on them, without being required to reauthenticate each time or to provide additional identity information.

For example, consider this sample scenario:

- An identity is authenticated as JDoe at `www.mycommercesite.com` (Server1).

- A hyperlink sends her to `coolproduct.affiliateco.com` (Server2) before stopping at `canada.purchasemenow.com` (Server3).

- She is never challenged by any of the Enforcer-protected web servers on these three domains.

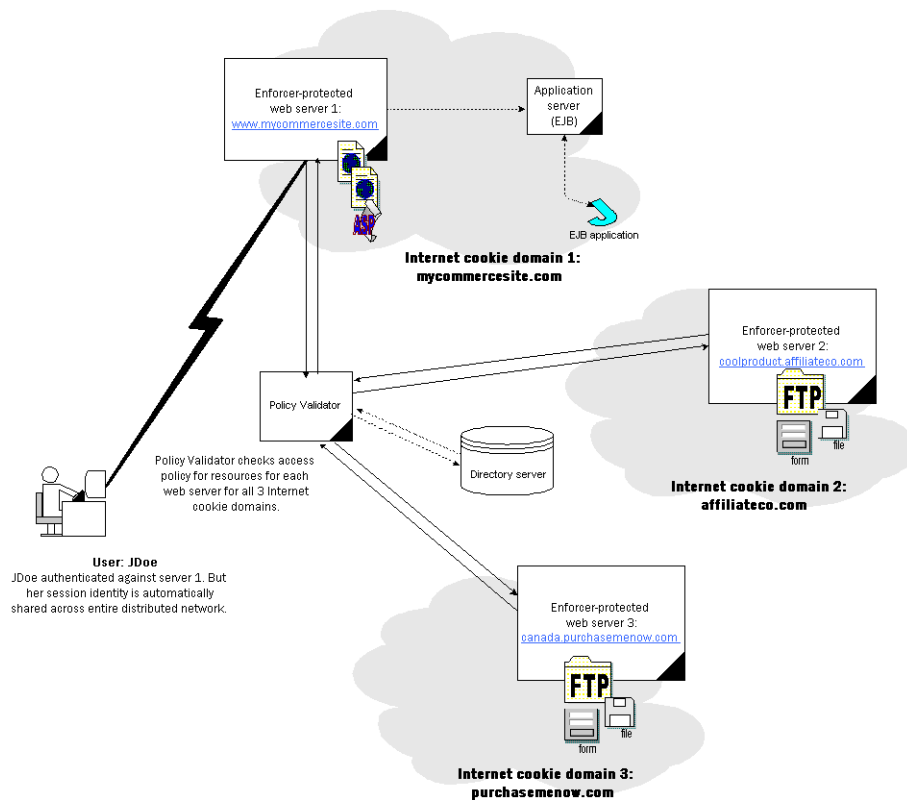Figure 14 on page 36 illustrates this sample scenario.

Figure 14 Multidomain SSO Scenario

How the Enforcer Plugin Supports Multidomain SSO Redirects

With multidomain SSO, the identity's browser is the central processing point for all communications. Cookies are set and redirect requests are distributed to and from web servers.

1   If an identity is denied access on a web server, and if the previous web server is included in the protected domains list (for details, see Setting Up Multidomain SSO with Select Access on page 36), the identity needs to be reauthenticated with a special URL that is redirected back to the originating server.

2   The previous web server recognizes this URL and redirects the browser to the new web server with the cookie that authenticates the identity on the second server.

Setting Up Multidomain SSO with Select Access

To ensure that an identity accessing content on Enforcer-protected domains is not unnecessarily rechallenged for information, refer to the steps in Table 6.

⚠   Due to the way Internet Explorer uses the Referer headers when linking to a non-protected (HTTP) page from a protected (HTTPS) page, multidomain SSO does not get triggered. For details, see Appendix E, Troubleshooting in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

Table 6 **Multidomain SSO Overview**

Step	For details, see...
1 Run the Setup Tool for each Enforcer plugin and create a matching list of domains so that they all share a mutually inclusive list of protected domains that allows analogous multidomain SSO: a When configuring your Enforcer plugin, perform a **Custom** configuration. b On the Enforcer plugin's **Multiple DNS domain SSO** setup screen, enter the cookie domain. c Repeat these steps on all Enforcer-protected web servers.	Setting up Multiple Domain Single Sign-on in the *HP OpenView Select Access 6.2 Installation Guide*
2 Register all Policy Validators with the same directory server. This is typically the same directory server that the Administration server uses as its Policy Store.	How rogue Policy Validators are detected on page 37
3 Ensure distributed web servers use the same authentication service with the same authentication method.	Configuring authentication methods on page 37

How rogue Policy Validators are detected

All Policy Validators obtain policy information from a common LDAP directory. This allows them to validate the nonces other Policy Validators generate. Because the nonce encodes identity-specific information, it must be decoded by a Policy Validator that is able to look up data on the same directory. The Policy Validator looks for two things:

• Identity-specific information to validate identity data. The identity data does not need to be on the same directory server as the Policy Store.

• Policy Validator-specific information. This confirms whether the nonce-generating Policy Validator is a rogue Validator or not. Only Policy Validators registered in the Policy Store are considered valid Validators. Policy Validators are automatically registered when you install and configure them with the Setup Tool.

If a Policy Validator cannot validate a cookie, the authentication fails and the user is forced to re-authenticate.

Configuring authentication methods

If you are using multidomain SSO, set up the same authentication methods *and* authentication services for resources in the Policy Matrix. This is important because the cookie contains information on the authentication service that validated the identity. If the server information in that cookie does not match the authentication service used for the new web site the identity is requesting, the identity is forced to reauthenticate. For example, if an identity is authenticated with password authentication on PassServ1, but moves to another part of your distributed network that uses password authentication on PassServ4, the identity is required to reauthenticate, despite your efforts of setting up multidomain SSO.

However, there may be times when you want to tier the level of authentication used. The more confidential the distributed content is, the stronger the authentication you likely want to use. This method of setting up authentication across a distributed network is called "step-up authentication," and is something you may want to consider for your network.

▶ For details on configuring your pool of authentication methods and servers, see Chapter 6, Setting Up Authentication Services in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

Understanding Nonces and Cookies

Since HTTP is a stateless protocol (that is, each transaction is completely separate from all others), HTTP cookies were devised as a method of preserving state between HTTP requests. This allows you to keep track of shopping baskets, viewing preferences, and so on.

⚠ Do not try to decode the contents of Select Access's nonce to extract identity information. This nonce is an internal data format that is subject to change without notice. If you want to extract identity information for personalization or customization purposes, you can use personalization attributes instead. For details, see Chapter 6, Implementing Select Access Personalization With Your Web Server in the *HP OpenView Select Access 6.2 Network Integration Guide*.

Identifying Identities Without Reauthentication

Nonces and cookies are used by Select Access to help identify identities as they navigate through one or more web servers. While both elements contain identity-specific information, only the cookie is set on the identity's web browser so the identity can be identified without reauthenticating.

Nonces: Issued by the Policy Validator and sent to the Enforcer plugin

A nonce is an opaque piece of data. Select Access's version of a nonce is typically translated into browser cookies. A nonce can simply be a large random number, or it can contain encrypted or digitally signed data, such as an expiry date. Select Access nonces can contain the following types of encrypted data:

- The identity's DN
- The name of the Policy Validator that issued the nonce
- The ID of the authentication services used to authenticate the identity
- The time of identity login
- The expiry date of the nonce
- A digital signature that ensures the nonce has not been tampered with

Cookies: Issued by the Enforcer plugin and set in the browser

Cookies are the HTTP-specific way of sending the nonce that the Policy Validator issues. The Enforcer plugin takes the nonce and packages it with additional information before allowing the web server to set a cookie in the identity's web browser. The browser then sends this cookie in its HTTP headers (see the Sample HTTP reply header) for each new content request the browser makes on behalf of the identity.

Sample HTTP reply header

The following code is a sample of an HTTP reply header:

```
Set-Cookie: PolicyUser=bWFub3dhci5jYS5iYWx0aW1vcmUuY29tOjk5ODh8dWlkPXN

rLG91PXVzZXJzLG91PXN0ZXZlLG89Y2EuYmFsdGltb3JlLmNvbXxmfHBhc3NfdmFsaWRhdG9

yfDtM3z4FzLNlwtEpKyGGtyl2k7OehNCdCQf2IcW+GV9gdn5n1jFYPsE21h/
nCuHtt38n6sXk1lh

BGv7t6qI18Rf13hnMg1Fq0qCZjUMO6rXzfESetBO6NL1zF/
wCIaM0r7oilummeK6gYVWukFwz3W+t2OYn

NnJk5nPzPPozI49RRvaNDA/; path=/; domain=.mycompany.com;
```

An identity's browser returns a similar string to the web server each time the identity tries to open a new session on that server, as illustrated in the Sample browser new session attempt below.

Sample browser new session attempt

The following code is a sample of a browser's new session attempt:

```
PolicyUser=Cookie:bWFub3dhci5jYS5iYWx0aW1vcmUuY29tOjk5ODh8dWlkPXNr

LG91PXVzZXJzLG91PXN0ZXZlLG89Y2EuYmFsdGltb3JlLmNvbXxmfHBhc3NfdmFsaWRhd

G9yfDtM3z4FzLNlwtEpKyGGtyl2k7OehNCdCQf2IcW+GV9gdn5n1jFYPsE21h/
nCuHtt38n6sXk1lhBGv7t6qI18

Rf13hnMg1Fq0qCZjUMO6rXzfESetBO6NL1zF/wCIaM0r7oilummeK6gYVWukFwz3W

+t2OYnNnJk5nPzPPozI49RRvaNDA==
```

This string allows the Policy Validator to identify the identities. This identity can then be used to present customized content for that identity, if the administrator activates personalization attributes that correspond to the identity's attributes.

▶ For details on how to use activated attributes to customize content, see Chapter 3, Building your Identities and Resources Trees in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

In most cases, the cookie prevents identities from having to reauthenticate each time they request a new web page on that server. However, depending on the type of cookie identities have, they might be required to reauthenticate between sessions:

- Single-session nonces help the Policy Validator keep track of an identity during a single session.

- Inter-session nonces are persistent and have an expiry date, after which time the Policy Validator again requires identities to authenticate themselves.

How Select Access Uses Nonces and Cookies to Authenticate Identities

Select Access uses nonces and cookies to authenticate returning identities through a simple process:

1   The Policy Validator includes a nonce in the XML reply it sends to the Enforcer plugin.

2   The Enforcer plugin sets the value of the nonce as a cookie and sends it to the HTTP client application.

3   The client application passes the cookie back to the web server in later requests in order to inform the Policy Validator that the identity has already been authenticated.

4   The Policy Validator may generate a new nonce that includes a new timestamp, and the cycle begins again.

5   If the identity ever fails to make a new request within the time limit encoded in the nonce's timestamp, the Policy Validator acts as it did on the first request; that is, it ignores the nonce and takes the unauthenticated path through a conditional access rule if one exists.

6   If the result of evaluating the conditional rule is deny, the reply sent to the client application may include information indicating that access might have been permitted had authentication taken place.

> Nonces are ignored during evaluation of conditional rules that do not contain an authentication decision point and do not use Select Auth.

How Select Access Uses Nonces and Cookies in Session Management

Session management involves administrators being able to force identities to explicitly log out of web sites. For security purposes, it is desirable to allow identities to explicitly log out so subsequent identities of the shared machine cannot take advantage of the session cookie. For details, see Forcing Logout on page 41. It works as follows:

1   When identities log into a web site protected by Select Access, they are issued an HTTP cookie representing their logon session. This cookie typically expires after ten minutes of idle time, a value you can configure for the Policy Validator. For details, see Chapter 7, Configuring the Policy Validator in the *HP OpenView Select Access 6.2 Installation Guide*.

2   Identities can be logged out in the following two ways:

• When the cookie expires, the identity must log on again if he wishes to access controlled content. The lifetime of the Select Access session cookie also ends when the identity completely closes the web browser.

• Policy Validator can explicitly log out identities by invalidating their session cookie. To log out identities, Policy Validator replaces an existing valid HTTP session cookie with an invalid cookie. Invalid cookies cause future connection attempts to fail during the authentication. When Select Access receives an invalid cookie from the web browser, it forces identities to log in again in order to access controlled resources.

Forcing Logout

By manipulating the way cookies are created and used, administrators can force identities to explicitly log out from your web site.

For some applications, particularly those where identities are using shared machines, it is desirable to allow identities to explicitly log out so that subsequent identities of the shared machine cannot take advantage of the session cookie. There are two ways to avoid this:

- Request that the identity close all browser windows on the machine, however, this is difficult to enforce.

- Log out the identity with an invalidated session cookie. To log out identities, Policy Validator replaces an existing valid HTTP session cookie with an invalid cookie. Invalid cookies cause future connection attempts to fail during the authentication. When Select Access receives an invalid cookie from the web browser, it forces identities to log in again in order to see controlled resources.

To use the logout feature with your web content

1   Create a new rule. For details, see Chapter 8, Creating Conditional Access Rules with the Rule Builder in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

2   Add a **Logout** terminal point.

3   Save this rule.

4   Select the identity and network resource pair and apply the corresponding conditional access control rule to it in the Policy Matrix.

5   Any time an identity follows a link to this page, the Select Access logout rule is invoked and the identity's cookie is replaced with the invalid cookie.

> We recommend that you make the logout URL a separate page, rather than display it within a frame. The reason for this is that some web browsers make parallel requests for the contents of each frame. If two or more of those frames contain URLs protected by Select Access, it is remotely possible that an updated cookie is sent in response to one URL while the logout cookie is sent in response to the other. In this case, which cookie the web server stores cannot be predicted. This problem does not occur if the logout URL is not included in a frameset.

6 Understanding Authentication

Select Access supports many authentication mechanisms. This chapter documents two of the more popular mechanisms:

- Password-based authentication: Requires that identities include a password as part of their login credentials.

- Certificate-based authentication: Forces identities and components to authenticate themselves with a trusted certificate. Select Access creates a more trusted environment.

➤ Processing a certificate query can be very time-consuming. It is an operation that involves multiple processes. Occasionally a certificate query for a transient identity might take priority over another query. If another query is interrupted by a certificate query, an informational message is logged that says: `short-circuiting`. Do not be alarmed by this message; it means that the Policy Validator is not rerunning the complete certificate verification process.

Chapter Overview

This chapter provides information on how Select Access uses password-based and certificate-based authentication.

Topics in this chapter include:

- Password Authentication Differences Among Directory Servers on page 43
- Authenticating Components with Certificates on page 44
- Authenticating Identities with Certificates on page 46
- Understanding Cryptographic Algorithms Used by Certificates on page 48

Password Authentication Differences Among Directory Servers

Select Access takes advantage of two main authentication routines:

- Validator-based authentication: This is the authentication routine of choice because it is faster than LDAP-based authentication. This authentication routine allows Select Access to read the password directly, thereby allowing the Policy Validator to perform the authentication.

- **LDAP-based authentication:** This is used as an alternative when a directory server does not support Validator-based authentication. This authentication routine lets Select Access bind as the identity before being allowed to read the password. In this case, authentication occurs outside the Policy Validator.

⚠ Certain directory servers do not allow Select Access to read the identity's password. When this happens, the password seemingly appears undefined, despite being present in the directory server.

Table 7 summarizes the subtleties among supported directory servers. Review this table and ensure you understand the implication of these differences.

Table 7   Password Authentication Summary

Directory Server	Authentication Routine	Read Password?	Set and Manage Passwords?
Sun ONE 5.1	Validator	Yes	Yes
Netscape 6.01	Validator	Yes	Yes
Novell eDirectory 8.5.1	LDAP	No	Yes
Critical Path Directory 4.0	LDAP	No	Yes
Microsoft Active Directory 5.0	LDAP	No	Via SSL only [a]
Oracle Internet Directory 3.0.1.1	Validator	Yes	Yes
eTrust Directory Server	Validator	Yes	Yes

a. See the *HP OpenView Select Access 6.2 Release Notes* for details.

Authenticating Components with Certificates

Select Access components communicate over TCP/IP, a protocol that enables two hosts to establish a connection and exchange streams of data. Because a single machine can host multiple instances of Select Access components, it is important that connections between Select Access components be protected.

If your default directory server (the one you define at the start of the Administration server's setup) runs over SSL, Select Access must be configured to use certificates. There are two ways you can choose to set up certificate-based authentication among components to ensure data streams are well protected:

- **Let Select Access handle certificates:** Select Access can distribute, manage, and verify components' certificates required for the negotiations of SSL connections. In this instance, configuration of certificate-based authentication and data encryption requires no intervention and additional setup is not required.

- **Import your own CA certificate:** You can import your own external CA certificate. The Administration server uses this certificate to authenticate and negotiate SSL connections. In this instance, intervention is required to configure authentication and data encryption, causing the setup to become somewhat more complex.

The Administration Server and Certificate Management

SSL certificates for your Select Access components are managed by the Administration server. As part of its duties, the Administration server is entirely responsible for the management of the CA certificate as well as the distribution of newly signed certificates to Select Access components.

There are two sides to certificate management:

- How the Administration server manages the certificates it creates
- How Select Access components validate certificates

These topics are described in the sections that follow.

How the Administration server manages certificates

Immediately after its installation and configuration, the Administration server waits for subsequent certificate requests from the InstallAnywhere wizard. This wizard acts on behalf of the administrator (that is, the installer uses the administrator's credentials). It verifies the request before creating a signed certificate using a CA certificate stored on the directory server (which you may or may not have imported), and registers the component that has been issued a signed certificate, thereby building a list of trusted components.

Carefully maintain this list of trusted components so that the Administration server can effectively manage certificates of those components it believes to be trusted. That means that if there is a perceived security transgression of any kind, especially one involving certificates or keys, you need to uninstall the affected component and remove it from any component list it belongs to (for example, the list of available Policy Validators used by Enforcer plugins).

When trust is breached

When you need to address a possible security transgression involving one or more components, you would likely want to remove a trusted component.

Regenerating SSL certificates

Regenerating your SSL certificates synchronizes them despite any change in your deployment. Regenerate certificates anytime there are concerns as to whether or not a change in your system would affect the way certificates are processed among all Select Access components. Important deployment changes that require SSL certificate regeneration include:

- When you remove a component whose trust has been breached. You cannot just stop a Policy Validator, or copy or delete files required by an Enforcer plugin. If you do so, the list of trusted components is not adequately maintained by the Administration server, which can degrade the security of the Select Access system.

  To update the component list (stored on the directory server or IDs used by other components):

  a   Uninstall the offending component from the host machine with the installer.

  b   Reinstall the component in question. This creates a new ID for it as well as regenerates its SSL client certificate.

- If you move or reinstall the Administration server. You can only have one Administration server on your Select Access-protected network. If you reinstall the Administration server elsewhere, regenerate certificates for all the Select Access components on your network.

- If your CA changes. In this case, you would need to regenerate certificates for all Select Access components.

How Select Access components validate certificates

The SSL system that Select Access employs creates three levels of validation.

1 Components check that the certificate presented is a valid certificate, because it comes from a trusted source.

2 The certificate confirms that the requesting component is what it claims to be. All certificates place the component's hostname or IP address into the certificate.

> ⚐ HP recommends that you use a host name in the certificate rather than an IP address as hostnames are less likely to change. If you change a component's IP address, the certificate for that component is no longer valid. You need to rerun the installer on the host machine so that you can regenerate a certificate to that machine that matches its changed IP address.

3 The component is allowed to connect and open an SSL-encrypted session. Using the component's ID in the certificate, the name or IP address is compared against the list of registered components.

This method of mutual authentication and subsequent data encryption decreases the likelihood of unauthorized access to the data stream as well as the data source.

Authenticating Identities with Certificates

There are three categories of identities Select Access can authenticate:

- The administrator who signs policy data to prevent indirect tampering with it. The data signer's certificate is configured with the Administration server. You can either let Select Access handle the management of this certificate, or import your own certificate.

- The local or remote administrator who runs the Policy Builder applet over HTTPS. The administrator's certificate is configured with the Administration server. You can either let Select Access handle the management of this certificate, or import your own certificate.

- Known and unknown identities who need to be authenticated via a certificate authentication service. Once they have been authenticated, identities can open an encrypted SSL connection using the HTTPS protocol.

This section describes the last kind of identity authentication.

How Policy Validator Performs Identity Lookups

The Policy Validator analyzes the identity's certificate to determine its validity using three steps, as shown in Table 8.

Table 8   Policy Validator Lookup Process

Lookup Step	Details
1 Find the identity to see that the individual making the request can be authenticated.	Finding the identity on page 47
2 Verify that the certificate the identity is presenting is valid.	Verifying the validity of the certificate on page 48
3 Create a transient identity profile so the Policy Validator can authenticate future access requests.	Creating a transient identity on page 48

Only after these steps take place does the policy for that identity get checked and a final allow or deny decision is made. If the Policy Validator authorizes access to an HTTPS resource, the identity's certificate is checked and the session is then encrypted with the algorithm in this certificate. Otherwise, if the resource is an HTTP resource, the certificate is ignored.

Finding the identity

When trying to locate the identity, the Policy Validator extracts information from the certificate and tries matching profiles against it. The procedure below summarizes this process:

1   The identity presents a certificate to the Policy Validator, which then validates it via the certificate server. Often, the `SubjectDN` of the certificate is similar to that of the profile on the LDAP directory server.

   ▶   If you have configured **Advanced Certificate** properties, then certificate CN may be mapped to another identity attribute. If this is the case, then the lookup behavior uses the attribute you have mapped to the CN instead, rather than the `SubjectDN`. For details on how you configure **Advanced Certificate** properties, see To configure advanced SecurID properties in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

2   The Policy Validator checks its cache to see if a `SubjectDN` exists from a previous lookup.

3   If no cached information matches, the Policy Validator looks for a matching `SubjectDN` on the directory server. One of two methods for locating an identity profile is used:

   —   If the certificate's `UID` is the identity's `RDN` (assuming all `UID`s are unique), the Policy Validator extracts the `UID` and looks in LDAP for the identity's authorization policy.

   —   If not, the Policy Validator does a `CN` search after it extracts the `CN` for the `SubjectDN`. The Policy Validator only checks the policy if the profile contains a compatible certificate.

The first match is automatically returned. For example, assume you have the following LDAP entries used as unique profiles for different identities:

```
CN=Jane Smith, OU=Sales, O=mycompany.com, C=CA
```

```
CN=Jane Smith, OU=Development, O=mycompany.com, C=UK
```

If the Policy Validator starts looking for "CN=Jane Smith", it finds both profiles. However, the Policy Validator does not know which one matches the certificate, so it looks for that certificate in the profile on the directory server.

4 If no match occurs, then a transient profile is created, and a profile is added to the Identities Tree.

> Create this transient identity location before configuring the certificate server.

Verifying the validity of the certificate

Verifying the validity of a certificate is a two-step process that checks to see if:

1 The identity's certificate is a compatible certificate.

2 The certificate is valid and has not been invalidated. The Policy Validator does this by looking up the certificate's revocation status with:

- Certificate Revocation Lists (CRL)
- Online Certificate Status Provider (OCSP)

Creating a transient identity

The Policy Validator creates a transient identity when:

- The identity profile is not in LDAP, but in the Certificate server's database.
- The certificate provided is a valid certificate.

This allows the Policy Validator to look up the identity's policy on the directory server, even though no profile exists for that identity. It does this by concatenating the certificate's UID or CN, and "synthesizes" identity profiles by permanently caching data. For details, see Validating Identities When Profiles Are Not on the Directory Server in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

Understanding Cryptographic Algorithms Used by Certificates

Cryptography uses algorithms to accomplish the following:

- Keep communications private
- Protect sensitive information
- Identify and authenticate the parties involved in the communications

Although several algorithms exist, Select Access uses the ones that are fast, easy to use, and have good security properties. Select Access uses these specific algorithms for the following functions:

- Nonces
- Certain aspects of SSL
- Certain aspects of policy signing

Encryption Methods

The following encryption methods are the most common, and are often used with an authentication method such as digital signatures to vouch for a message's integrity.

- Symmetric key cryptography (also known as private key cryptography). The algorithms in this category have the following characteristics:
  - They use only one key for both encryption and decryption.
  - The security of this type of algorithm rests in the key; therefore it must remain secret or *private*.
- Asymmetric key cryptography (also known as public key cryptography). The algorithms in this category have the following characteristics:
  - The key used to encrypt information is different from the key used to decrypt information.
  - The key used to encrypt, called the *public* key, can be made public without compromising the message's security.
  - The key used to decrypt, called the *private* key, is held by a specific person.
- Cryptographic message digest. This type of algorithm allows you to output a hash or digest that you compare to a digest generated locally to check, for example, that your email or certificate has not been tampered with.

Table 9 provides an overview to how some common algorithms can be used by Select Access.

Table 9   Common Algorithms Used by Select Access

Algorithm	How Its Used
MD5 (also known as Digest or Message Digest)	Used to create secure credentials such as cookies and nonces. For details, see Understanding Nonces and Cookies on page 38. Caution! Using this algorithm generates the cookie at a faster rate than other algorithms, but anyone with access to the `validatorData` object in the directory server can forge cookies.
RSA (Rivest, Shamir, and Adleman)	Used for both encryption (opening SSL sessions) and authentication. For example, if you select it during the Policy Validator's configuration, you can use it to create secure credentials such as cookies and nonces. Note: With this algorithm, you get the highest level of security, but the cookie is generated and verified at a slower rate than with other algorithms.
SHA-1 (Secure Hash Algorithm)	Used to create digital signatures to secure policy data because it produces a 160-bit hash value.

Table 9   Common Algorithms Used by Select Access (cont'd)

Algorithm	How Its Used
3-DES (Triple Data Encryption Standard)	If you import a CA certificate that uses this symmetric algorithm, you can use it to open SSL sessions.
AES (Advanced Encryption Standard)	If you import a CA certificate that uses this symmetric algorithm, you can use it to open SSL sessions.
RC4 (Rivest Cipher)	If you import a CA certificate that uses this symmetric algorithm, you can use it to open SSL sessions. This algorithm is three times faster than DES'.

A Monitoring Select Access Operations

Select Access allows you to configure two kinds of alerts that assist in monitoring significant incidents involving identities and components. This appendix explains where alerts are configured and describes two types of alerts.

Topics in this appendix include:

- Configuring Alerts on page 51
- Alert Notification Decision Point on page 51

Configuring Alerts

Alerting is a feature that notifies an administrator via email of significant incidents involving identities and components. Alerts can be configured in the Rule Builder's Alert Notification decision point. This component sends an alert notification when an identity reaches a decision point in a rule. The alert notification goes to wherever the audit policy defines such an alert should go. For example, an administrator could insert an Alert Notification decision point in a rule that he applies against each of his company's confidential resources, such as payroll information, employee records, and prospective new customers. For details, see Alert Notification Decision Point on page 51.

While the Alert Notification decision point involves monitoring identities' movements, event notification via email involves monitoring components' activities.

Alert Notification Decision Point

The Alert Notification decision point triggers a log message (which can be routed using the Policy Validator's audit settings) when an identity passes or fails a specific criteria in the rule. The alert can be configured to notify an administrator immediately of an incident, regardless of whether it involves identities or components.

The Alert Notification decision point has the following benefits:

- The administrator gets notified immediately when an identity tries to access a resource on which the alert decision point is applied.
- You can filter messages based on their level of severity.

For example, if you want to be alerted each time an identity tries to access the company payroll information, even though she is allowed access, you would create a rule that includes this decision point. For details, see Chapter 8, Creating Conditional Access Rules with the Rule Builder in the *HP OpenView Select Access 6.2 Policy Builder Guide*.

Event Notification Via Email

An email alert is a message sent to addresses you specify, to notify the addressees of a specific (usually severe) message that has been triggered. For example, you can configure email alerts so that an administrator is immediately notified when a Policy Validator fails.

Event notification via email has the following benefits:

- The administrator gets notified immediately when a component experiences a problem.
- You can configure alerts to be sent to several people.
- The alerts can be collected into a report using Select Audit.
- You can find out exactly which component sent the alert.

For details, see the *HP OpenView Select Audit 1.0 Administration Guide*.

Index