Sun Java™ System

Access Manager 6 Technical Overview

2005Q1

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817-7643-10

Contents

Preface 7

Who Should Use This Book 7
Before You Read This Book 8
Conventions Used in This Book 8
Typographic Conventions 8
Symbols 9
Default Paths and File Names 9
Shell Prompts 10
Related Documentation 11
Books in This Documentation Set 11
Access Manager Policy Agent Documentation 12
Other Server Documentation 13
Accessing Sun Resources Online 13
Contacting Sun Technical Support 13
Related Third-Party Web Site References 14
Sun Welcomes Your Comments 14

Chapter 1 Overview of Access Manager 15
An Identity Management Paradigm 15
The Problem 16
The Solution 16
How Access Manager Works 17
Identity Administration 18
Access Management 18
Service Management 18
Federation Management 19
Access Manager Architecture 19

Sun Java System Directory Server 21
Access Manager Components 21
What’s New in This Release 21

Chapter 2 Identity Management 25
Basic Directory Server Concepts 25
Overview of the Directory Tree 25
Directory Entries and the Base DN 27
Directory Server Schema 27
Static and Dynamic Groups 27
Managed and Filtered Roles 28
How Access Manager Works with Directory Server 28
Access Manager Objects Are Added to Directory 28
Groups 29
Users 29
Services 29
Roles 29
Policies 29
Containers 29
People Containers 30
Group Containers 30
Delegated Administration and Self-Registration 30
Identity Management Interfaces 31

Chapter 3 Access Management 33
Authentication 33
Basic Authentication Concepts 34
Authentication Modules 35
Active Directory Authentication Module 36
Anonymous Authentication Module 36
Certificate Authentication Module 36
HTTP Basic Authentication Module 37
JDBC Authentication Module 38
LDAP Authentication Module 38
Membership Authentication Module 38
MSISDN Authentication Module 38
RADIUS Authentication Module 38
SAML Authentication Module 38
SafeWord Authentication Module 38
SecurID Authentication Module 39
UNIX® Authentication Module 39
Windows Desktop SSO Module 39

Windows NT Authentication Module . . . 39
Authentication Services . . . 39
Core Authentication Service . . . 39
Authentication Configuration Service . . . 40
Client Detection . . . 40
Authentication Types . . . 40
Redirection URLs . . . 41
Account Locking . . . 41
Authentication Module Chaining . . . 42
Fully Qualified Domain Name Mapping . . . 42
Persistent Cookie . . . 43
Session Upgrade . . . 43
Validation Plug-in Interface . . . 43
JAAS Shared State . . . 43
Authentication Interfaces . . . 44
User Interface . . . 44
Programming Interfaces . . . 45
Policy Management and Configuration . . . 46
Policy Framework . . . 46
Policy Configuration . . . 46
Policy Agents . . . 47
Policy Types . . . 47
Conditional Policy . . . 47
Referral Policy . . . 48
Policy Management Architecture . . . 49
Single Sign-On . . . 51
Cross-Domain Single Sign-On . . . 51
Policy Agents . . . 51
Cross-Domain Controller . . . 52

Chapter 4 Services Management . . . 55
How Services Work in Access Manager . . . 55
Core Services . . . 56
Service Plug-Ins . . . 56
Attribute Types . . . 57
Dynamic Attributes . . . 57
User Attributes . . . 57
Organization Attributes . . . 57
Global Attributes . . . 58
Policy Attributes . . . 58
Identity Server Core Services . . . 58
Administration . . . 58
Authentication . . . 59

Authentication Configuration 60
Client Detection 60
Logging 60
Naming 60
Password Reset 61
Platform 61
Policy Configuration 61
SAML 61
Session 61
User 61
The Service Configuration Interface 62

Chapter 5 Federation Management 63
The Need for Federated Identities 63
The Liberty Alliance Project 64
The Circle of Trust 65
Federation Management Architecture 67
Identity Federation Framework 67
Identity Web Services Framework 68
Identity Service Instance Specifications (ID-SIS) 71
Supporting Components 71
The Federation Management Process 71
Federation Single Sign-On Process 72
Pre-Login Process 72
System Flow 75

Glossary 77

Index 79

Preface

This *Technical Overview* provides a high-level overview of how Sun Java™ System Access Manager (formerly Sun Java System Identity Server) components work together to consolidate identity management and to protect enterprise assets and web-based applications. It explains basic Access Manager concepts and terminology. This book is designed to help you identify topics relevant to your enterprise needs so that you can explore those topics more fully in other Access Manager documentation.

Who Should Use This Book

This *Technical Overview* is intended for use by IT administrators and software developers who implement an integrated identity management and web access platform using Sun Java System servers and software. It is recommended that administrators understand the following technologies:

- Lightweight Directory Access Protocol (LDAP)
- Java™ technology
- JavaServer Pages™ (JSP) technology
- HyperText Transfer Protocol (HTTP)
- HyperText Markup Language (HTML)
- eXtensible Markup Language (XML)

Before You Read This Book

Access Manager is a component of Sun Java Enterprise System, a software infrastructure that supports enterprise applications distributed across a network or Internet environment. You should be familiar with the documentation provided with Sun Java Enterprise System, which can be accessed online at `http://docs.sun.com/coll/entsys_04q4`.

Because Sun Java System Directory Server is used as the data store in an Access Manager deployment, you should be familiar with the documentation provided with that product. Directory Server documentation can be accessed online at `http://docs.sun.com/coll/DirectoryServer_04q2`.

Conventions Used in This Book

The tables in this section describe the conventions used in this book.

Typographic Conventions

The following table describes the typographic changes used in this book.

Table 1 Typographic Conventions

Typeface	Meaning	Examples
`AaBbCc123` (Monospace)	API and language elements, HTML tags, web site URLs, command names, file names, directory path names, onscreen computer output, sample code.	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`

`AaBbCc123` (Monospace bold)	What you type, when contrasted with onscreen computer output.	`%` **`su`**
		`Password:`

AaBbCc123 (Italic)	Book titles, new terms, words to be emphasized. A placeholder in a command or path name to be replaced with a real name or value.	Read Chapter 6 in the *User's Guide*. These are called *class* options. Do *not* save the file. The file is located in the *install-dir*/`bin` directory.

Symbols

The following table describes the symbol conventions used in this book.

Table 2 Symbol Conventions

Symbol	Description	Example	Meaning
`[ ]`	Contains optional command options.	`ls [-l]`	The `-l` option is not required.
`{ | }`	Contains a set of choices for a required command option.	`-d {y|n}`	The `-d` option requires that you use either the `y` argument or the `n` argument.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
>	Indicates menu item selection in a graphical user interface.	File > New > Templates	From the File menu, choose New. From the New submenu, choose Templates.

Default Paths and File Names

The following table describes the default paths and file names used in this book.

Example

Table 3 Default Paths and File Names

Term	Description
AccessManager-base	Represents the base installation directory for Access Manager. The Access Manager 2005Q1 default base installation and product directory depends on your specific platform: Solaris™ systems: `/opt/SUNWam` Linux systems: `/opt/sun/identity`
DirectoryServer-base	Represents the base installation directory for Sun Java System Directory Server. Refer to the product documentation for the specific path name.

Table 3 Default Paths and File Names *(Continued)*

Term	Description
ApplicationServer-base	Represents the base installation directory for Sun Java System Application Server. Refer to the product documentation for the specific path name.
WebServer-base	Represents the base installation directory for Sun Java System Web Server. Refer to the product documentation for the specific path name.

Shell Prompts

The following table describes the shell prompts used in this book.

Table 4 Shell Prompts

Shell	Prompt
C shell on UNIX or Linux	*machine-name*%
C shell superuser on UNIX or Linux	*machine-name*#
Bourne shell and Korn shell on UNIX or Linux	`$`
Bourne shell and Korn shell superuser on UNIX or Linux	`#`
Windows command line	`C:\`

Related Documentation

To access Sun technical documentation online, go to `http://docs.sun.com`.

You can browse the documentation archive or search for a specific book title, part number, or subject.

Books in This Documentation Set

Table 5 Access Manager 6 2005Q1 Documentation Set

Title	Description
Technical Overview `http://docs.sun.com/doc/817-7643`	Provides a high-level overview of how Access Manager components work together to consolidate identity management and to protect enterprise assets and web-based applications. Explains basic Access Manager concepts and terminology
Deployment Planning Guide `http://docs.sun.com/doc/817-7644`	Provides information about planning a deployment within an existing information technology infrastructure
Administration Guide `http://docs.sun.com/doc/817-7647`	Describes how to use the Access Manager console as well as manage user and service data via the command line.
Migration Guide `http://docs.sun.com/doc/817-7645`	Describes how to migrate existing data and Sun Java System product deployments to the latest version of Access Manager. (For instructions about installing and upgrading Access Manager and other products, see the *Sun Java Enterprise System 2005Q1 Installation Guide*.)
Performance Tuning Guide (this guide) `http://docs.sun.com/doc/817-7646`	Describes how to tune Access Manager and its related components.
Federation Management Guide `http://docs.sun.com/doc/817-7648`	Provides information about Federation Management, which is based on the Liberty Alliance Project.
Developer's Guide `http://docs.sun.com/doc/817-7649`	Offers information on how to customize Access Manager and integrate its functionality into an organization's current technical infrastructure. Contains details about the programmatic aspects of the product and its API.
Developer's Reference `http://docs.sun.com/doc/817-7650`	Provides summaries of data types, structures, and functions that make up the Access Manager public C APIs.

Table 5 Access Manager 6 2005Q1 Documentation Set *(Continued)*

Title	Description
Release Notes http://docs.sun.com/doc/817-7642	Available after the product is released. Contains last-minute information, including a description of what is new in this current release, known problems and limitations, installation notes, and how to report issues with the software or the documentation.

Access Manager Policy Agent Documentation

Documentation for the Access Manager Policy Agents is available on the following documentation Web site:

http://docs.sun.com/coll/S1_IdServPolicyAgent_21

Policy Agents for Access Manager are available on a different schedule than the server product itself. Therefore, the documentation set for the policy agents is available outside the core set of Access Manager documentation. The following titles are included in the set:

- *Policy Agents For Web and Proxy Servers Guide* documents how to install and configure an Access Manager policy agent on various web and proxy servers. It also includes troubleshooting and information specific to each agent.
- *J2EE Policy Agents Guide* documents how to install and configure an Access Manager policy agent that can protect a variety of hosted J2EE applications. It also includes troubleshooting and information specific to each agent.
- The *Release Notes* are available online after a set of agents is released. The *Release Notes* include a description of what is new in the current release, known problems and limitations, installation notes, and how to report issues with the software or the documentation.

Other Server Documentation

For other server documentation, go to the following:

- Directory Server documentation
  `http://docs.sun.com/coll/DirectoryServer_05q1`
- Web Server documentation
  `http://docs.sun.com/coll/WebServer_05q1`
- Application Server documentation
  `http://docs.sun.com/coll/ApplicationServer8_ee_04q4`
- Web Proxy Server documentation
  `http://docs.sun.com/prod/s1.webproxys#hic`

Accessing Sun Resources Online

For product downloads, professional services, patches and support, and additional developer information, go to the following:

Download Center
`http://wwws.sun.com/software/download/`

Sun Java System Services Suite
`http://www.sun.com/service/sunps/sunone/index.html`

Sun Enterprise Services, Solaris Patches, and Support
`http://sunsolve.sun.com/`

Developer Information
`http://developers.sun.com/prodtech/index.html`

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in the product documentation, go to:

`http://www.sun.com/service/contacting`.

Related Third-Party Web Site References

Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to `http://docs.sun.com` and click Send Comments. In the online form, provide the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the guide or at the top of the document.

For example, the title of this guide is *Sun Java System Access Manager 6 2005Q1 Technical Overview*, and the part number is 817-7643.

Chapter 1

Overview of Access Manager

Sun Java System Access Manager 6 2005Q1 provides a comprehensive solution for managing federated identities and for enforcing authorized access to network services and resources. It tightly integrates the policy, identity management, service management, and SAML to simplify the administration of users and to provide a single identity across a range of web and application servers. Access Manager also serves the base for the Liberty specification implementation.

This chapter provides an overview Access Manager and how its components work together. Topics include:

- "An Identity Management Paradigm" on page 15
- "How Access Manager Works" on page 17
- "Access Manager Architecture" on page 19
- "What's New in This Release" on page 21

An Identity Management Paradigm

Think of all the different types of information a company must store and be able to make available through its enterprise. Now consider the various enterprise users who must make use of that information in order for the company's business to run smoothly. For example, the following are routine information transactions that occur every day in a typical company:

- A rank-and-file employee looks up a colleague's phone number in the corporate phone directory.
- A managing employee looks up the salary histories of all her reports to help determine an individual's merit raise.

- An administrative assistant adds a new hire to the corporate database, which triggers the company’s health insurance provider to add the new hire to its enrollment.
- An engineer sends an internal URL for a specification document to another engineer who works for a partner company.
- A customer logs into the company’s website and looks for a product in the company’s online catalog.
- A vendor submits an online invoice to the company’s accounting department.

In each of these examples, the company must determine who is allowed to view its information or use its applications. Some information such as the company’s product descriptions and advertising can be made available to everyone, even the public at large, in the company’s online catalog. Other information such as accounting and human resources information must be restricted to only employee use. And some internal information is appropriate to share with partners and suppliers, but not with customers.

The Problem

Many enterprises grant access to information on a per-application basis. For example, an employee might have to set up a user name and password to access the company’s health benefits administration website, and a separate user name and password to access the accounting department online forms. A customer sets up a user name and password to access the “Customers” branch of the company website. For each website or service, there is an administrator who converts the enterprise user’s input into a data format that the service can recognize. Each service added to the enterprise must be provisioned and maintained separately.

The Solution

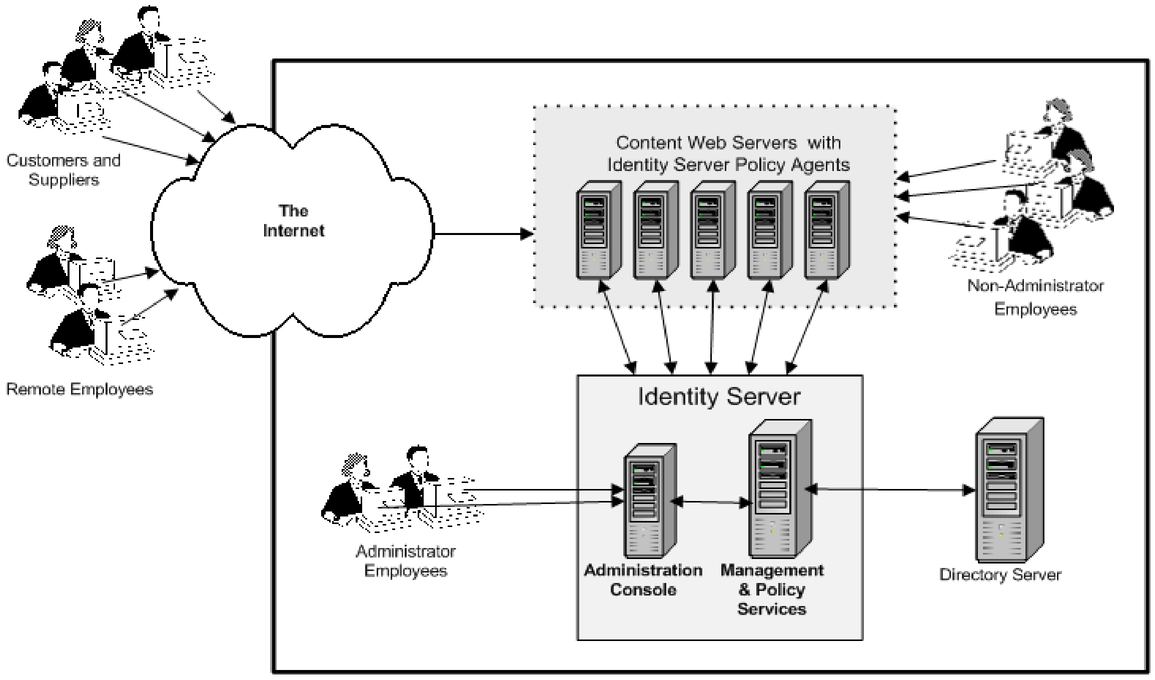
Access Manager reduces the administrative costs and eliminates the redundant user information associated with per-application solutions. Access Manager creates a single record or *directory entry* for each enterprise user, and enables an administrator to assign specific rules or *policies* governing which information or services each user can access. Policy *agents* can be deployed on application or web servers to enforce the policies. Together, a user’s directory entry and its associated access policies comprise the user’s enterprise *identity*. Access Manager makes it possible for a user to access many resources in the enterprise with just one identity.

How Access Manager Works

When an enterprise user or an external application tries to access content stored on a company's web server, the policy agent intercepts the request and directs it to Access Manager. Access Manager asks the user to present credentials such as a username and password. If the credentials match those stored in the central Directory Server, Access Manager verifies that the user is who he says he is. Next, Access Manager evaluates the policies associated with the user's identity, and then determines whether the user is allowed to view the requested information.

Finally, Access Manager either grants or denies the user access to the information. Figure 1-1 illustrates one way Access Manager can be configured to act as the gatekeeper to a company's information resources.

Figure 1-1 Access Manager is the gatekeeper to a company's enterprise resources.



Access Manager consolidates four major features into a single product that can be viewed in a single administration console:

- Identity Administration
- Access Management
- Service Management
- Federation Management

Identity Administration

Access Manager provides an identity framework for creating and managing directory objects such as organizations, groups, roles, and userIDs. When you use Access Manager to create or modify user objects, you update the entries stored in Directory Server. Access Manager schema includes pre-defined administrator userIDs and associated access control instructions (ACIs). This makes it possible to delegate user management tasks to various administrators—and to non-administrators as well—in the enterprise. The Identity Management functionality is further described in Chapter 2, "Identity Management" on page 25.

Access Management

Access Manager implements authentication service and policy administration to regulate access to a company's information and applications. These features make it possible to verify that a user is who he says he is, and that the user is authorized to access web or application servers deployed within the enterprise. The Access Management functionality is further described in Chapter 3, "Access Management" on page 33.

Service Management

Access Manager provides a service management SDK that gives application developers the interfaces necessary to register and un-register services as well as to manage schema and configuration information. It also provides a number of services that it uses for authentication and for its own administration. The Service Management functionality is further described in Chapter 4, "Services Management" on page 55.

Federation Management

Identity federation allows a user to link the many local identities he has configured among multiple service providers. With one *federated identity*, the individual can log in at one service provider's site and move to an affiliated service provider site without having to re-authenticate or re-establish his identity. The Federation Management functionality is further described in Chapter 5, "Federation Management" on page 63.

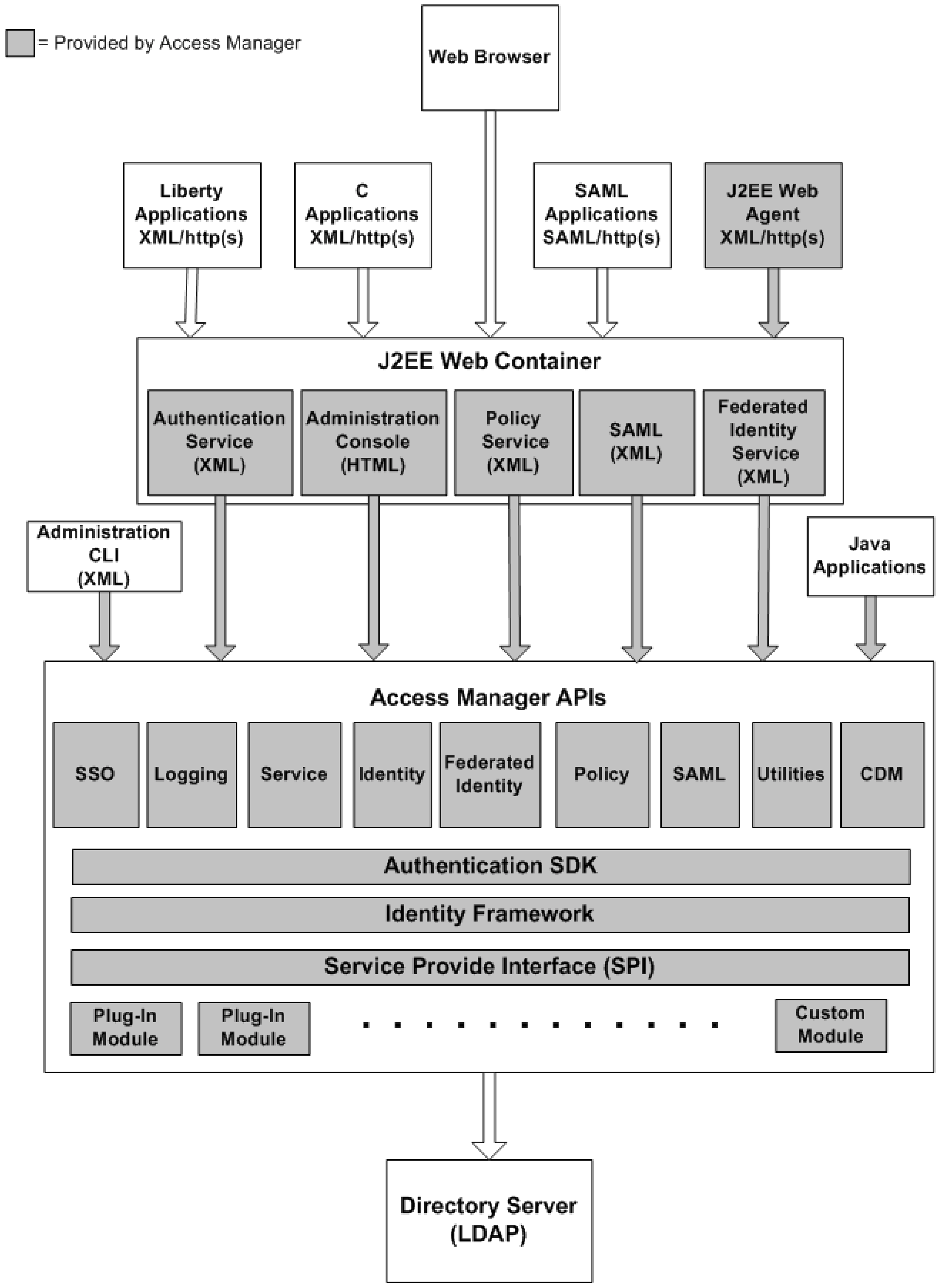
Access Manager Architecture

Access Manager uses a Java technology-based architecture for scalability, performance, and ease of development. It leverages industry standards including the following:

- HyperText Transfer Protocol (HTTP)
- eXtensible Markup Language (XML)
- Simple Object Access Protocol (SOAP)
- Security Assertions markup Language (SAML) specification

Figure 1-2 illustrates how Access Manager integrates all of these technologies and connects to Directory Server. The Access Manager common identity infrastructure is built upon Directory Server which uses the LDAP protocol.

Figure 1-2 Access Manager Architecture.



Sun Java System Directory Server

In an Access Manager deployment, Directory Server acts as the centralized repository for user identities. Identities are stored as directory entries using the LDAP protocol and Directory Services Markup Language (DSML). LDAP is the "lightweight" version of the Directory Access Protocol (DAP) used by the ISO X.500 standard. DSML enables you to represent directory entries and commands in XML. This makes it possible for XML-based applications using HTTP to take advantage of directory services while making full use of the existing web infrastructure.

Access Manager Components

Access Manager functions are delivered as a collection of Java servlets, JavaBeans components, and JSP modules. Authentication Service, Policy Service, and an Administration Console are examples of such functions. These run inside the Java virtual machine of a J2EE container such as Sun Java System Web Server or Sun Java System Application Server.

Access Manager includes APIs for Single Sign-On, Logging, Identity, Federated Identity, Policy, SAML, and more. These public Java APIs provide an interface that external applications can use to implement either default or customized behavior.

Policy agents are an integral part of the identity management solution. Installed on web servers or web proxy servers in the enterprise, policy agents protect individual servers from unauthorized intrusions.

What's New in This Release

New features in Access Manager 2005Q1 include the following:

- Product name has changed from Identity Server to Access Manager
- Support for new web containers: BEA WebLogic 8.1.x and IBM WebSphere Application Server 5.1.x
- New authentication modules:
  - Java Database Connectivity (JDBC)
  - Mobile Station ISDN (MSISDN)
  - Active Directory

  - Security Assertion Markup Language (SAML)

- Policy Management includes a new Resource Name plug-in: `HttpURLResourceName`.

- Console enhancements:

  - Ability to customize the display of organizations as well as different attributes for each object type in the left navigation pane to include a descriptive name (such as the cn).

  - Ability to modify the contents of the drop-down menu in the left navigation pane (for example, to add custom groups or roles).

- Session failover:

  - Web-container independent: Support for Web Server, Application Server, IBM WebSphere Application Server, or BEA WebLogic as the web container.

  - Session repository uses the Berkeley DB by Sleepycat Software, Inc and Sun Java System Message Queue (Message Queue) as the communications broker.

  - Connection pooling based on the JDBC 2.0 driver connection pool interfaces.

- Federation Management:

  - Support for the Liberty Alliance Project (LAP) Name Identifier Mapping Protocol

  - Support for the LAP Identity Web Services Framework (ID-WSF) Discovery Service Specification, Version 1.1

  - Support for the LAP ID-WSF Authentication Service Specification

  - Support for the LAP Metadata Description and Discovery Specification

  - Support for the LAP Liberty Identity Federation Framework (ID-FF) Extended Profiles:

    - Dynamic Identity Provider Proxying

    - Affiliation Federation

    - One-time Federation

    - Name Identifier Mapping Profile

    - Name Identifier Encryption Profile

- Client SDK:
  - Repackaged SDK (Authentication, Service Management, User Management, SAML, Policy Client, and Session components) so Java application developers can better integrate with Access Manager.
  - Removed the dependency on the serverconfig.xml file and minimized the footprint of the jar files.
- Performance tuning script is available to tune Application Server 8.1 as a web container

Chapter 2

Identity Management

Built upon Sun Java™ System Directory Server, Sun Java System Access Manager 6 provides a means for creating and modifying directory entries and access policies. Together, a user's directory entry and its associated policies form the user's identity. This chapter explains how Directory Server and Access Manager work together to achieve consolidated identity management.

Topics in this chapter include the following:

- "Basic Directory Server Concepts" on page 25
- "How Access Manager Works with Directory Server" on page 28

Basic Directory Server Concepts

Directory Server provides a central repository for storing and managing information. Almost any kind of information can be stored, from identity profiles and access privileges to information about application and network resources, printers, network devices and manufactured parts. Access Manager connects to Directory Server and accesses the identity profiles and services information stored there. The information is stored in directory entries, and entries are grouped hierarchically in a *directory tree*.

Overview of the Directory Tree

The Directory Server directory tree, also known as a directory information tree or DIT, mirrors the tree model used by most file systems. The tree's root, or first entry, appears at the top of the hierarchy. The root of the tree is called the *root suffix*. You can build on the default directory tree to add any data relevant to your directory installation.

When you install Access Manager, you are asked to name an Access Manager root suffix. The Access Manager root suffix identifies the part of the directory tree that is managed by Access Manager. The Access Manager root suffix can start beneath the root suffix of the directory tree, or it can replace the root suffix of the directory tree.

In Figure 2-1, the directory tree root suffix is `dc=example,dc=com`. Additional subtrees have been added to reflect an organizational hierarchy. In the example:

– `dc` refers to the domain component of the enterprise

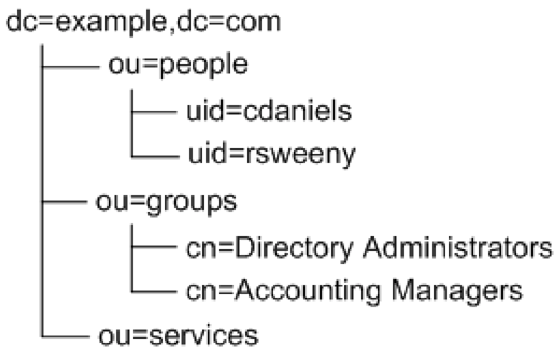
– `ou` refers to an organizational unit or group to which an object belongs

– `uid` refers to the string or name that Access Manager associates with a user object

– cn refers to a common name given to a directory object; the name that is visible to end-users of an application

When Access Manager is installed, if the same root suffix `dc=example,dc=com` is specified, then all user entries in the entire directory tree can be managed by Access Manager.

Figure 2-1 Sample Directory Tree



Directory Entries and the Base DN

Each user, service, and resource in the directory is represented by a directory entry. A directory entry stores parameter values which describe a user, service, or resource.

In LDAP, you can query an entry and request all entries below it in the directory tree. This subtree is called the base distinguished name, or *base DN*. For example, in the sample directory tree (Figure 2-1) when you use Access Manager to add a user to a group, Access Manager connects to Directory Server to find the user's entry. On the back end, the LDAP search function requests entries specifying a base DN of `ou=people,dc=example,dc=com`. The search operation examines only the `ou=people` subtree in the `dc=example,dc=com` directory tree, and ignores the `ou=services` subtree.

Directory Server Schema

The predefined schema included with Directory Server contains both the standard LDAP attributes and object classes as well as additional application-specific schema to support the features of the server.

The directory tree mechanism is not well suited for associations between dispersed entries, for frequently changing organizations, or for data that is repeated in many entries. As a solution, groups and roles provide more flexible associations between entries, and class of service simplifies the management of data that is shared within branches of your directory. Access Manager schema leverages the attributes and object classes that come with Directory Server.

Static and Dynamic Groups

A group is an entry that specifies the other entries that are its members. When you know the name of a group, it is easy to retrieve all of its member entries.

- Static groups explicitly name their member entries. Static groups are suitable for groups with few members, such as the group of directory administrators.
- Dynamic groups specify a filter, and all entries that match are members of the group. These groups are dynamic because membership is defined every time the filter is evaluated.
- Access Manager groups are also used for defining policy subjects. See "Policy Configuration" on page 46 for related information. Note that Access Manager groups are *not* used for services inheritance.

The advantage of groups is that they make it easy to find all of their members. Static groups may simply be enumerated, and the filters in dynamic groups may simply be evaluated. The disadvantage of groups is that given an arbitrary entry, it is difficult to name all the groups of which it is a member.

Managed and Filtered Roles

Roles are an alternative entry grouping mechanism that automatically identifies all roles of which any entry is a member. When you retrieve an entry in the directory, you immediately know the roles to which it belongs. This overcomes the main disadvantage of the group mechanism.

- *Managed* roles are the equivalent of static groups, except that membership is defined in each member entry and not in the role definition entry.
- *Filtered* roles are similar to dynamic groups. They define a filter that determines the members of the role.
- In Directory Server, *nested* roles name other role definitions, including other nested roles. The set of members of a nested role is the union of all members of the roles it contains. However, it's important to note that nested roles are *not* supported in the Access Manager administration console.

How Access Manager Works with Directory Server

When you install Access Manager, it adds its own specialized object classes, roles, and services to the directory tree. These form an identity framework that enables you to use the Access Manager administration console to create and manage the directory entries in Directory Server.

Access Manager Objects Are Added to Directory

Access Manager directory objects extend the Directory Server schema. Since they are abstractions based upon Directory Server objects, Access Manager objects are similar—but not always identical—to Directory Server objects.

Groups

An Access Manager group represents a collection of users with a common function, feature or interest. Typically, this grouping has no privileges associated with it. Groups can exist at two levels, within an organization and within other managed groups such as a sub group. Users can be added to managed groups either statically or dynamically. You must manually add or delete each individual user to a static group; dynamic groups are formed automatically through the use of search filter.

Users

A user represents the identity of a person.

Services

A *service* is a group of attributes that are managed together by the Access Manager console. Access Manager services are discussed in greater detail in Chapter 4, "Services Management" on page 55.

Roles

An Access Manager *role*, like a Directory Server role, is an entry mechanism similar to the concept of a *group*. Identity Server uses roles to apply access control instructions (ACIs). ACIs define who has access to specific directory entries. Access Manager roles are also used as policy subjects for the purpose of service inheritance. See "Policy Configuration" on page 46 for related information.

Policies

Policies are similar to ACIs in the way they define access rules. But while ACIs describe who has access to directory entries, policies describe who has access to specific resources such as a server or a document stored on a server. Access Manager objects are added to a policy through the policy's subject definition.

Containers

The container entry is used to group objects when, due to object class and attribute differences, it is not possible to use an organization entry. It is important to remember that the Access Manager container entry and the Access Manager organization entry are not necessarily equivalent to the LDAP object classes `organizationalUnit` and `organization`. They are abstract Identity entries. Ideally, the organization entry will be used instead of the container entry.

People Containers

A People Container is the default LDAP organizational unit to which all users are assigned when they are created within an organization. People containers can be found at the organization level and at the people container level as a sub People Container. They can contain only other people containers and users.

Group Containers

A Group Container is used to manage groups. It can contain only groups and other group containers. The group container Groups is dynamically assigned as the parent entry for all managed groups.

Delegated Administration and Self-Registration

When you install Access Manager, Access Manager automatically creates four administrator roles and adds them to the directory:

- Top-Level Admin Role
- Top-level Help Desk Admin Role
- Top-level Policy Admin Role
- People Admin Role

Access control instructions (ACIs) are associated with each administrator role. When you add a user, or *member*, to one of these pre-defined roles, the member is accorded the directory access privileges associated with the role. For example, users who are assigned to the Top-level Admin Role can access and modify all entries in the directory tree. A user who is a member of the Top-Level Help Desk Admin Role can search all entries, but can modify only the password information in each entry. A user who is a member of the Top-level Policy Admin role can modify only policy-related information in each entry. A user who is a member of the People Admin role has read and write access to all user-related information in each entry, but not policies.

The Identity Framework enables you to create additional administrator roles at the organization level and at the group level of the directory tree. In this way, the administration workload can be selectively distributed, or *delegated*, among a large number of administrators who have restricted access, rather than to just small number of omni-privileged administrators. This speeds up administration workflow.

Administration can also be delegated down to non-administrators. Users can access the Access Manager console via the HTTP and a browser. This makes it possible for non-administrators to gain restricted access to the company's resources or applications without having to install a proprietary application. For example, a company can set up its online product catalog so that customers must register a username and password before accessing the catalog. Through self-registration, the customer can create his own account and password without any intervention by an administrator. This reduces the administrator workload.

Identity Management Interfaces

To bridge the gap between Directory Server object classes and Access Manager functionality, Access Manager provides the following interfaces:

- ums.xml

  This file defines a set of *templates* that contain configuration information needed to set up each identity-related object created with Access Manager as an LDAP entry in the Directory Server data store.

- Identity Management Software Development Kit (SDK)

  The SDK is used to integrate the management functions of Access Manager into external applications or services.

- Access Manager console

  When you modify a user entry using the Access Manager console, the modification is automatically made in the Directory Server.

Chapter 3

Access Management

Sun Java™ System Access Manager 6 2005Q1 implements authentication and policy administration to regulate access to the many types of information stored in a company's enterprise. When an enterprise user requests information, Access Manager verifies that 1) the user is who he says he is, and 2) the user is authorized to access the specific resource he's requested. This chapter provides an overview view of Access Manager access management features and functionality.

Topics included in this chapter are:

- "Authentication" on page 33
- "Policy Management and Configuration" on page 46

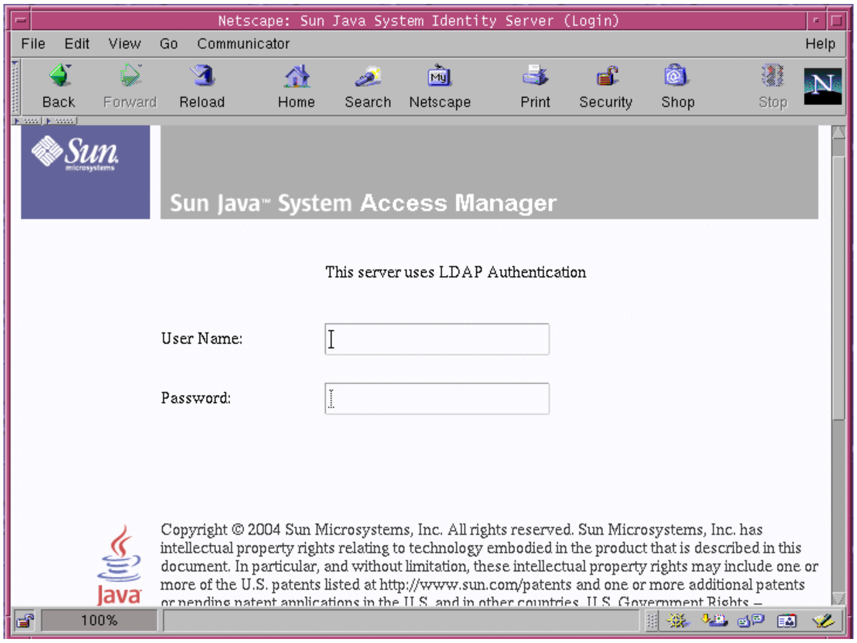
Authentication

Authentication is the process of verifying that a person is who he says he is. Access Manager comes with an authentication service for verifying the identities of users who request access to web resources within an enterprise. For example, a company employee who needs to look up a colleague's phone number uses a browser to go to the company's online phone book. To log in to the phone book service, the employee must provide his user name and password. Access Manager compares the user's input with data stored in Directory Server. If Access Manager finds a match for the user name, and if the given password matches the password stored in Directory Server, then Access Manager can verify the user's identity, and the user is authenticated. Access is granted, and the corporate phone book is displayed to the user.

Basic Authentication Concepts

The Authentication Service has a user interface that is separate from the Access Manager administration console. The user interface provides a dynamic and customizable means for gathering authentication credentials. When a user requests access to a protected resource, the Authentication Service presents a web-based login page. In Figure 3-1, the default Access Manager login page, is displayed and prompts the user for user name and password.

Figure 3-1 LDAP Authentication User Interface


Netscape: Sun Java System Identity Server (Login)
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Sun microsystems
Sun Java™ System Access Manager
Java
100%

This server uses LDAP Authentication

User Name:

Password:

Copyright © 2004 Sun Microsystems, Inc. All rights reserved. Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and in other countries. U.S. Government Rights –

In Access Manager, authentication is achieved through the use of authentication modules and features. These are described in the following sections:

- Authentication Modules
- Authentication Services
- Authentication Types
- Client Detection
- Account Locking

- Authentication Module Chaining
- Fully Qualified Domain Name Mapping
- Persistent Cookie
- Session Upgrade
- Validation Plug-in Interface
- JAAS Shared State

Authentication Modules

Access Manager is installed with a set of default authentication *modules*. An authentication module is a plug-in that collects user information such as a user ID and password, and then checks the information against entries in a database. If the user information provided meets the authentication criteria, then the user is granted access to the requested resource; otherwise, the user is denied access to the requested resource.

After granting or denying access, Access Manager checks for information on where to redirect the user. There is an order of precedence in which Access Manager checks for this information. The order is based on whether the user was granted or denied access to the protected resource, and on the *type* of authentication specified. See "Authentication Types" on page 40 for more information on authentication types.

You can use the Access Manager administration console to enable and the configure authentication modules that come with Access Manager by default. See the *Administration Guide* for detailed information on enabling and configuring default authentication modules.

You can also write your own custom authentication modules to plug in to the Access Manager authentication framework. For more information on writing custom modules see the *Developer's Guide*.

The following are the authentication modules that are installed by default when you install Access Manager.

- Active Directory Authentication Module
- Anonymous Authentication Module
- Certificate Authentication Module
- HTTP Basic Authentication Module

- JDBC Authentication Module
- LDAP Authentication Module
- Membership Authentication Module
- MSISDN Authentication Module
- RADIUS Authentication Module
- SAML Authentication Module
- SafeWord Authentication Module
- SecurID Authentication Module
- UNIX® Authentication Module
- Windows Desktop SSO Module
- Windows NT Authentication Module

Active Directory Authentication Module

This module works similarly to the LDAP authentication module, but uses the Microsoft Active Directory instead of an LDAP directory. Using this module makes it possible to have both LDAP and Active Directory coexist under the same organization.

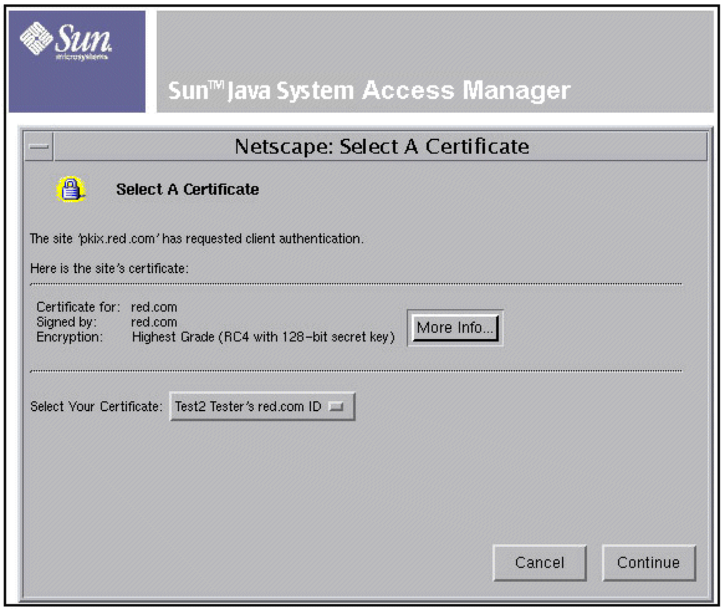
Anonymous Authentication Module

This module allows a user to log in without specifying credentials. You can create an Anonymous user so that anyone can log in as Anonymous without having to provide a password. Anonymous connections are usually customized by the Access Manager administrator so that Anonymous users have limited access to the server.

Certificate Authentication Module

This module allows a user to log in through a personal digital certificate (PDC). The module can require the use of the Online Certificate Status Protocol (OCSP) to determine the state of a certificate. Use of the OCSP is optional. The user is granted or denied access to a resource based on whether or not the certificate is valid.

Figure 3-2 Certificate-based Authentication Login Requirement Screen


Sun™ Java System Access Manager

Netscape: Select A Certificate

Select A Certificate

The site 'pkix.red.com' has requested client authentication.

Here is the site's certificate:

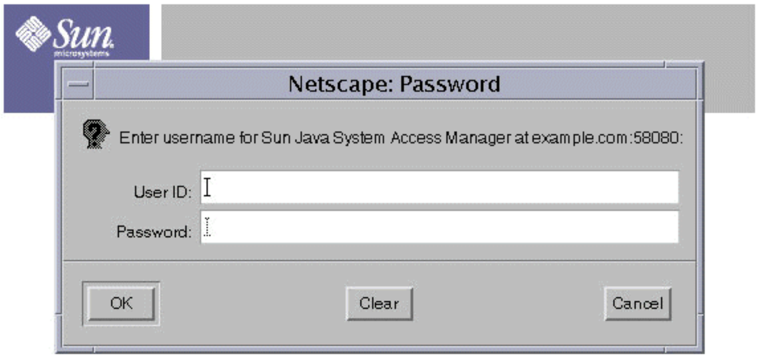
Certificate for: red.com
Signed by: red.com
Encryption: Highest Grade (RC4 with 128-bit secret key)

Select Your Certificate: Test2 Tester's red.com ID

HTTP Basic Authentication Module

This module allows login using the HTTP's basic authentication with no data encryption. A user name and password are requested through the use of a web browser. Credentials are validated internally using the LDAP authentication module.

Figure 3-3 HTTP Basic Authentication Login Requirement Screen


Netscape: Password

Enter username for Sun Java System Access Manager at example.com:58080:

User ID:

Password:

JDBC Authentication Module

The Java Database Connectivity (JDBC) authentication module makes it possible for Access Manager to authenticate users through any Structured Query Language (SQL) databases that provide JDBC-enabled drivers. The connection to the SQL database can be either directly through a JDBC driver or through a JNDI connection pool.

LDAP Authentication Module

This module allows for authentication using LDAP bind, a Directory Server operation which associates a user ID password with a particular LDAP entry. You can define multiple LDAP authentication configurations for an organization.

Membership Authentication Module

Membership authentication is implemented similarly to personalized sites such as my.site.com, or mysun.sun.com. When membership authentication is enabled, a user can *self-register*. This means the user can create an account, personalize it, and access it as a registered user—all without the help of an administrator.

MSISDN Authentication Module

The Mobile Station Integrated Services Digital Network (MSISDN) authentication module enables authentication using a mobile subscriber ISDN associated with a device such as a cellular telephone. It is a non-interactive module. The module retrieves the subscriber ISDN and validates it against the Directory Server to find a user that matches the number.

RADIUS Authentication Module

This module allows for authentication using an external Remote Authentication Dial-In User Service (RADIUS) server.

SAML Authentication Module

The Security Assertion Markup Language (SAML) authentication module receives and validates SAML Assertions on a target server.

SafeWord Authentication Module

This module allows for authentication using Secure Computing’s SafeWord® PremierAccess™ server software and SafeWord tokens.

SecurID Authentication Module

This module allows for authentication using RSA ACE/Server software and RSA SecurID authenticators.

UNIX® Authentication Module

This Solaris only module allows for authentication using a user's UNIX identification and password.

Windows Desktop SSO Module

This module is specific to Windows and is also known as Kerebos authentication. The user presents a Kerberos token to Access Manager through the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) protocol. The Windows Desktop SSO authentication plug-in module provides a client (user) with desktop single sign-on. This means that a user who has already authenticated with a key distribution center can be authenticated with Access Manager *without having to provide the login information again*.

Windows NT Authentication Module

This module allows for authentication against a Microsoft Windows® NT server.

NOTE In order to actualize the NT authentication module, Samba 2.2.2 must be downloaded and installed. Samba is a file and print server for blending Windows and UNIX® machines together without requiring a separate Windows NT/2000 Server. More information, and the download itself, can be accessed at http://wwws.sun.com/software/download/products/3e3af224.html.

Authentication Services

Two authentication services comprise the framework for integrating pluggable and customized authentication modules into Access Manager: Core authentication service, and authentication configuration service.

Core Authentication Service

This is a general authentication service used for server-related attribute configuration. Some of the attributes described in this service are default attributes for all Access Manager authentication modules.

The core authentication service must be registered as a service to an organization before any user can log in using the other authentication modules. The core authentication service allows the Access Manager administrator to define default values for an organization's authentication parameters. These values can then be used if no overriding value is defined in the specified authentication module. The default values for the Core Authentication Service are defined in the amAuth.xml file and stored in Directory Server after installation.

Authentication Configuration Service

This services describes all the dynamic attributes for service-based authentication.This service is used for roles. When you assign a service to a role, you can also assign other attributes such as a success URL or an authentication post-processing class to the role.

Client Detection

An initial step in the authenticating process is to identify the type of client making the `HTTP(S)` request. This Access Manager feature is known as client detection. The URL information is used to retrieve the client's characteristics. Based on these characteristics, the appropriate authentication pages are returned. For example, when a Netscape browser is used to request a web page, Access Manager displays an HTML login page. Once the user is validated, the client type (Netscape browser) is added to the session token.

Authentication Types

When you install Access Manager, a number of authentication types are automatically configured for you. Each type of authentication uses configured login URLs and a redirection URL in order of precedence. The following types of authentication are available to you by default when you install Access Manager.

Organization-based Authentication. This method of authentication allows a user to authenticate to an organization or sub-organization in the directory information tree.

Role-based Authentication. This method of authentication allows a user to authenticate to a *role* within an organization or sub-organization of the directory information tree. A role is a grouping of like items in the directory. A *static* role is created when an attribute is assigned to a specific user or container in the directory.

A filtered role is dynamically generated based on an attribute contained in the a user's or container's LDAP entry. For example, all users that contain a particular attribute, for example `employee`, can be automatically included in a filtered role named `employees`.

Service-based Authentication. This method of authentication allows a user to authenticate to a specific service or application registered to an organization or sub-organization.

User-based Authetnication. This method of authentication allows a user to authenticate using an authentication process configured specifically for him or her.

Authentication Level-based Authentication. This method of authentication allows an administrator to specify the security level of the modules to which identities can authenticate.

Module-based Authentication. This method of authentication allows a user to specify the module to which they will authenticate.

Redirection URLs

Upon a successful or failed authentication, Access Manager looks for information on where to redirect the user. There is an order of precedence in which the application will look for this information based on the authentication method and whether the authentication has been successful or has failed. A detailed description of the order of precedence is described in the *Access Manager Administration Guide*.

Account Locking

The Authentication Service provides a feature that can *lock out*, or prevent a user from completing the authentication process, after *n* failures. This feature is disabled by default, but can be enabled using the Access Manager console. Only modules that throw an Invalid Password Exception can leverage the Account Locking feature. Email notifications are sent to administrators regarding any account lockouts. Account locking activities are also logged.

Access Manager supports two types of account locking are supported: Physical Locking and Memory Locking.

Physical Locking. By default, user accounts are active, or *physically unlocked*. Physical locking is initiated by changing the status of an LDAP attribute in the user's profile to inactive. The account is locked until the attribute is changed to active.

Memory Locking. *Memory locking* is enabled by changing the Login Failure Lockout Duration attribute to a value greater then 0. The user's account is then locked in memory for the number of minutes specified. The account will be unlocked after the time period has passed.

Authentication Module Chaining

One or more authentication modules can be configured so a user must pass authentication credentials to all of them. This is referred to as *authentication chaining*. Authentication chaining in Access Manager is achieved using the JAAS framework integrated in the Authentication Service. Module chaining is configured under organization, user, role, or service Authentication Configuration. Each registered module is assigned one of the following control flags:

Requisite. The LoginModule is required to succeed. If it succeeds, authentication continues down the LoginModule list. If it fails, control immediately returns to the application (authentication does not proceed down the LoginModule list).

Required. Authentication to this module is required to succeed. If any of the required modules in the chain fails, the whole authentication chain will ultimately fail. However, whether a required module succeeds or fails, the control will continue down to the next module in the chain.

Sufficient. The LoginModule is not required to succeed. If it does succeed, control immediately returns to the application (authentication does not proceed down the LoginModule list). If it fails, authentication continues down the LoginModule list.

Optional. The LoginModule is not required to succeed. If it succeeds or fails, authentication still continues to proceed down the LoginModule list.

Once authentication to all modules in the chain is successful, control is returned to the Authentication Service (from the JAAS framework) which validates all the user IDs used to authenticate, and then maps them to one user. The mapping is achieved by configuring the User Alias List attribute in the user's profile. If all the maps are correct, then a valid session token is issued to the user. If all the maps are not correct, then the user is denied a valid session token.

Fully Qualified Domain Name Mapping

Fully Qualified Domain Name (FQDN) mapping enables the Authentication Service to take corrective action in the case where a user may have typed in an incorrect URL. This is necessary, for example, when a user may specifies a partial host name or IP address to access protected resources.

Persistent Cookie

A persistent cookie is one that continues to exist after the web browser is closed, allowing a user to login with a new browser session without having to reauthenticate.

Session Upgrade

The Authentication Service allows for the upgrading of a valid session token based on a second, successful authentication performed by the same user to one organization. If a user with a valid session token attempts to authenticate to a resource secured by his current organization and this second authentication request is successful, the session is updated with the new properties based on the new authentication. If the authentication fails, the user's current session is returned without an upgrade. If the user with a valid session attempts to authenticate to a resource secured by a different organization, the user will receive a message asking whether they would like to authenticate to the new organization. The user can, at this point, maintain the current session or attempt to authenticate to the new organization. Successful authentication will result in the old session being destroyed and a new one being created.

Validation Plug-in Interface

An Administrator can write username or password validation logic suitable to their organization, and plug this into the Authentication Service. (This functionality is supported only by the LDAP and Membership authentication modules.) Before authenticating the user or changing the password, Access Manager will invoke this plugin. If the validation is successful, authentication continues; if it fails, an authentication failed page will be thrown. The plugin extends the com.iplanet.am.sdk.AMUserPasswordValidation class which is part of the Service Management SDK.

JAAS Shared State

The JAAS shared state provides sharing of both user ID and password between authentication modules. Options are defined for each authentication module for:

- Organization
- User

- Service
- Role

If the module fails with the credentials from the shared state, it restarts the authentication by prompting for its required credentials. If it fails again, the module is marked failed.

After a commit, an abort or a logout, the shared state will be cleared.

Authentication Interfaces

The Authentication Service provides two means by which you can implement authentication in your enterprise:

- User Interface
- Programming Interfaces

Use the User Interface when you want to enable or to customize authentication modules that are installed by default with Access Manager. Use the Programming Interfaces when you want to develop a new authentication module.

User Interface

A user with a web browser can authenticate to Access Manager using the authentication user interface. This interface is accessed by entering the Authentication Service User Interface login URL in the browser's location bar. After entering the URL, the user is prompted to submit verifying credentials based on the invoked authentication module(s). Once the credentials have been passed back to Access Manager (assuming successful authentication), the user can gain access based on their privileges:

- Administrators can access the administration portion of the Access Manager console to manage their organization's identity data.
- Users can access their own profiles to modify personal data.
- A user can access a resource defined as a redirection URL parameter appended to the login URL.
- A user can access the resource protected by a policy agent.

NOTE An initial step in the authenticating process is to identify the type of client making the HTTP(S) request. The URL information is used to retrieve the browser's characteristics and, based on these characteristics, the correct authentication pages are returned; for example, HTML pages. Once the user is validated, the client type is added to the session token.

Programming Interfaces

Access Manager provides Java APIs and SPIs, as well as C-APIs.

Authentication Via The Java API

External Java applications can authenticate to Access Manager using the Authentication API For Java Applications. This API provides interfaces to initiate the authentication process and communicate authentication credentials to the Authentication Service; the Access Manager API are based on the Java Authentication and Authorization Service (JAAS) specification. JAAS are Java packages that enable services to authenticate and enforce access controls upon users. They implement a version of the standard Pluggable Authentication Module (PAM) framework, and support user-based authorization.

NOTE The JAAS packages, starting with javax.security.auth, can be found in the Javadocs for Java 2 Platform, Standard Edition (J2SE), version 1.4.2.

Developers can incorporate the API classes and methods into their Java applications to allow communication with the Authentication Service. The external application's Java request is converted to an XML message format and passed to Access Manager over HTTP(S). Once received, the XML message is converted back into a Java request which can be interpreted by the Authentication Service.

The Authentication Service returns login requirement screens based on the authentication module being accessed. The user returns authentication credentials and is allowed or denied access based on these credentials. See the Javadoc for a comprehensive listing of Java APIs.

Authentication Via The C API

Access Manager also includes an Authentication API for C applications to authenticate to the Access Manager. This API provides functions to initiate the authentication process and communicate authentication credentials to the Authentication Service. After passing the authentication process, a validated session token is sent back to the C application. See the *Access Manager 6 Developer's Reference* for a comprehensive listing of C-APIs.

Policy Management and Configuration

A policy is a rule that describes who is allowed or *authorized* to access a specific web resource. Access Manager provides a policy management service and policy configuration service for creating such rules and evaluating policies. Policy agents are an integral part of an Access Manager deployment, and they help to enforce access policies.

For example, in a typical enterprise, one policy is applied to an entire organization. The policy specifies that all users within the organization can access all web servers within the enterprise firewall. Another policy is applied to employees in the Human Resources group, and specifies that they are authorized to access all web servers that store Human Resources data. A third policy specifies that employees who are mangers are authorized to confidential personnel records stored on Human Resources web servers. When an employee tries to access the confidential records stored on the Human Resources web servers, the policy service evaluates the policies that apply to the employee requesting the records. The policy service determines that although the employee is authorized to access servers within the firewall, the employee is not member of the Human Resources group nor of any manager group. Therefore the employee is not authorized to view the confidential personnel records he is requesting. An “access denied” message is displayed.

Policy Framework

The Policy Service provides a framework for defining, modifying, granting, revoking and deleting policies within an enterprise. The policy framework interacts with Sun Java System Directory Server for data storage. The policy framework also includes C APIs for policy evaluation, Java™ APIs for both policy evaluation and administration, and a selection of downloadable policy agents that enforce the policies.

Policy Configuration

The Policy Configuration Service is also provided to allow the configuration of policy-related attributes per organization. You can define in this service the resource name implementations and Directory Server data stores to use for authorization. For additional information, see “Chapter 33, Policy Configuration Attributes” of the *Sun Java System Access Manager Administration Guide*.

Policy Agents

The Policy Agent is the Policy Enforcement Point (PEP) for a server on which enterprise resources are stored. Policy agents are provided as self-contained components, and are separate from Access Manager. All of the Access Manager Policy Agents can be downloaded from the Sun Microsystems Download Center.

In one scenario, for example, a Human Resources web server is protected remotely by Access Manager, and has policy agent installed on it. This agent prevents personnel without the proper credentials and policy from viewing confidential salary information and other sensitive data. The policies are defined by an Access Manager administrator, stored within the Access Manager deployment and used by the policy agent to allow or deny users access to the remote server's content. More information on installing and managing the policy agents can be found in the *Sun Java System Access Manager Policy Agents Guide*

Policy Types

There are two types of policies that can be configured using Access Manager: *conditional* policies and *referra*l policies. A conditional policy, also referred to as a *normal* policy, specifies two things: 1) a resource, and 2) who is allowed to access the resource. Only a Top-Level Administrator can create or manage conditional policies that apply to the whole directory. A referral policy makes it possible for a Top-Level Administrator to delegate policy configuration tasks to an Organization Administrator.

Conditional Policy

A policy that defines access permissions is a *conditional* policy, also referred to as a *normal* policy. A conditional policy consists of Rules, Subjects, and Conditions.

Rules

A rule contains a resource, one or more sets of an action, and a value. It defines the policy.

- A *resource* defines the specific object that is being protected; for instance, an HTML page or a user's salary information accessed using a human resources service.
- An *action* is the name of an operation that can be performed on the resource; examples of web server actions are POST or GET. An allowable action for a human resources service might be `canChangeHomeTelephone`.
- A *value* defines the permission for the action, i.e: allow or deny.

Subjects

A subject defines the user or collection of users (for instance, a group or those who possess a specific role) that the policy affects. Subjects are assigned to policies. The default subjects are:

- Access Manager Roles
- LDAP Groups
- LDAP Roles
- LDAP Users
- Organization

NOTE Nested roles could be evaluated correctly as LDAP Roles in the subject of a policy definition. However, the LDAP entry representing the nested role must be created using the Directory Server console. Access Manager console and SDK do not support creating nested roles.

Conditions

A condition defines the situations in which a policy is applicable; for instance, a 7 am to 10 am condition defined in a policy restrains the subject(s) to access between 7 am to 10 am.

Referral Policy

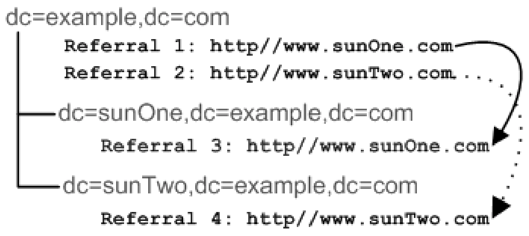
A policy used for delegation is a *referral* policy. A referral policy delegates both policy creation and policy evaluation. It consists of one or more rules and one or more referrals.

- A *rule* defines the resource whose policy creation or evaluation is being referred.
- A *referral* defines the identity object to which the policy creation or evaluation is being referred.

NOTE In most cases, a referral is an identity object although it can be any object that implements the Referral interface.

For example, consider a deployment whose root level organization is dc=example,dc=com with sub-organizations dc=sunOne,dc=example,dc=com and dc=sunTwo,dc=example,dc=com. See Figure 3-4. In order to define or evaluate policies at dc=sunOne,dc=example,dc=com or dc=sunTwo,dc=example,dc=com, two referral policies must be created. One policy points from the root level to dc=sunOne,dc=example,dc=com. One policy points from the root level to dc=sunTwo,dc=example,dc=com.

Figure 3-4 Root-level policies

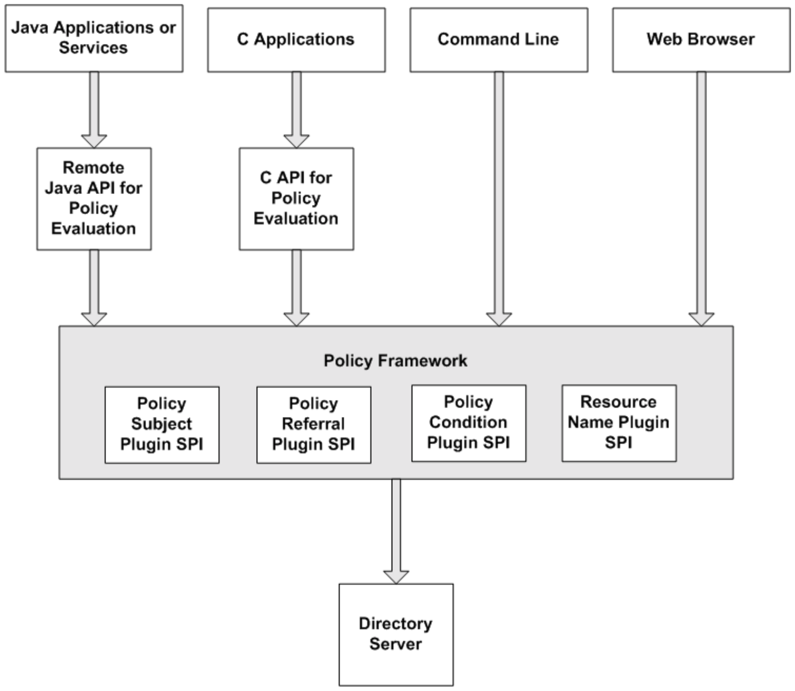

dc=example,dc=com
Referral 1: http//www.sunOne.com
Referral 2: http//www.sunTwo.com
dc=sunOne,dc=example,dc=com
Referral 3: http//www.sunOne.com
dc=sunTwo,dc=example,dc=com
Referral 4: http//www.sunTwo.com

Each referral policy contains the resource (or resource prefix) being managed. If dc=sunOne,dc=example,dc=com manages http://www.sunOne.com/, the referral policy at dc=example,dc=com contains http://www.sunOne.com/ in its rule and refers policy creation to the dc=sunOne,dc=example,dc=com sub-organization. Only after creating root level referral policies can policies at the sub-organization level be created.

Policy Management Architecture

The Policy management feature allows for the protection of all types of applications and resources. Figure 3-5 illustrates the architecture of the Policy Service. As shown, custom agents or applications can be written to protect other types of resources including services or other applications.

Figure 3-5 Policy Management Architecture


Java Applications or Services
C Applications
Command Line
Web Browser
Remote Java API for Policy Evaluation
C API for Policy Evaluation
Policy Framework
Policy Subject Plugin SPI
Policy Referral Plugin SPI
Policy Condition Plugin SPI
Resource Name Plugin SPI
Directory Server

The process for protected web resources begins when a web browser requests a URL that resides on a remote server; the server's installed policy agent intercepts the request and checks for existing authentication credentials (a session token). If none exists or the existing authentication level or policy conditions are insufficient, the request is redirected to the Authentication Service where the user must go through the authentication process.

Once the user session is created or upgraded with a successful authentication, Access Manager responds to the browser request with a redirect to the original resource. The agent now finds a sufficient session token and issues a request to the Naming Service. (The *Naming Service* defines the URLs that can be used to access Access Manager internal services.) The Naming Service returns locators for the Policy Service which the agent will use to check the user's policy, and for the Session Service which will be used to verify the user's session. Based on the aggregate of all policies assigned to the user, the individual is either allowed or denied access to the protected resource.

Single Sign-On

When a user wants to access *multiple* resources protected by Access Manager, the Session Service provides proof of authentication.This makes it possible for a user to access more than one service in a single user session without having to re-authenticate.

For example, when a user logs into his email account, he provides his username and password. Once verified, the mail service calls the Single Sign-On (SSO) API to generate an SSO or *session* token which holds the user's identity information. The API also generates a *token ID*, a random identification string associated with the session token. The session token is then sent back to the requesting browser in the form of a cookie.

When the user logs into another service within the same domain, for example the corporate phone book, the same session token is passed to the phone book service and validated, verifying the user's previous authentication. The user is granted access to the corporate phone book without being asked to re-enter his username and password. This is *single sign-on*, and the Session Service is what gives Access Manager this functionality.

Cross-Domain Single Sign-On

A user authenticated to Access Manager in one domain can access protected resources in another domain. This functionality is called cross-domain single sign-on (CDSSO). In a single-domain deployment, policy agents are configured to re-direct requests from un-authenticated users to the Authentication Service for login. In a cross-domain SSO deployment, agents are configured to re-direct requests from un-authenticated users to the Cross-Domain Controller servlet for login.

Policy Agents

A *policy agent* protects the web server or application server on which a resource lives by evaluating and enforcing a user's assigned policies. Two types of policy agents are supported by Access Manager: the web agent and the J2EE/Java agent. The web agent enforces URL-based policy, while the J2EE/Java agent enforces both J2EE-based security and URL-based policy.

Cross-Domain Controller

The Cross-Domain Controller (CDC) is a servlet that communicates with policy agents outside its own domain, and then checks for a user's SSO information. If no SSO information exists for the user, request is redirected back to servlet. If SSO information for the user is available, the servlet extracts the information and redirects the browser back to the Policy Agent. After receiving the request from the servlet, the policy agent reads the SSO information and uses it to set the cookie credentials in the foreign domain.

Figure 3-6 Single-domain SSO vs. Cross-domain SSO

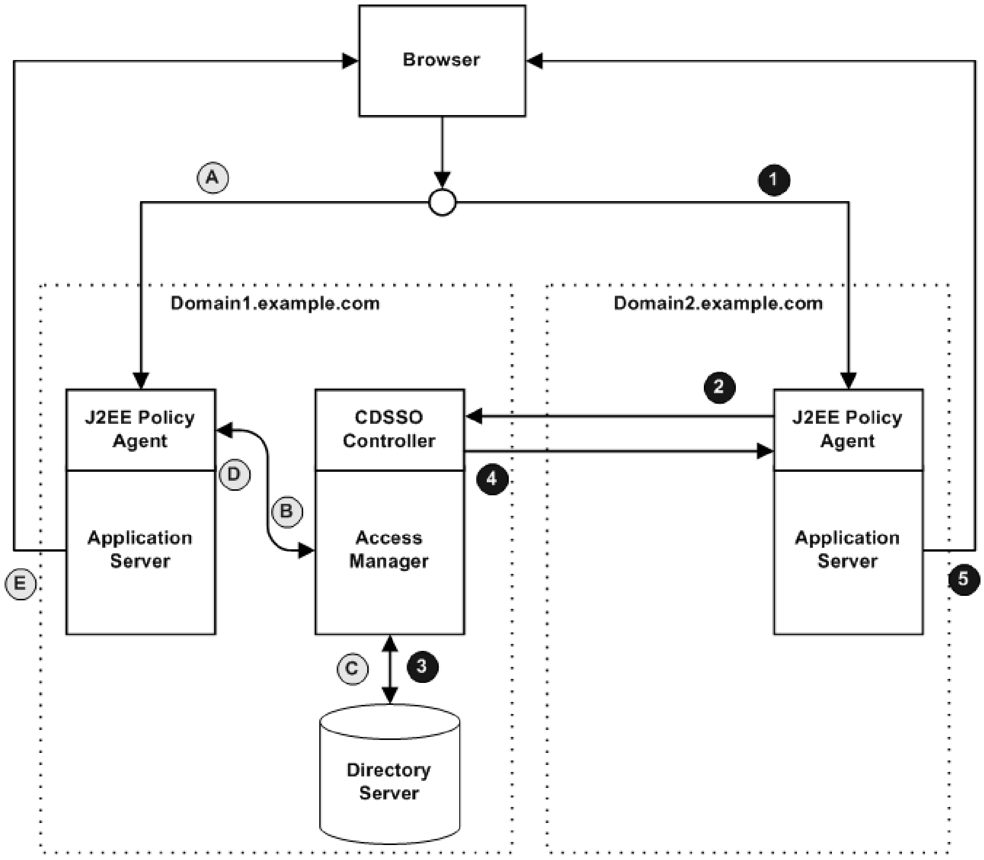


Figure 3-6 compares the sequence of SSO events in a home domain (Domain1) with the sequence of SSO events in a foreign domain (Domain2). The home domain contains Access Manager with the CDC enabled. A foreign domain is any domain other than the home domain. In this example, within Domain1, when the user tries to access a protected resource in Domain1 via a browser (A), the request is intercepted by a policy agent installed on the Application Server. The policy agent redirects the request directly to the Access Manager Authentication service (B). The Authentication service checks the user's credentials in Directory Server (C), and returns an SSO token to the policy agent (D). Once, authenticated, the Application Server displays the protected web content in the browser (E).

In a cross-domain scenario, when the user tries to access a protected resource in Domain2 via a browser (1), the request is intercepted by a policy agent installed on the Application Server. The policy agent redirects the request to cross-domain controller which resides with Access Manager in Domain 1 (2). If SSO information for the user is available in Directory Server (3) the servlet extracts the information and redirects it back to the Domain2 policy agent (4). After receiving the request and the SSO information from the servlet, the Domain2 policy agent reads the SSO information and uses it to set the cookie in Domain1 for authentication purposes. Once the user is authenticated, the Application Server displays the protected web content in the browser (5).

Chapter 4

Services Management

Sun Java™ System Access Manager 6 2005Q1 comes with its own set of core services which make user access and policy management possible. Access Manager also provides the necessary tools for administrators to define, integrate and manage groups of attributes which form service plug-ins. Both eXtensible Markup Language (XML) files and Java™ interfaces are used for this purpose.

This chapter describes how Access Manager core services and service plug-ins work together. Topics included in this chapter are:

- "How Services Work in Access Manager" on page 55
- "Attribute Types" on page 57
- "Identity Server Core Services" on page 58
- "The Service Configuration Interface" on page 62

How Services Work in Access Manager

In Access Manager, a *service* is a group of attributes that are managed together by the Access Manager console. The attributes can be just bits of related information such as an employee's name, job title, and email address. But attributes are typically used as configuration parameters for a software module such as a mail application or payroll service.

A company's mail service provides a good example of how a service plug-in can extend the Access Manager core services. In this example, a company stores in its Directory Server one user profile for each of its employees. Now the company wants to leverage those user profiles to work with its mail application. First, the developer creates a *service definition* in the form of an XML file. The XML file describes the object classes and attributes that will be used by the mail application. Mail-related attributes might include the employee’s email address and password, the name of the mail server he uses, his email storage limit, time-out limit, and so forth.

Core Services

Access Manager comes with a number of services that work together to perform its basic functions. See “Identity Server Core Services” on page 58 for a complete services listing. The following example illustrates how Authorization, Session, Policy, Naming and Logging Services work together.

When the employee logs in to a mail application, the application first communicates with Access Manager to *authenticate*, or verify, the employee’s user ID and password. The Authentication Service verifies that the login credentials the user has provides matches the user profile stored in Directory Server.

Access Manager also issues a request to the Naming Service which defines the URLs used to access other Access Manager services. The Naming Service returns locators for the Policy Service which is used to examine all policies assigned to the user.

Once the user is authenticated and granted access to a resource, Access Manager issues a *session token* or marker that helps to keep track of things such as when the user logged in and how long it’s been since the user stopped actively using the application. The Logging Service records information such as user activity, traffic patterns, and authorization violations. These records are useful when troubleshooting.

Service Plug-Ins

In the same example, the mail application works with the APIs in the Access Manager Software Development Kit (SDK) layer. Access Manager validates the attribute values associated with the user’s profile against the mail service definition. This helps the mail application to determine which mail server the employee is authorized to use, whether the employee is within his email storage limit, whether he’s exceeded his time-out limit, and so forth. Based on each of these determinations, the mail application either grants or denies the employee access to his email. If the employee is granted access, the Access Manager session management service comes into play. When the employee’s mail session sits idle for too long, for example, Access Manager communicates with the mail application which causes the mail session to time-out or expire.

The Access Manager SDK gives application developers the interfaces necessary to register and un-register services with Access Manager, as well as to manage service attributes and configuration information. With one centralized repository for user profiles, and one SDK, it's possible for multiple services or applications to leverage the same user data.

Attribute Types

The attributes that make up an Identity Server service are classified as one of the following types: *Dynamic, Policy, User, Organization* or *Global*. Using these types to subdivide the attributes in each service allows for a more consistent arrangement of the service schema and easier management of the service parameters.

Dynamic Attributes

A dynamic attribute can be assigned to an Identity Server configured role or organization. When the role is assigned to a user or a user is created in an organization, the dynamic attribute then becomes a characteristic of the user. For example, a role is created for an organization's employees. This role might contain the organization's address and a fax number, two things that remain static for all employees. When the role is assigned to each employee, these dynamic attributes are inherited by each employee.

User Attributes

These attributes are assigned directly to each user. They are not inherited from a role or an organization and, typically, are different for each user. Examples of user attributes include `userid`, `employee number` and `password`. User attributes can be added or removed from the User service by modifying the `amUser.xml` file. For more information, see the *Access Manager 6 2005Q1 Developer's Guide*.

Organization Attributes

Organization attributes are only assigned to organizations. No object classes are associated with organization attributes. Attributes listed in the authentication services are defined as organization attributes because authentication is done at the organization level rather than at a subtree or user level.

Global Attributes

Global attributes are applied across the Identity Server configuration. They cannot be applied to users, roles or organizations as the goal of global attributes is to customize the Identity Server application. There is only one instance of a global attribute in the Identity Server configuration. There are no object classes associated with global attributes. Examples of global attributes include log file size, log file location, port number or a server URL that Identity Server can use to access data.

Policy Attributes

Policy attributes specify the access control actions (or privileges) associated with a service. They become a part of the rules when rules are added to a policy. Examples include `canForwardEmailAddress` and `canChangeSalaryInformation`. The actions specified by these attributes can be associated with a specific resource. See "Policy Framework" on page 46 for related information.

Identity Server Core Services

Default services are provided with Identity Server and are defined by XML files located in the following directory:

```
/etc/opt/SUNWam/config/xml
```

Some of these services, when configured through the Service Configuration interface, define values that are applied across the Identity Server application. Others are registered to a specific organization configured within Identity Server and are used to define default values for the organization.

Administration

The Administration service allows for the configuration of the console at both the application level (similar to a *Preferences* or *Options* menu for the Identity Server application) as well as at a configured organization level (*Preferences* or *Options* specific to a configured organization).

Authentication

There are a number of Access Manager authentication modules including a base module. This allows the administrator the opportunity to choose the method each defined organization uses to verify users' identities.

Anonymous. This module allows for log in without specifying a user name and password. Anonymous connections have limited access to the server and are customized by the administrator.

Certificate-based. This module allows a user to log in through a personal digital certificate (PDC). The module comes with an Online Certificate Status Protocol (OCSP) which can determine the state of a certificate.

Core. This module is the general configuration base for the Identity Server authentication services. It must be registered and configured to use any of the specific services. It allows the administrator to define default values that will be picked up for those not specifically set in the Anonymous, Certificate-based, HTTPBasic, LDAP, Membership, RADIUS, SafeWord, SecurID, Windows, and Unix services.

HTTPBasic. This module uses basic authentication, which is the HTTP protocol's built-in authentication support. The Web Server issues a client request for username and password, and sends that information back to the server as part of the authorized request. Access Manager retrieves the username and password and then internally authenticates the user to the LDAP authentication module. The HTTPBasic module internally initializes the LDAP authentication module attributes. In order for HTTPBasic to function correctly, the LDAP authentication module must be configured (registering the HTTPBasic module alone will not work).

Kerberos. This module allows a client or user *who has already authenticated by a Kerberos Distribution Center (KDC)* to be authenticated by Access Manager without having to provide the login information again. This is also known as desktop single sign-on.

LDAP. This module allows for authentication using LDAP bind, an operation which associates a password with a particular LDAP entry.

Membership (Self-Registration). This module allows a new user to self-register for authentication with a login and password.

NT. This module allows for authenticating users using a Windows 2000™ server.

RADIUS. This module allows for authenticating users using an external Remote Authentication Dial-In User Service (RADIUS) server.

SafeWord. This module allows for authenticating users using Secure Computing's SafeWord™ or SafeWord PremierAccess™ authentication servers.

SecurID. This module allows for authenticating users using RSA's ACE/Server authentication server. Note that SecurID is not supported on the Solaris x86 platform.

UNIX. This module allows for authenticating users using the Access Manager UNIX service. Note that UNIX authentication service is not supported on the Windows 2000 platform.

Authentication Configuration

The Authentication Configuration service allows you to configure authentication on for roles, users and services and organizations to set the rules determining the precedence of the authentication modules.

Client Detection

The Client Detection service defines attributes to detect the type of client being used, and performs actions based on client type.

Logging

The Logging service provides status and error messages related to Access Manager administration. An administrator can configures values such as log file size and log file location. Access Manager can record events in flat text files or in a relational database.

Naming

The Naming service is used to get and set URLs, plug-ins and configurations as well as request notifications for various other Identity Server services such as session, authentication and logging.

Password Reset

Access Manager provides a Password Reset service to allow users to receive via email a new password, or to reset their password, for access to a given service or application protected by Access Manager.

Platform

The Platform service is where additional servers can be added to the Identity Server configuration as well as other options applied at the top level of the Identity Server application.

Policy Configuration

Policy Configuration defines user privileges to web resources, allowing an administrator to allow or deny access to `http` and `https`-based URLs.

SAML

The Security Assertion Markup Language (SAML) service defines a framework for exchanging security assertions among security authorities. This makes interoperability across different platforms possible, enabling authentication and authorization, and attribute services.

Session

The Session service defines values for an authenticated user session such as maximum session time and maximum idle time.

User

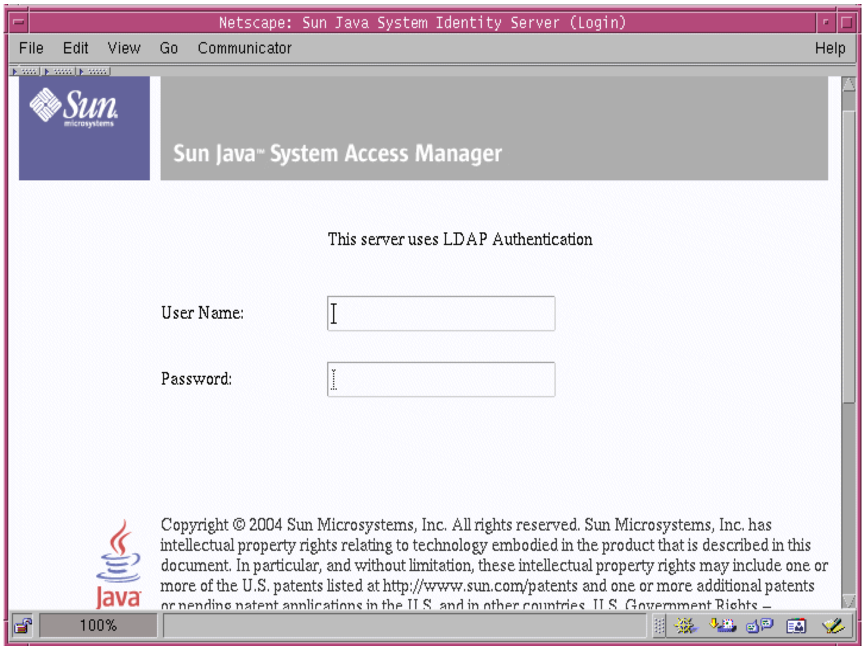
Default user preferences are defined through the user service. (These include time zone, locale and DN starting view).

The Service Configuration Interface

Services are configured and managed through the Service Configuration module. Organization-specific services which are not covered by the Identity Server default service packages can be written using XML (based on the Identity Server services document type definition or DTD) and added into the interface under the Other Configuration heading.

The Service Configuration module is for displaying service configurations on a global level. In other words, it is a view of the default configurations of all available services in Identity Server, whether registered or not. When a service is registered and activated by an organization, the initial default data assigned to the service is displayed under the service’s Service Configuration page. Figure 4-1 is a screenshot of the graphical user interface.

Figure 4-1 Service Configuration View


Netscape: Sun Java System Identity Server (Login)
File
Edit
View
Go
Communicator
Help
Sun microsystems
Sun Java™ System Access Manager
User Name:
Password:
Java
100%

This server uses LDAP Authentication

Copyright © 2004 Sun Microsystems, Inc. All rights reserved. Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and in other countries. U.S. Government Rights –

Chapter 5

Federation Management

This chapter explains the concept of identity federation, and describes the role of the Federation Management module in Sun™ Java System Access Manager 6 2005Q1.

- "The Need for Federated Identities" on page 63
- "The Liberty Alliance Project" on page 64
- "The Circle of Trust" on page 65
- "Federation Management Architecture" on page 67
- "The Federation Management Process" on page 71

The Need for Federated Identities

Consider the many times an individual accesses services on the Internet in a single day. At work, he uses the company intranet to perform a multitude of business-related tasks such as reading and sending email, looking up information in the company phone book and other internal databases, and submitting expense reports and other business-related online forms. At home after work, he checks his personal email, then logs into an online news service to check his baseball team's standings. He may finalize his travel plans via his travel agent's website, and then does some online shopping at his favorite clothing store. Each time he accesses a service on the Internet, he must log in and identify himself to the service provider.

A *local identity* refers to the set of attributes or information that identify a user to a particular service provider. These attributes typically include a name and password, plus an email address, account number or other identifier. For example, the individual in our scenario is known to his company's network as an employee number, but he is known to his travel agent as Joe Smith. He is known to his online news service by an account number, and he is known to his favorite clothing store by a different account number. He uses one email name and address for his personal email, and a different email name and address for his workplace. Each of these different user names represents a different local identity.

Identity federation allows a user to consolidate the many local identities he has configured among multiple service providers. With one *federated identity*, the individual can log in at one service provider's site and move to an affiliated service provider site without having to re-authenticate or re-establish his identity. For example, with a federated identity, the individual might want to access both his personal email account and his business email account from his workplace, and move back and forth between the two services without having to log in each time. Or at home he might want to log in to an online clothing store, then access the online news service, and communicate with his travel agent. It is a convenience for the user to be able to access all of these services without having to provide different user names and passwords at each service site. It is a valuable benefit to the user when he can do so safely, and knowing that his identity information is secure.

The Liberty Alliance Project was implemented to make this possible.

The Liberty Alliance Project

In 2001 Sun Microsystems joined with other major companies to form the Liberty Alliance Project, the premier open standards organization for federated identity and identity-based services. The Liberty Alliance Project develops specifications and guidelines for implementing complete network identity infrastructures and for deploying identity-based web services.

The members of the Liberty Alliance Project represent some of the world's most recognized brand names and service providers, driving products, services and partnerships across a spectrum of consumer and industrial products, financial services, travel, retailing, telecommunications and technology. For more information including listings of Liberty web service products, specifications, cased studies, and white papers, see the Liberty Alliance Project website:

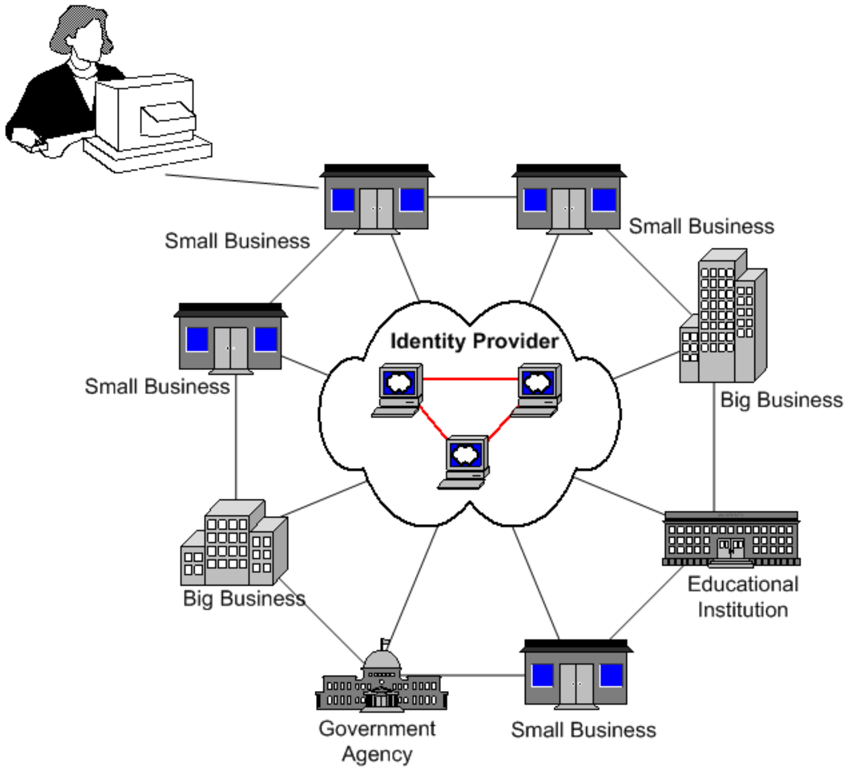
`http://projectliberty.org/`

The Circle of Trust

The goal of the Liberty Alliance Project is to enable individuals and organizations to easily conduct network transactions while protecting the individual's identity. This goal can be achieved only when commercial and non-commercial organizations join together into a *circle of trust*. In a circle of trust, service providers agree to join together in order to exchange user authentication information using Liberty web service technologies. This circle of trust must contain at least one identity provider, a service that maintains and manages identity information. The circle of trust also includes service providers that offer web-based services to users; it also includes users themselves. Once a Circle Of Trust is established, single sign-on is enabled between all the providers.

The circle of trust is also known as an *authentication domain*, although it is not a DNS domain or a domain in the Internet sense of the word. Figure 5-1 illustrates a user accessing a small business, a service provider that is associated with other businesses and agencies in a circle of trust.

Figure 5-1 The Circle of Trust



Account federation occurs when a user chooses to unite distinct service accounts and identity provider accounts. The user retains individual account information with each provider in the circle. At the same time, the user establishes a link that allows the exchange of authentication information between them. Users can choose to federate any or all identities they might have with the service providers that have joined this circle. When a user successfully authenticates with one service provider, she can access any of the her accounts within the circle of trust in a single session *without having to reauthenticate.*

Federation Management Architecture

The Access Manager 6.1 release implemented an Identity Federation Framework (ID-FF) version 1.1 Access Manager 2005Q1 adds new federation features defined in Identity Federation Framework 1.2 specifications, including a web service framework to facilitate deployment of customer Identity Web Services. Client APIs are provided for web service consumers to communicate with web service providers.

The following three major components make it possible for identity information to be exchanged among service providers:

- Identity Federation Framework
- Identity Web Services Framework
- Identity Service Instance Specifications (ID-SIS)

Identity Federation Framework

The Identity Service Instance Specifications build upon the Web Service Framework and Federation Framework to provide services. The federation framework specifies core protocols, schema and concrete profiles that allow developers to create a standardized, multiple-vender, identity federation network. These include the following:

Opt-in account linking. Users can choose to federate different service provider accounts.

Authentication context. Service providers with federated accounts communicate the type and level of authentication that should be used when the user logs in.

Account linking termination. Users can choose to stop their account federation.

Identity provider introduction. This feature provides the means for service providers to discover which identity providers a *principal* uses. A principal can be an organization or individual who interacts with the system. This is important when there are multiple identity providers in an identity federation network.

Name Registration. This feature enables a service provider or identity provider to register with each other a new name identifier for a principal at any time following federation.

Single Sign-on and Federation Protocol TA protocol that defines the process that a user at a service provider goes through to authenticate their identity with an identity provider. It also specifies the means by which a service provider obtains an Authentication Assertion from an identity provider to allow single sign-on to the user. There are two types of Single Sign-On which either the identity or service provider can implement:

- SOAP-based Single Sign On and Federation Protocol, which relies on a SOAP call from provider to provider. This is primarily the Browser Artifact SSO profile.
- Form POST-based Single Sign On and Federation Protocol, which rely on an HTTP form POST to communicate between providers.

Single Log-Out Protocol. TA protocol used to synchronize the session log-out functionality across all sessions that were authenticated and created by a particular identity provider. There are two types which either the identity or service provider can implement:

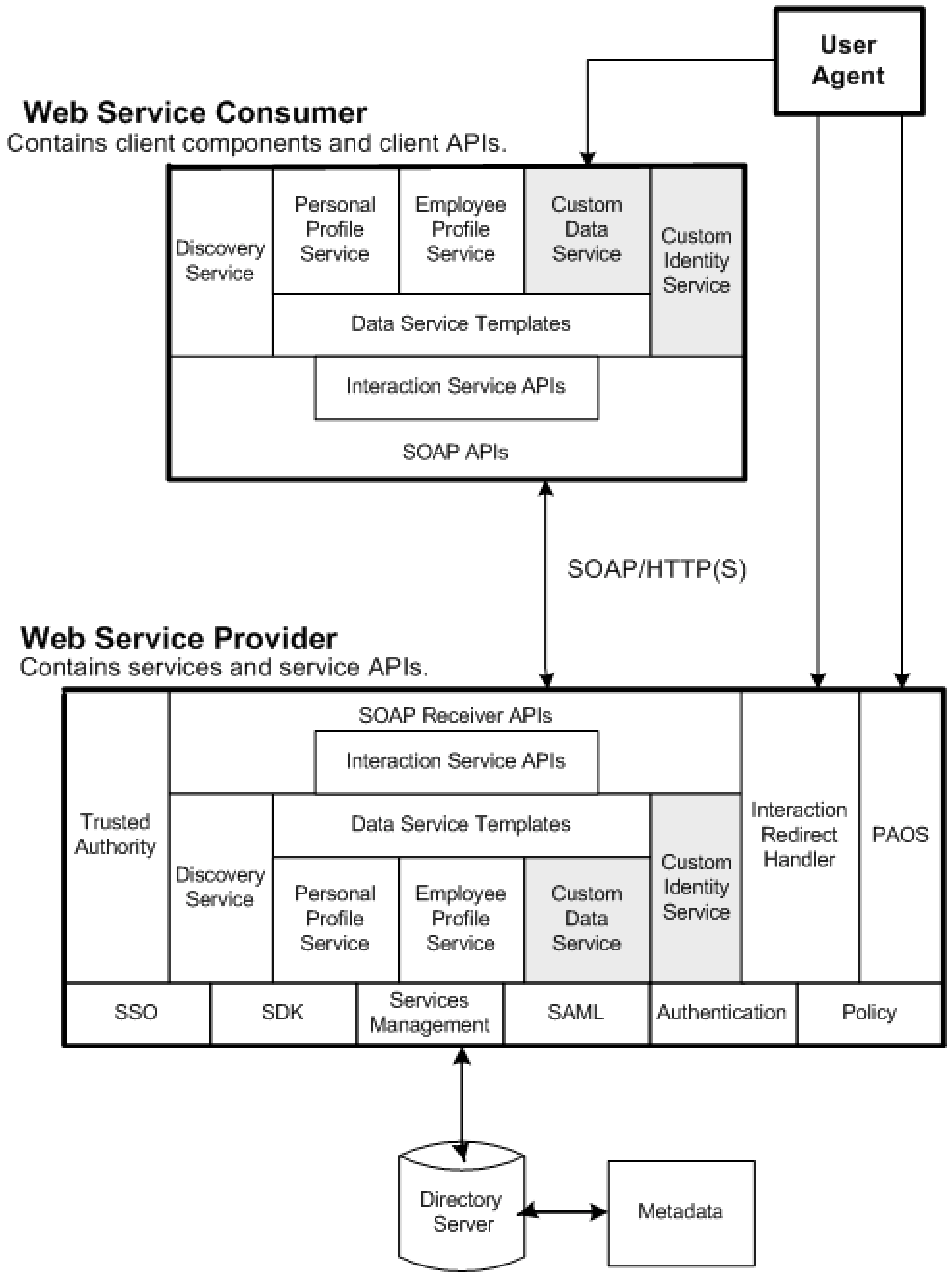
- SOAP-based Single Log-Out Protocol relies on asynchronous SOAP messaging calls between providers.
- HTTP Redirect-based Single Log-Out Protocol

Identity Web Services Framework

The Web Services Framework consists of a set of schema, protocols and profiles for providing a basic identity services, such as identity service discovery and invocation.

Three parties are required for identity federation in a basic Liberty Web Services environment: a user agent, a web service consumer, and a web service provider. Figure 5-2 illustrates the internal architectures of a Liberty Web Services Consumer and a Web Service Provider.

Figure 5-2 Liberty II Architecture.



The Web Service Consumer components and the Web Service Provider components are newly implemented components in Access Manager 2005Q1. The components in the bottom layer of the Web Service Provider were implemented in Access Manager 6.1. These components include Single-Sign On (SS0), the Access Manager SDK, Service Management Services, SAML, Authentication modules, and a Policy Service. In Figure 5-2, the Data Service and Identity Service represent custom services that you can add to the Web Services Framework.

The Web Services Framework consists of a set of schema, protocols and profiles for providing a basic identity services, such as identity service discovery and invocation. This framework is new in Liberty II and includes the following:

Discovery Service. An identity service that allows a requester to discover resource offerings.

SOAP Binding. A set of Java APIs for sending and receiving ID-* messages using SOAP and XML.

Data Services Template. A common data service layer for developing identity services. Includes common utilities for message error-checking and verification, and remote client APIs to support customer data service instances.

Security Mechanisms. Defines a set of authentication mechanism and security properties which are factored into authorization decisions enforced by the targeting identity-based web services. Each mechanism contains both peer entity authentication (null / TLS / CClientTLS) and message authentication (null / X509 / SAML).

Interaction Service. A protocol for simple interaction of Web Services Framework participants with a Principal.

Trusted Authority. APIs for creating security tokens used for authentication and authorization in Liberty II-enabled services.

Identity Service Instance Specifications (ID-SIS)

The Liberty Service Instance Specifications build upon the Web Services Framework and Federation Framework to provide services such as contacts, presence detection or wallet services that depend on network identity. The following Service Instance Specifications implementations are new in Access Manager 2005Q1.

Personal Profile Service. An instance of a data-oriented web service which offers profile information regarding a principal's personal life. A shopping portal that offers information such the principal's account number and shopping preferences is an example of a personal profile service.

Employee Profile Service. Similar to the Personal Profile Service, except that An instance of a data-oriented web service which offers profile information regarding a principal's workplace. An online corporate phone book that provides an employee name, office building location, and telephone extension number is an example of an employee profile service.

Supporting Components

Metadata Service. A library of command-line tools for loading metadata into the Access Manager data store.

Reverse HTTP Bindings. A protocol and set of APIs for retrieving data from Access Manager via clients such as cell phones.

The Federation Management Process

The Federation Management process begins with authentication. By default, Access Manager comes with two options for user authentication. The first is the proprietary Authentication Service; the second is the Liberty-enabled Federation Management module. With the first option, when a user tries to access a resource protected by Access Manager, he is redirected to the Authentication Service using an Access Manager login page. When the user provides credentials, the Authentication Service verifies his identity, and either allows or denies access.

With Liberty-enabled Federation Management, when a user attempts to access a resource protected by Access Manager, the authentication process begins with a search for a valid Access Manager session token. If a session token is found, the user is granted access to the requested page. The requested page, which belongs to a member of the circle of trust, contains a link which provides the user an opportunity to federate his identity. When the user clicks this link, he is directed through the Federation Single Sign-On Process. If no session token is found, the user is directed through the Pre-Login Process.

Federation Single Sign-On Process

When a user signs on to access a protected resource or service, Access Manager sends a request to the identity provider for authentication confirmation. If the identity provider sends a positive response, the user gains access to all provider sites within the circle of trust. If the identity provider sends a negative response, the user is directed to authenticate again using the Federation Single Sign-On process.

In federated single sign-on, the user selects an identity provider and then sends his or her credentials for authentication. Once authentication is complete and access is granted, the user is redirected to the Access Manager Authentication Service. The user is automatically granted a session token and redirected to the requested page which contains a link to allow the user to federate his or her identity. As long as the session token remains valid, the user can access other services offered by other service providers in the circle of trust without having to sign on again.

Pre-Login Process

The purpose of the Pre-Login process is to establish a valid user session at the service provider site. When no Access Manager session token is found, the pre-login process begins with the search for another type of cookie, a Federation cookie.

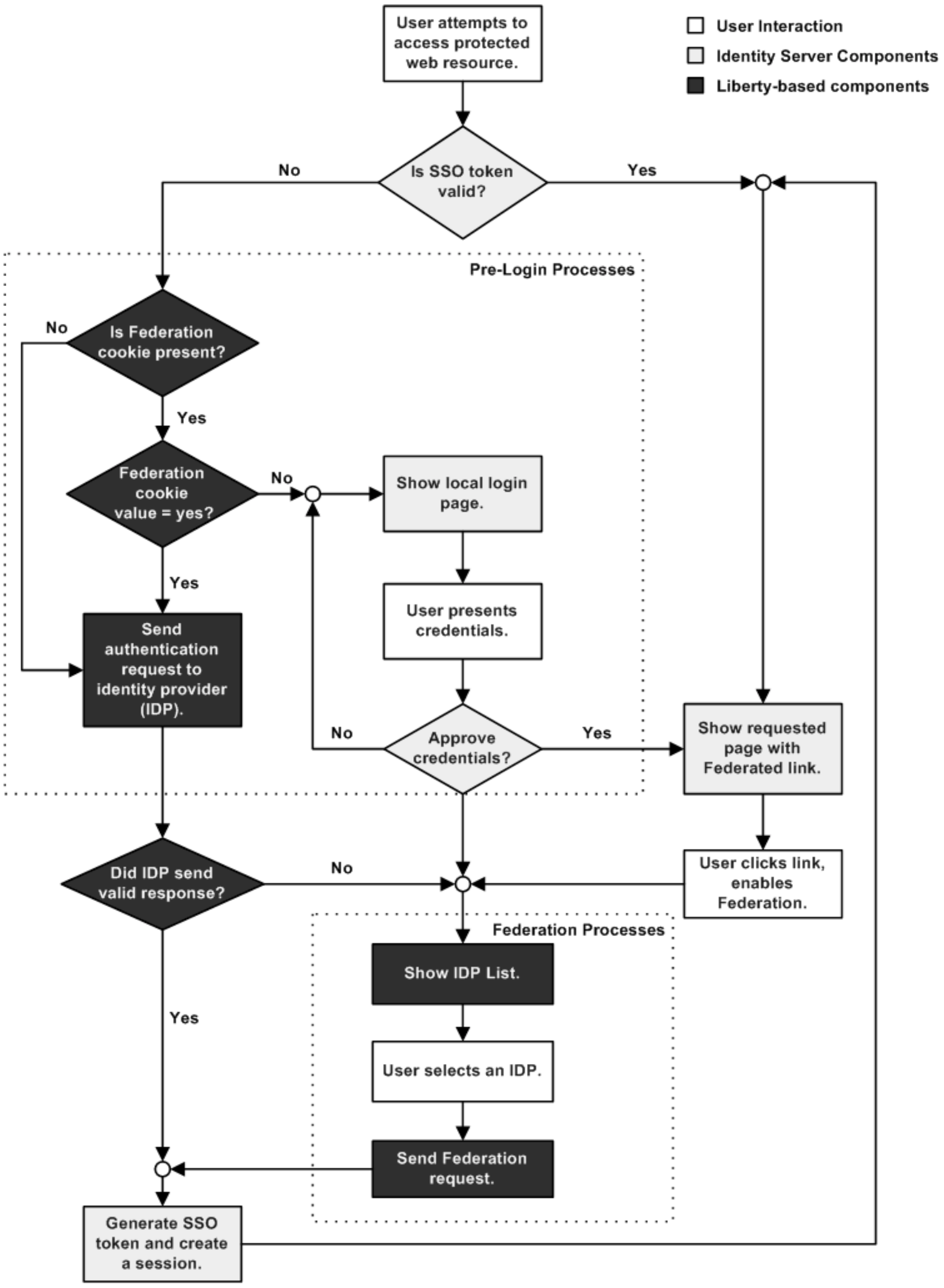
If, after the search for an Access Manager session token proves null, a Federation cookie is found and its value is “no,” an Access Manager login page is displayed and the user submits credentials to the Authentication Service. When authenticated by the Access Manager, the user is redirected to the requested page which contains a link to allow the user to federate their identity. If the user clicks this link, he is directed through the Federation Single Sign-On Process.

If, after the search for an Access Manager session token proves null, a valid Federation Cookie is found an its value is “yes,” it means the user has already been federated but not authenticated by an identity provider within the Circle of Trust. This is confirmed by sending a request for authentication to the user’s chosen identity provider.

If no Federation cookie is found at all, a passive authentication request is sent to the user's chosen identity provider. A passive authentication request does not allow identity provider interaction with the user. When an affirmative response is received back from the identity provider, the user is redirected to the Access Manager Authentication Service. There, the user is granted a session token and redirected to the requested page. When the response from the identity provider is negative (for example, if the session has timed out), the user is sent to a common login page where he can choose to do a local login or Federation Single Sign-On.

Figure 5-3 illustrates the differences between the Pre-Login process path and the Identity Federation path.

Figure 5-3 Liberty-enabled Access Manager Authentication Process Flow

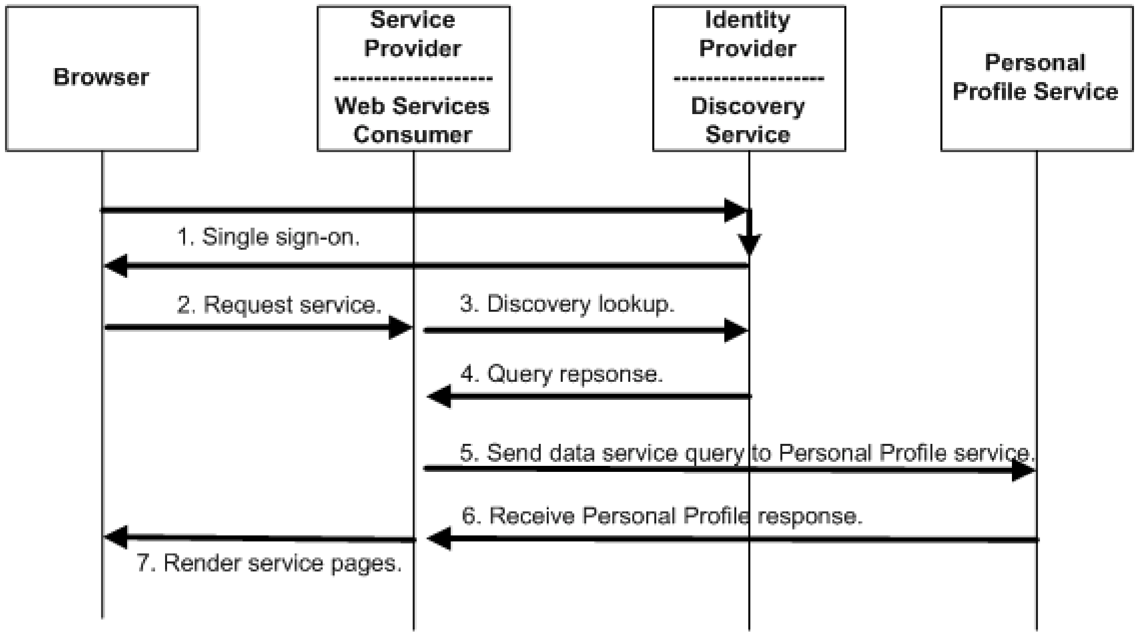

User attempts to access protected web resource.
User Interaction
Identity Server Components
Liberty-based components
Is SSO token valid?
No
Yes
Pre-Login Processes
Is Federation cookie present?
No
Yes
Federation cookie value = yes?
No
Yes
Show local login page.
User presents credentials.
Send authentication request to identity provider (IDP).
Approve credentials?
No
Yes
Show requested page with Federated link.
Did IDP send valid response?
No
User clicks link, enables Federation.
Federation Processes
Show IDP List.
Yes
User selects an IDP.
Send Federation request.
Generate SSO token and create a session.

System Flow

Figure 5-4 provides a high-level view of the system flow between various parties in a Liberty web services environment. In this figure, note the following:

- The browser represents a *user agent*, a device used by an enterprise user.
- The Service Provider also acts as a Web Service Consumer.
- The Identity Provider hosts the Discovery Server.
- The Personal Profile Service represents a Web Service Provider.

Figure 5-4 The interaction between Liberty II components.



This is what happens on the back end when an employee looks up a colleague’s phone number in an online corporate phone book:

1. The user's browser, Service Provider and Identity Provider complete the Federation Single-Sign-On process.

   An assertion with an attribute statement containing the Discovery Service *resource offering* will be included in the `ID-FF AuthnResponse`. This information, is used by any client to contact Discovery Service.

2. The user's browser requests access to services hosted on the Web Service Consumer.

   This requires contacting user's Personal Profile service.

3. The Web Service Consumer sends a discovery lookup query to the Discovery Service.

   The Web Service Consumer determines user's discovery resource offering from the bootstrap Assertion obtained in Step 1, then sends a discovery lookup query to the Discovery Service to determine where the user's Personal Profile instance is hosted.

4. The Discovery service returns a discovery lookup response to the Web Service Consumer.

   The lookup response contains the resource offering for the user's Personal Profile Service instance.

5. The Web Service Consumer sends a Data Services Template query to the SOAP end point of the Personal Profile Service instance.

   The query asks for the user's personal profile attributes, such as home phone number. The required authentication mechanism specified in the Personal Profile Service resource offering must be followed.

6. The Personal Profile Service instance authenticates and validates authorization or policy, or both, for the requested user or Web Service Consumer, or for both.

   If user interaction is required for some attributes, the Interaction Service will be invoked to query the user for consents or for attribute values. The Personal Profile Service instance returns a Data Services Template response to the Web Service Consumer after collecting all required data.

7. The Web Service Consumer processes the Personal Profile Service response, and then renders service pages containing the colleague’s contact information to the user's browser.

Glossary

Refer to the Java Enterprise System glossary for a complete list of terms that are used in this documentation set.

`http://docs.sun.com/source/816-6873`

Index

A

account locking 41
- memory 42
- physical 41

administration service 58

anonymous authentication 59

anonymous authentication module 36

APIs
- federation management 74
- Identity Server components 21

architecture
- policy service 49

attributes
- dynamic 57
- global 58
- organization 57
- policy 58
- user 57

authentication
- account locking 41
  - memory 42
  - physical 41
- authentication context 67
- domain 65
- FQDN mapping 42
- LDAP authentication 59
- methods 40
- module chaining 42
- modules
  - anonymous 36
  - authentication configuration 40
  - HTTP Basic 37
  - LDAP 38
  - membership 38
  - NT 39
  - SafeWord 38, 39
- persistent cookies 43
- session upgrade 43
- user interface 41
- validation plug-in interface 43

authentication configuration service 40

authentication modules
- anonymous 36
- authentication configuration 40
- HTTP Basic 37
- LDAP 38
- membership 38
- NT 39
- SafeWord 38, 39
- Unix 39

C

certificate-based authentication 59

circle of trust 65

client detection 60

console
- user interface 41

containers 29

core authentication 59

D

Data Services Template 70
directory information tree 25
DIT. See directory information tree.
domain component 26
dynamic attributes 57

E

Employee Profile Service 71

F

federation 66
Federation Framework 67
federation management
  circle of trust 65
  Liberty Alliance Project 64
  local identity 63
  overview 63
  pre-login process 72
  protocols and APIs 74
  trusted authority 70
filtered roles 28
FQDN mapping
  and authentication 42

G

global attributes 58
group containers 30
groups 27, 29

H

HTTP Basic authentication module 37
HTTPBasic authentication 59

I

identity
  defined 25
  identity federation 64
  local identity 63
  overview of 16
Identity Management
  containers 29
  group containers 30
  groups 29
  people containers 30
  policies 29
  roles 29
  services 29
  users 29
identity provider 67
  identity provider introduction 79
Interaction Service 70

L

LDAP authentication 59
LDAP authentication module 38
Liberty Alliance Project
  account linking termination 67
  authentication context 67
  data services template 70
  discovery service 70
  employee profile service 71
  identity provider 67
  identity provider introduction 79
  interaction service 70
  local identity 63
  metadata service 71
  name registration 67
  opt-in account linking 67
  personal profile service 71
  pre-login process 72
  reverse HTTP bindings 71

security mechanisms 70
single log-out protocol 68
single sign-on 72
SOAP Binding 70
trusted authority 70
Linux, default base directory for 9
local identity 63
logging service 60

M

managed roles 28
membership authentication 59
membership authentication module 38
Metadata Service 71
methods
authentication 40
module chaining
and authentication 42

N

name registration 67
naming service
defined 60
used in example 50
nested roles 28
normal policy 47
NT authentication 59
NT authentication module 39

O

opt-in account linking 67
organization attributes 57
organizational unit 26
overview
authentication
user interface 41

P

password reset 61
people containers 30
Persistent 43
persistent cookies
and authentication 43
Personal Profile Service 71
platform service 61
policy
attributes 58
normal 47
referral 48
policy configuration service 61
policy service
architecture 49
defined 46
policy types
normal 47
referral 48

R

RADIUS authentication 60
referral policy 48
Reverse HTTP Bindings 71
roles 28, 29
filtered 28
managed 28
nested 28
root suffix 25

S

SafeWord authentication 60
SafeWord authentication module 38, 39
SAML service 61

SecurID authentication 60
Security Mechanisms 70
self-registration service 59
service configuration
  interface 62
  service configuration module 62
services
  default services
    administration 58
    authentication
      anonymous 59
      authentication configuration 60
      certificate-based 59
      core service 59
      HTTPBasic 59
      Kerberos 59
      LDAP 59
      membership 59
      NT 59
      RADIUS 60
      SafeWord 60
      SecurID 60
      UNIX 60
    client detection 60
    logging 60
    naming 60
    overview 58
    password reset 61
    platform 61
    policy configuration 61
    SAML 61
    self-registration 59
    session 61
    user 61
  defined 29
session service 61
session upgrade
  and authentication 43
single sign-on 72
SOAP Binding 70
static groups 27

T

trusted authority 70

U

UNIX authentication 60
Unix authentication module 39
user attributes 57
user id 26
user interface 41
user service 61
users 29

V

validation plug-in interface
  and authentication 43