Sun Java™ System

Identity Server Technical Overview

2004Q2

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

Part No: 817-5706-10

Contents

About This Book

This *Technical Overview* provides a high-level overview of how Sun Java™ System Identity Server components work together to consolidate identity management and to protect enterprise assets and web-based applications. It explains basic Identity Server 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 Identity Server documentation.

Audience for This Guide

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)

Because Sun Java System Directory Server is used as the data store in an Identity Server deployment, administrators should also be familiar with the documentation provided with that product. The latest Directory Server documentation can be accessed online.

Identity Server 2004Q2 Documentation Set

The Identity Server 2004Q2 documentation includes two sets:

- Identity Server 2004Q2 Core Documentation
- Identity Server Policy Agent Documentation

Identity Server 2004Q2 Core Documentation

The Identity Server 2004Q2 documentation set contains the following titles:

- *Technical Overview* (`http://docs.sun.com/doc/817-5706`) provides a high-level overview of how Identity Server components work together to consolidate identity management and to protect enterprise assets and web-based applications. It also explains basic Identity Server concepts and terminology.
- *Migration Guide* (`http://docs.sun.com/doc/817-5708`) provides details on how to migrate existing data and Sun Java System product deployments to the latest version of Identity Server. (For instructions about installing Identity Server and other products, see the *Sun Java Enterprise System 2004Q2 Installation Guide* (`http://docs.sun.com/doc/817-5760`).
- *Administration Guide* (`http://docs.sun.com/doc/817-5709`) describes how to use the Identity Server console as well as manage user and service data via the command line.
- *Deployment Planning Guide* (`http://docs.sun.com/doc/817-5707`) provides information on planning an Identity Server deployment within an existing information technology infrastructure.
- *Developer's Guide* (`http://docs.sun.com/doc/817-5710`) offers information on how to customize Identity Server and integrate its functionality into an organization's current technical infrastructure. It also contains details about the programmatic aspects of the product and its API.

- *Developer's Reference* (http://docs.sun.com/doc/817-5711) provides summaries of data types, structures, and functions that make up the public Identity Server C APIs.
- *Federation Management Guide* (http://docs.sun.com/doc/817-6362) provides information about Federation Management, which is based on the Liberty Alliance Project.
- The *Release Notes* (http://docs.sun.com/doc/817-5712) will be available online after the product is released. They gather an assortment of 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.

Updates to the *Release Notes* and links to modifications of the core documentation can be found on the Identity Server page at the Sun Java System 2004Q2 documentation web site (http://docs.sun.com/prod/entsys.04q2). Updated documents will be marked with a revision date.

Identity Server Policy Agent Documentation

Policy agents for Identity Server documents are available on this Web site:

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

Policy agents for Identity Server 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 Identity Server documentation. The following titles are included in the set:

- *Web Policy Agents Guide* documents how to install and configure an Identity Server 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 Identity Server policy agent that can protect a variety of hosted J2EE applications. It also includes troubleshooting and information specific to each agent.
- The *Release Notes* will be available online after the set of agents is released. There is generally one *Release Notes* file for each agent type release. The *Release Notes* gather an assortment of 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.

Updates to the *Release Notes* and modifications to the policy agent documentation can be found on the Policy Agents page at the Sun Java System documentation web site. Updated documents will be marked with a revision date.

Your Feedback on the Documentation

Sun Microsystems and the Identity Server technical writers are interested in improving this documentation and welcomes your comments and suggestions. Use the following web-based form to provide feedback to us:

`http://www.sun.com/hwdocs/feedback/`

Please provide the full document title and part number in the appropriate fields. The part number can be found on the title page of the book or at the top of the document, and is usually a seven or nine digit number. For example, the part number of the Technical Overview is 816-5706-10.

Documentation Conventions Used in This Guide

In the Identity Server documentation, certain typographic conventions and terminology are used. These conventions are described in the following sections.

Typographic Conventions

This book uses the following typographic conventions:

- *Italic type* is used within text for book titles, new terminology, emphasis, and words used in the literal sense.
- `Monospace font` is used for sample code and code listings, API and language elements (such as function names and class names), filenames, pathnames, directory names, HTML tags, and any text that must be typed on the screen.
- *Italic serif font* is used within code and code fragments to indicate variable placeholders. For example, the following command uses *filename* as a variable placeholder for an argument to the `gunzip` command:

  `gunzip -d` *filename*`.tar.gz`

Terminology

The following terms are used in the Identity Server documentation set:

- *Identity Server* refers to Identity Server and any installed instances of the Identity Server software.
- *Policy and Management services* refers to the collective set of Identity Server components and software that are installed and running on a dedicated deployment container such as a web server.
- *Directory Server* refers to an installed instance of Sun Java System Directory Server.
- *Application Server* refers to an installed instance of Sun Java System Application Server (also known as Sun ONE Application Server.)
- *Web Server* refers to an installed instance of Sun Java System Web Server (also known as Sun ONE Web Server).
- *Web container that runs Identity Server* refers to the dedicated J2EE container (such as Web Server or Application Server) where the Policy and Management Services are installed.
- *IdentityServer_base* represents the base installation directory for Identity Server. The Identity Server 2004Q2 default base installation and product directory depends on your specific platform:
    - Solaris™ systems: `/opt/SUNWam`
    - Linux systems: `/opt/sun/identity`

  The product directory is `/SUNWam` for Solaris systems and `/identity` for Linux systems. When you install Identity Server 2004Q2, you can specify a different directory for `/opt` on Solaris systems or `/opt/sun` on Linux systems; however, do not change the `/SUNWam` or `/identity` product directory.

  For the base installation directory of the following products, refer to the documentation for the specific product.
- *DirectoryServer_base* represents the base installation directory for Sun Java System Directory Server.
- *ApplicationServer_base* is a variable place holder for the home directory for Sun Java System Application Server.
- *WebServer_base* is a variable place holder for the home directory for Sun Java System Web Server.

Related Information

Useful information can be found at the following locations:

- Directory Server documentation:
  http://docs.sun.com/coll/DirectoryServer_04q2
- Web Server documentation:
  http://docs.sun.com/coll/S1_websvr61_en
- Application Server documentation
  http://docs.sun.com/coll/s1_asseu3_en
- Web Proxy Server documentation:
  http://docs.sun.com/prod/s1.webproxys#hic
- Download Center:
  http://wwws.sun.com/software/download/
- Technical Support:
  http://www.sun.com/service/sunone/software/index.html
- Professional Services:
  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

Related Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

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 by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Chapter 1

Overview of Identity Server

Sun Java™ System Identity Server is an identity management solution designed to meet the needs of rapidly expanding enterprises. Identity Server enables you to get identities for your employees, your partners and suppliers into one online directory. Then it provides a means for establishing policies and permissions regarding who has access to which information in your enterprise. Identity Server is the key to all your data, your services, and who has access to what—it's the key to all your internal and external business relationships.

This chapter provides an overview Identity Server and how its components work together. Topics include:

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

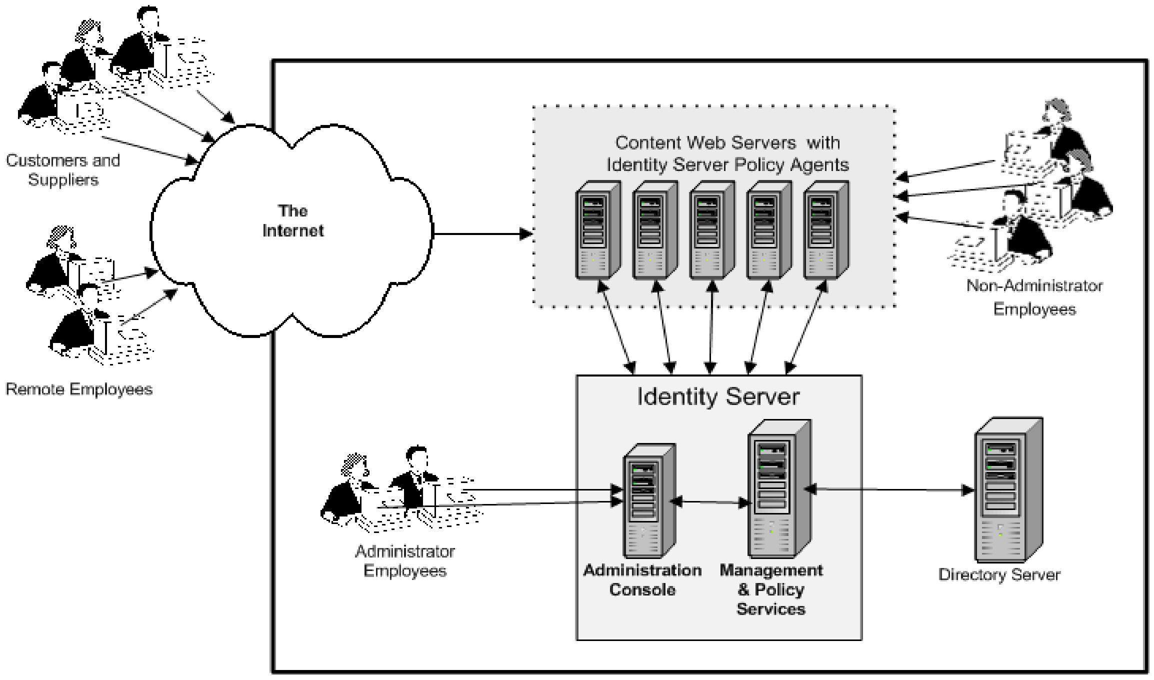
Identity Server reduces the administrative costs and eliminates the redundant user information associated with per-application solutions. Identity Server creates a single record or *directory entry* for each enterprise user, and enables an administrator to assign specific rules or *policies* governing 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*. Identity Server makes it possible for a user to access many resources in the enterprise with just one identity.

How Identity Server 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 Identity Server. Identity Server asks the user to present credentials such as a username and password. If the credentials match those stored in the central Directory Server, Identity Server verifies that the user is who he says he is. Next, Identity Server evaluates the policies associated with the user’s identity, and then determines whether the user is allowed to view the requested information.

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

Figure 1-1 Identity Server is the gatekeeper to a company’s enterprise resources.



Identity Server 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

Identity Server provides an identity framework for creating and managing directory objects such as organizations, groups, roles, and userIDs. When you use Identity Server to create or modify user objects, you update the entries stored in Directory Server. Identity Server 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

Identity Server 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

Identity Server 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 45.

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 53.

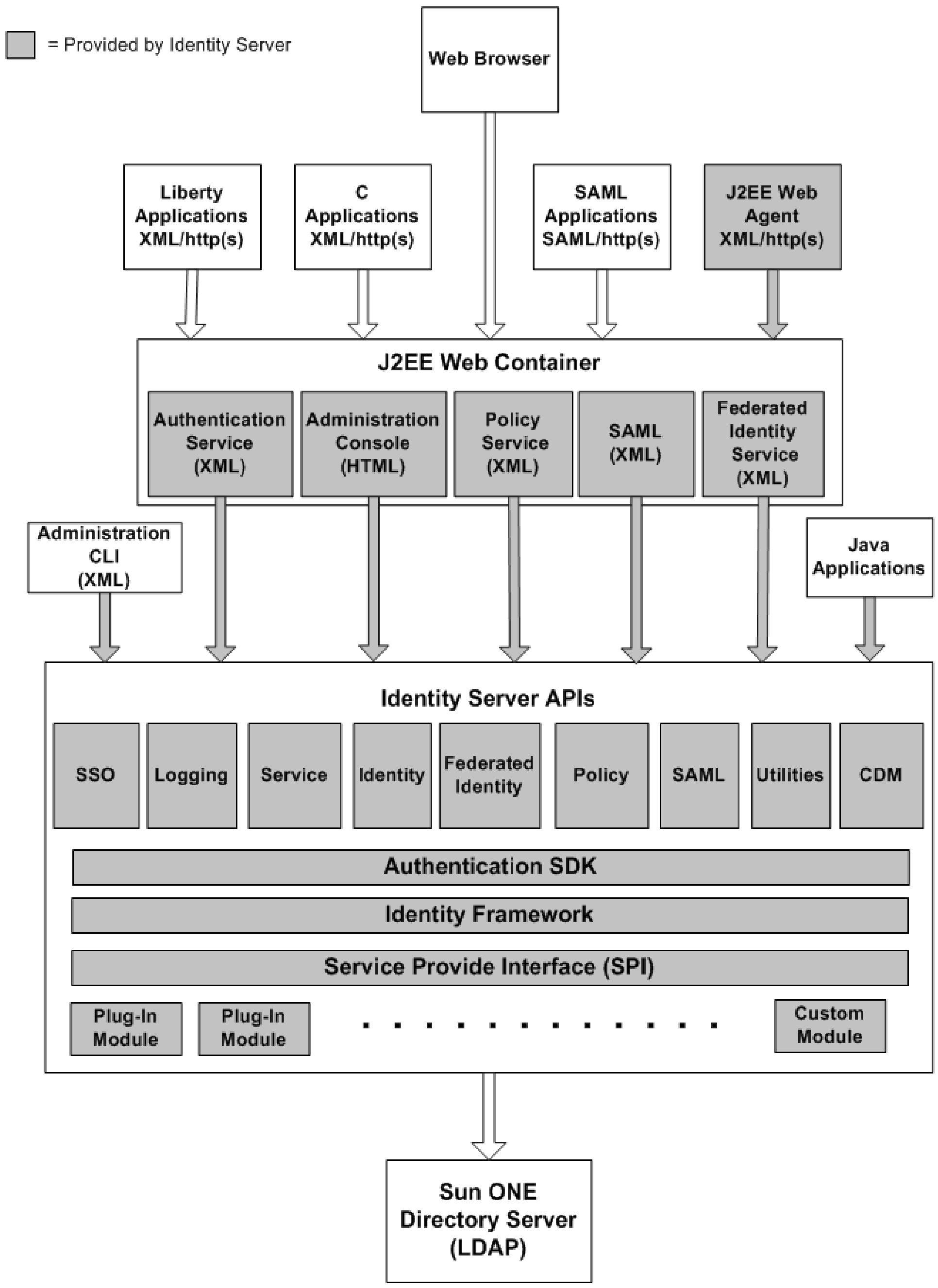
Identity Server Architecture

Identity Server 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 Identity Server integrates all of these technologies and connects to Directory Server. The Identity Server common identity infrastructure is built upon Directory Server which uses the LDAP protocol.

Figure 1-2 Identity Server Architecture.


= Provided by Identity Server
Web Browser
Liberty Applications XML/http(s)
C Applications XML/http(s)
SAML Applications SAML/http(s)
J2EE Web Agent XML/http(s)
J2EE Web Container
Authentication Service (XML)
Administration Console (HTML)
Policy Service (XML)
SAML (XML)
Federated Identity Service (XML)
Administration CLI (XML)
Java Applications
Identity Server APIs
SSO
Logging
Service
Identity
Federated Identity
Policy
SAML
Utilities
CDM
Authentication SDK
Identity Framework
Service Provide Interface (SPI)
Plug-In Module
Plug-In Module
Custom Module
Sun ONE Directory Server (LDAP)

Sun Java System Directory Server

In an Identity Server 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.

Identity Server Components

Identity Server 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.

Identity Server 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 Identity Server 2004Q2 include the following:

- Enhancements to Federation Management
- Enhancements to SAML
- Customized JAAS Authorization Framework
- Enhancements to Administration Console
- Session Failover for Application Server
- Nested Groups Support
- Enhancements to Authentication

Enhancements to Federation Management

The Federation Management component in Identity Server 2004Q2 is based on the Liberty 2.0 specification. The Liberty 2.0 architecture consists of multi-layered specifications set based on open standards, SAML and SOAP. Identity Server 2004Q2 extends the Identity Federation Framework provided in previous versions and offers two additional Liberty-based frameworks.

For more information, see the *Identity Server 2004Q2 Federation Management Guide* (http://docs.sun.com/doc/817-6362). This guide contains detailed information about the following three frameworks.

Identity Federation Framework

The liberty Identity Federation Framework (ID-FF) specifies protocols, schema and profiles for creating a Liberty-enabled environment. Identity Server 2004Q2 extends this framework which was first offered in Identity Server 6.1.

Liberty Identity Web Services Framework

The Liberty Identity Web Services Framework (ID-WSF) specifies protocols, schema and profiles for implementing identity services such as Identity Service Discovery and invocation. This is new in Identity Server 2004Q2.

Identity Service Instance Specification

The Liberty Identity Service Instance Specification (ID-SIS) use the Federation Framework and Web Services Framework to provide network identity services. A Personal Profile Service that can be used out-of-box, and a sample Employee Profile Service are included with Identity Server. The framework and services are new in Identity Server 2004Q2.

Enhancements to SAML

Identity Server 2004Q2 implements all mandatory parts of the Security Assertion Markup Language (SAML) 1.1 specifications which are also used in Liberty 2.0 specifications. SAML 1.0 specifications implemented in Identity Server 6.1 continue to be supported. Identity Server 2004Q2 uses the SAML 1.0 format for Liberty 1.1 data; for Liberty 2.0 data, the SAML 1.1. is used. The versioning of all SAML requests, responses, and data elements is handled automatically. All new features defined in SAML 1.1 specifications will be implemented.

Customized JAAS Authorization Framework

Identity Server implements and extends the JAAS interface for authorization. The Identity Server implementation offers the following added benefits over the standard JAAS interface:

- Identity Server policy is centralized, allowing distributed applications to execute policies defined in single secure store. This makes policies easier to manage across an enterprise.
- The Identity Server framework can handle non-boolean decisions such as " What is the salary of employee A?"
- Policies are resource-centric instead of user-centric to better serve the needs of web-based internet portals. For a given a resource, you can define who can use it and in what way.

Enhancements to Administration Console

The Identity Server administration console was enhanced to include the following new features. For detailed information, see the *Identity Server Administration Guide* (`http://docs.sun.com/doc/817-5709`).

Centralized Agents Management

Administrators can now view, create, modify and delete agent profiles using the Identity Server administration console.

Display Options and Available Actions

You can now use the Display Options view to customize the way in which Identity Server objects such as organizations, roles, and containers are displayed in the Identity Server console. Not all display options are available for all object types. For certain Identity Server object types, you can define user access rights through the Available Actions view.

Session Failover for Application Server

Identity Server 2004Q2 provides session failover using Sun Java System Application Server 7.0.0_01 Enterprise Edition (EE) as a web container.

Session failover automatically and transparently redirects an Identity Server request to a secondary server if the primary server fails because a hardware or software problem occurs or if the server is temporarily shut down.

NOTE Application Server 7.0.0_01 EE is not a component of the Sun Java Enterprise System 2004Q2 release. To obtain a copy of this release, contact your Sun Microsystems technical representative.

For more information, see the *Identity Server Deployment Planning Guide* (`http://docs.sun.com/doc/817-5707`).

Nested Groups Support

This release of Identity Server supports nested groups, which are "representations" of existing groups contained in a single group. As opposed to sub-groups, nested groups can exist anywhere in the directory information tree (DIT). Nested groups allow you to quickly set up access permissions for a large number of users. For detailed information on nested groups, see the *Identity Server Administration Guide* (`http://docs.sun.com/doc/817-5709`).

Configuration and Tuning Scripts

Identity Server 2004Q2 provides new scripts for post-installation configuration and for performance tuning. For detailed information on the following scripts, see the *Identity Server Administration Guide* (`http://docs.sun.com/doc/817-5709`).

Configuration Script

After you have installed the first instance of Identity Server using the Java Enterprise System installer, you can use the a script to reconfigure the instance, or to deploy and configure additional Identity Server instances. First you edit the configuration variables in the silent mode input file, and then your run the `amconfig` script.

Tuning Scripts

The `amtune` scripts allow you to tune the performance of Identity Server, as well as optimize the performance settings for various components of your Identity Server deployment.

Enhancements to Authentication

For detailed information about the following new features, see the *Identity Server Developer's Guide* (`http://docs.sun.com/doc/817-5710`):

- JAAS Shared State
- Agent Authentication
- Java Database Connectivity Authentication Module Sample
- Java Card Digital Identity Authentication Module Sample
- Windows Desktop Single Sign-On

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 in the Authentication Configuration for each of the following objects:

- Organization
- User
- Service
- Role

Upon failure, the module prompts for its required credentials. After failed authentication, the module stops running, or the logout shared state clears

Agent Authentication

Agent authentication is now supported in LDAP and Application authentication modules.

Java Database Connectivity Authentication Module Sample

Java Database Connectivity (JDBC) technology provides authentication of users against an external database such as Oracle, MySQL, or Sybase databases. This module leverages container provided connection pools and has a pluggable password transform that translates encryption for varying password formats. This module also provides for configuration of the SQL statement that is used to retrieve a password from the database.

The JDBC sample provided with this Beta version of Identity Server 2004Q2 is not an officially supported authentication module. The sample and related information is located in this directory:

IdentityServer_base`/SUNWam/samples/authentication/spi/jdbc`

Java Card Digital Identity Authentication Module Sample

The Java Card Digital Identity (JCDI) Authentication module provides for authentication of Java Card (Certificate and Serial Number) using the com.sun.jndi.ldap. LdapCtxFactory package.The JCDI authentication sample demonstrates the use of Java Card authentication with Identity Server. The sample has two components:

- Remote client
- Server JCDI authentication module

The remote client component is located in the following directory:

IdentityServer_base`/samples/authentication/api/jcdi directory`

The server JCDI authentication module is located in the following directory:

IdentityServer_base`/samples/authentication/spi/jcdi directory`

The JDBC sample provided with this Beta version of Identity Server 2004Q2 is not an officially supported authentication module.

Windows Desktop Single Sign-On

Kerberos authentication is supported in this release of Identity Server. A new Windows Desktop Single Sign-On module allows a client or user *who has already been authenticated by a Kerberos Distribution Center (KDC)* to be authenticated by Identity Server without having to provide the login information again. The Microsoft Internet Explorer (5.01 or newer) on Windows 2000 or later is the only available client that currently supports this protocol. Therefore the module is designed for Windows desktop users. JDK 1.4 or above is required to utilize the new features of the Kerberos V5 authentication module, and Java GSS APIs are required to perform Kerberos-based SSO in this module.

Chapter 2

Identity Management

Built upon Sun Java™ System Directory Server, Sun Java System Identity Server 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 Identity Server work together to achieve consolidated identity management.

Topics in this chapter include the following:

- "Basic Directory Server Concepts" on page 25
- "How Identity Server 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. Identity Server 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 Identity Server, you are asked to name an Identity Server root suffix. The Identity Server root suffix identifies the part of the directory tree that is managed by Identity Server. The Identity Server 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

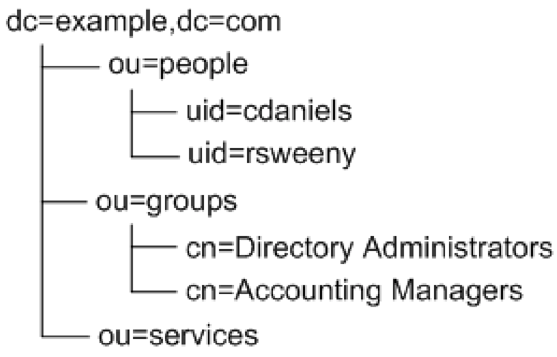
– `ou` refers to an organizational unit or group to which an object belongs

– `uid` refers to the string or name that Identity Server 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 Identity Server 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 Identity Server.

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 Identity Server to add a user to a group, Identity Server 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. Identity Server 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.
- Identity Server groups are also used for defining policy subjects. See "Policy Configuration" on page 39 for related information. Note that Identity Server 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 Identity Server administration console.

How Identity Server Works with Directory Server

When you install Identity Server, 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 Identity Server administration console to create and manage the directory entries in Directory Server.

Identity Server Objects Are Added to Directory

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

Groups

An Identity Server 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 Identity Server console. Identity Server services are discussed in greater detail in Chapter 4, “Services Management” on page 45.

Roles

An Identity Server *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. Identity Server roles are also used as policy subjects for the purpose of service inheritance. See “Policy Configuration” on page 39 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. Identity Server 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 Identity Server container entry and the Identity Server 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 Identity Server, Identity Server 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 Identity Server 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 Identity Server functionality, Identity Server 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 Identity Server 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 Identity Server into external applications or services.

- Identity Server console

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

```
fjdkslfjd
dkfjslfjd
    djfkdflsfj
    djfkdlfj
    jdkfldfj
    djfkdsl
```

Chapter 3

Access Management

Sun Java™ System Identity Server 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, Identity Server 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 Identity Server access management features and functionality.

Topics included in this chapter are:

Authentication

Authentication is the process of verifying that a person is who he says he is. Identity Server 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. Identity Server compares the user's input with data stored in Directory Server. If Identity Server finds a match for the user name, and if the given password matches the password stored in Directory Server, then Identity Server can verify the user's identity, and the user is authenticated. Access is granted, and the user corporate phone book is displayed to the user.

Client Detection

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

Basic Authentication

There are a number of ways to access Identity Server for basic authentication. Java™ and C applications can use the Java and C authentication APIs to connect to Identity Server, while end users open a connection to Identity Server using a web browser.

Users Using A Web Browser

A user with a web browser can authenticate to Identity Server using the Authentication User Interface. The URI for this web-based interface is:

`http://`*identity_server_host.domain_name:port*`/amserver/UI/Login`.

After entering the URL, the user is prompted to submit verifying credentials. Once the credentials have been passed back to Identity Server, the user gains (or is denied) access based on their privileges:

- Administrators can access the identity administration portion of the console to manage authentication data.
- End users can modify personal data in their own user profiles.
- Upon authorization, a specified success or failure URL defines what the user will access.

Java Applications

External Java applications can authenticate to the Identity Server to access any of its protected resources using the Authentication API for Java. This API provides interfaces to initiate the authentication process and communicate authentication credentials to the Authentication Service. The API is defined in a Java package called `com.sun.identity.authentication`. Developers incorporate the classes and methods from this package into their Java applications to allow communication with the Authentication Service. The application's Java request is first converted to an XML message format and passed to the Identity Server over HTTP(S). Once received, the XML message is converted back into a Java request which can be interpreted by the Authentication Service.

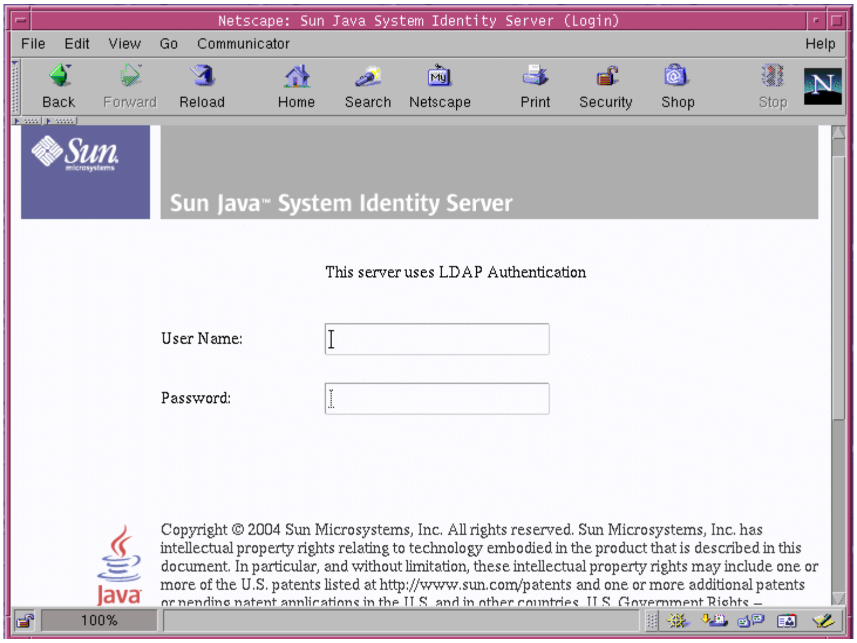
C Applications

Identity Server also includes resources for C applications to authenticate to the Identity Server. 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.

The Authentication User Interface

The Authentication Service has a separate user interface from the Identity Server console. It provides the web-based interface for all authentication modules installed in the Identity Server deployment. The user interface provides a dynamic and customizable means for gathering authentication credentials by presenting the web-based login requirement pages to a user requesting access. Figure 3-1 is a screenshot of the default user interface for LDAP authentication.

Figure 3-1 LDAP Authentication User Interface



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 –

Single Sign-On

When a user wants to access *multiple* resources protected by Identity Server, 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 Identity Server this functionality.

Cross-Domain Single Sign-On

A user authenticated to Identity Server 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 Identity Server: 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 3-2 Single-domain SSO vs. Cross-domain SSO

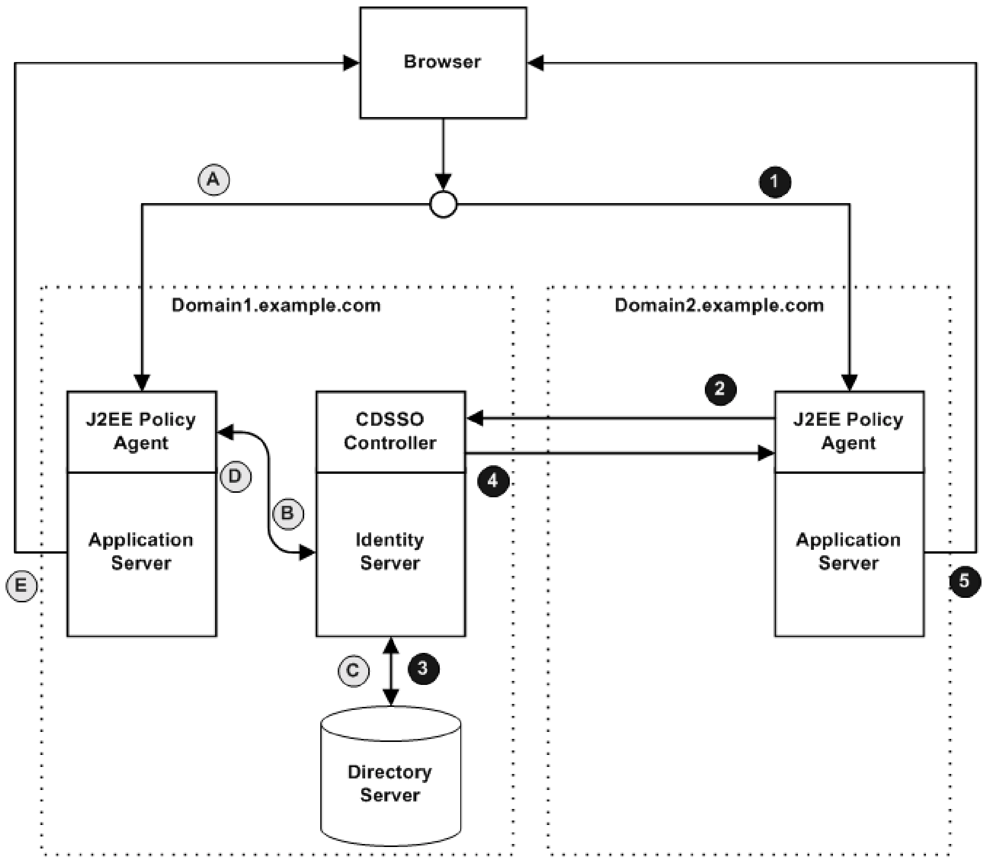


Figure 3-2 compares the sequence of SSO events in a home domain (Domain1) with the sequence of SSO events in a foreign domain (Domain2). The home domain contains Identity Server with the CDC enabled. A foreign domain is any domain other than the home domain. In this example, within Domain1, when the 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 Identity Server 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 Identity Server 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 the 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 (E).

Policy Management and Configuration

A policy is a rule that describes who is allowed or *authorized* to access a specific web resource. Identity Server provides a policy management service and policy configuration service for creating such rules and evaluating policies. Policy agents are an integral part of an Identity Server 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 Identity Server 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 Identity Server. All of the Identity Server 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 Identity Server, 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 Identity Server administrator, stored within the Identity Server 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 Identity Server Policy Agents Guide*

Policy Types

There are two types of policies that can be configured using Identity Server: *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:

- Identity Server 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. Identity Server 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

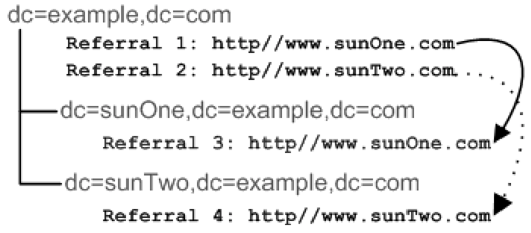
A policy used for delegation is a *referral* policy. A referral policy delegates both policy creation and policy evaluation. It consists of one or more rules and one or more referrals.

- 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 3-3. 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 3-3 Root-level policies

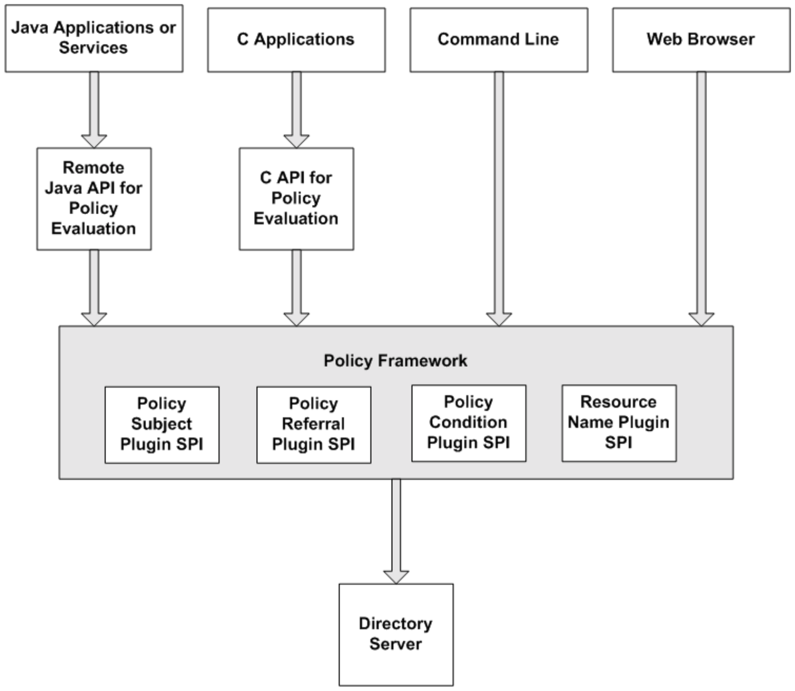

dc=example,dc=com
Referral 1: http//www.sunOne.com
Referral 2: http//www.sunTwo.com
dc=sunOne,dc=example,dc=com
Referral 3: http//www.sunOne.com
dc=sunTwo,dc=example,dc=com
Referral 4: http//www.sunTwo.com

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 3-4 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 3-4 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, Identity Server 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 Identity Server 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.

Chapter 4

Services Management

Sun Java™ System Identity Server comes with its own set of core services which make user access and policy management possible. Identity Server 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 Identity Server core services and service plug-ins work together. Topics included in this chapter are:

- "How Services Work in Identity Server" on page 45
- "Attribute Types" on page 47
- "Identity Server Core Services" on page 48
- "The Service Configuration Interface" on page 52

How Services Work in Identity Server

In Identity Server, a *service* is a group of attributes that are managed together by the Identity Server 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 Identity Server 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

Identity Server comes with a number of services that work together to perform its basic functions. See "Identity Server Core Services" on page 48 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 Identity Server 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.

Identity Server also issues a request to the Naming Service which defines the URLs used to access other Identity Server 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, Identity Server 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 Identity Server Software Development Kit (SDK) layer. Identity Server 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 Identity Server session management service comes into play. When the employee's mail session sits idle for too long, for example, Identity Server communicates with the mail application which causes the mail session to time-out or expire.

The Identity Server SDK gives application developers the interfaces necessary to register and un-register services with Identity Server, 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 *Sun One Identity Server Programmer'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 39 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 Identity Server 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 Webserver issues a client request for username and password, and sends that information back to the server as part of the authorized request. Identity Server 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 Identity Server 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 Identity Server 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 Identity Server administration. An administrator can configures values such as log file size and log file location. Identity Server 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

Identity Server 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 Identity Server.

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

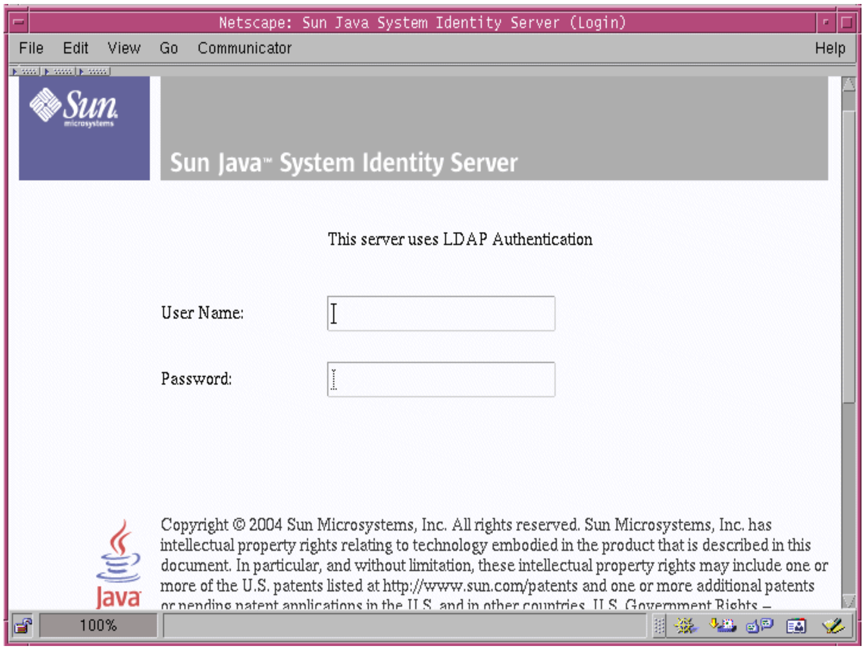
Default user preferences are defined through the user service. (These include time zone, locale and DN starting view).

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 Identity Server
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 –

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 Identity Server 2004Q2.

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:

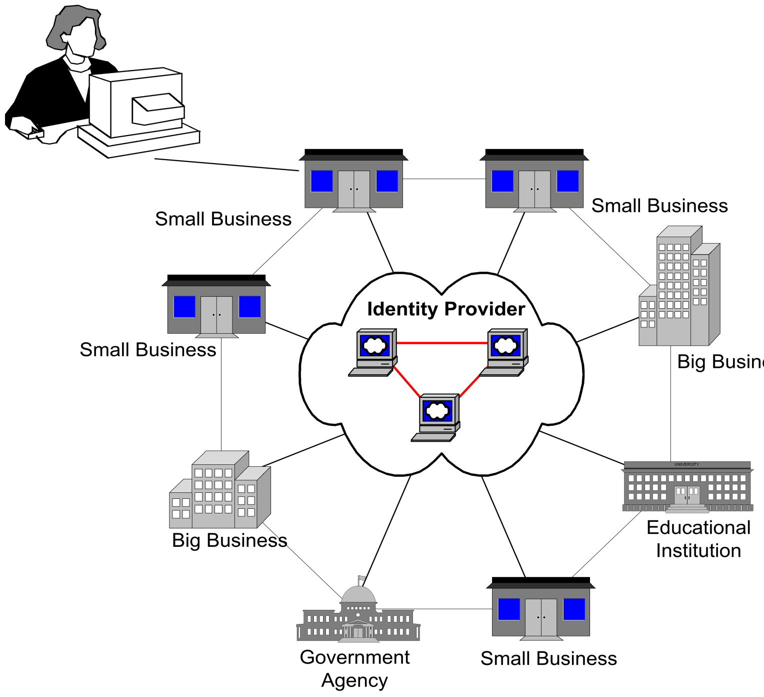
`http://projectliberty.org/`

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 Identity Server 6.1 release implemented an Identity Federation Framework (ID-FF) version 1.1 Identity Server 2004Q2 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:

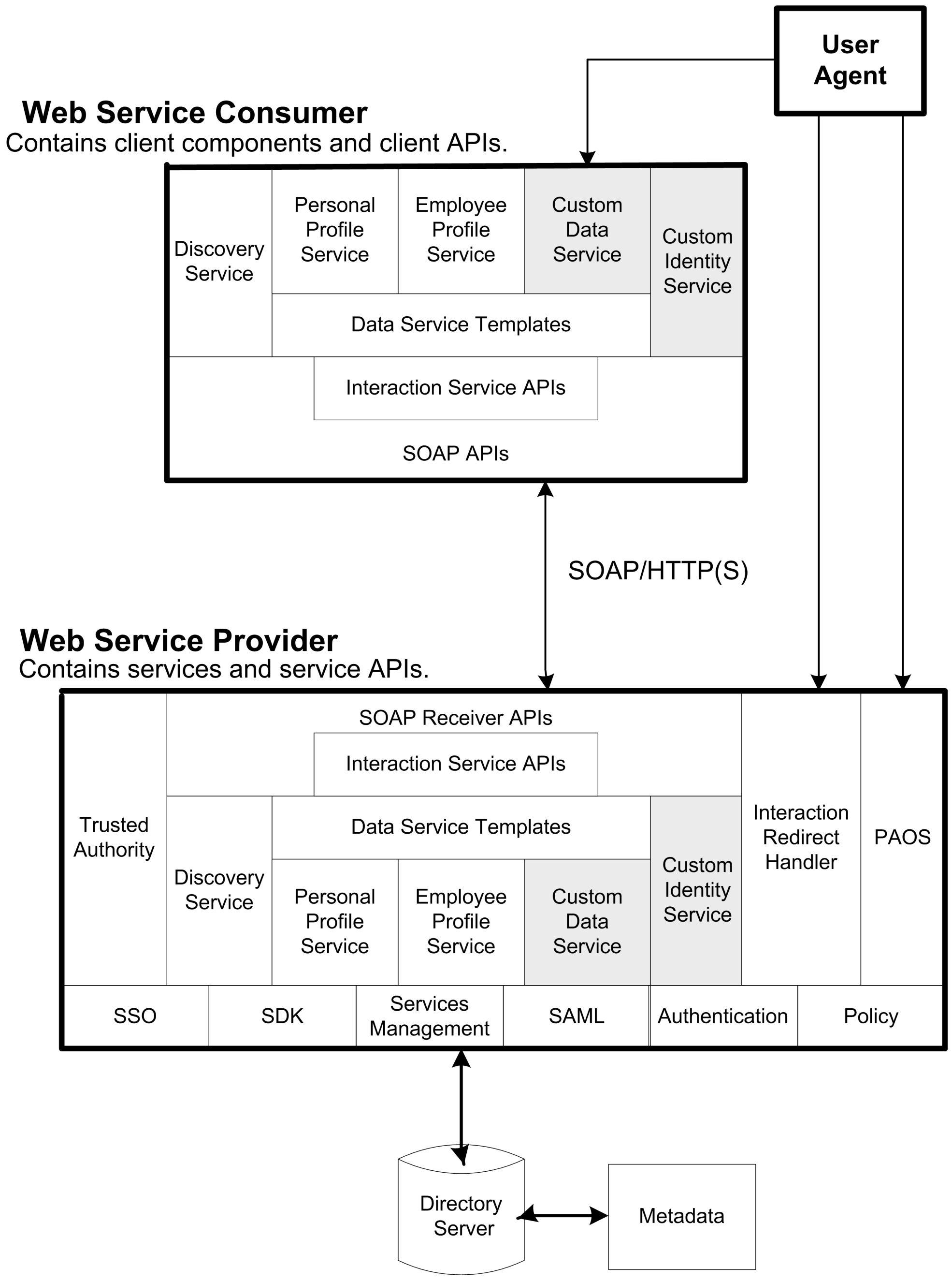
- SOAP-based Single Log-Out Protocol relies on asynchronous SOAP messaging calls between providers.
- HTTP Redirect-based Single Log-Out Protocol

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 Identity Server 2004Q2. The components in the bottom layer of the Web Service Provider were implemented in Identity Server 6.1.These components include Single-Sign On (SS0), the Identity Server 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 Identity Server 2004Q2.

Personal Profile Service. An instance of a data-oriented web service which offers profile information regrading 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 Identity Server data store.

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

The Federation Management Process

The Federation Management process begins with authentication. By default, Identity Server 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 Identity Server, he is redirected to the Authentication Service using an Identity Server 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 Identity Server, the authentication process begins with a search for a valid Identity Server 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 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, Identity Server 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 Identity Server 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 Identity Server session token is found, the pre-login process begins with the search for another type of cookie, a Federation cookie.

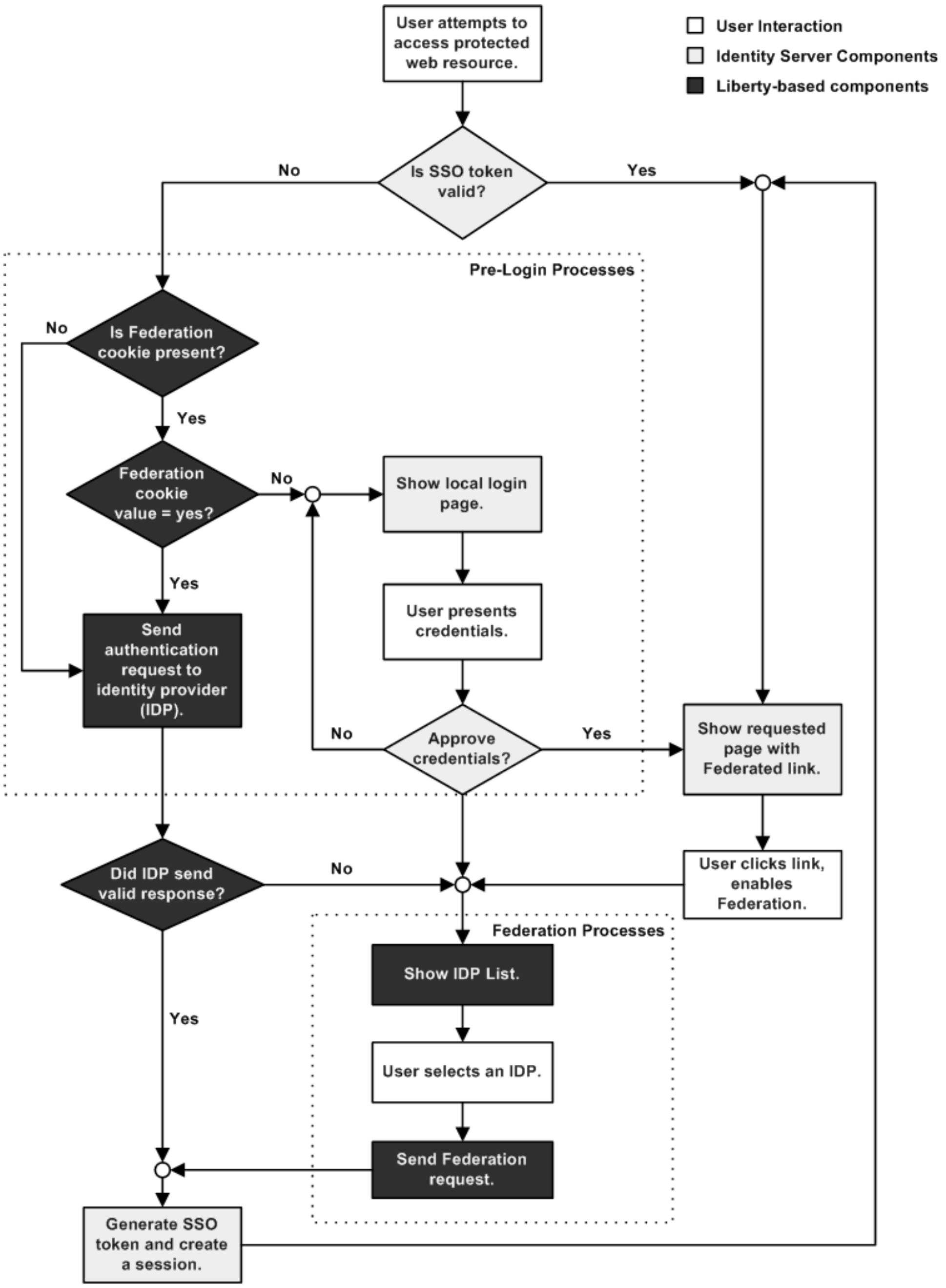
If, after the search for an Identity Server session token proves null, a Federation cookie is found and its value is “no,” an Identity Server login page is displayed and the user submits credentials to the Authentication Service. When authenticated by the Identity Server, the user is redirected to the 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 Identity Server 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 Identity Server 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 Identity Server Authentication Process Flow

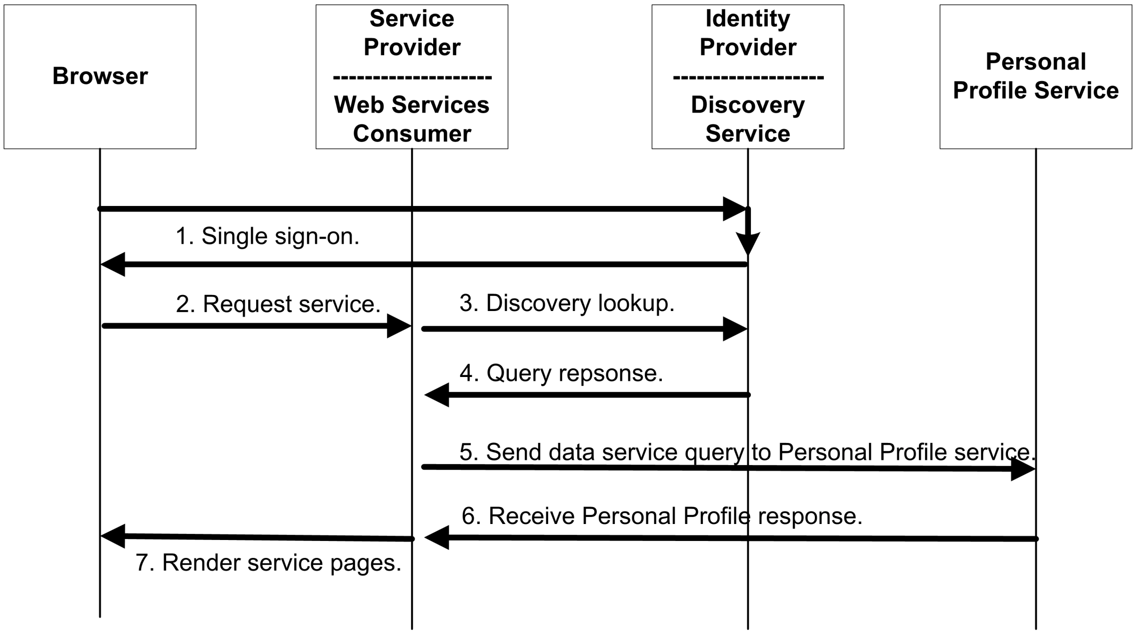

User attempts to access protected web resource.
User Interaction
Identity Server Components
Liberty-based components
Is SSO token valid?
No
Yes
Pre-Login Processes
Is Federation cookie present?
No
Yes
Federation cookie value = yes?
No
Show local login page.
Yes
User presents credentials.
Send authentication 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.html`

Index