

IBM® Security Access Manager for Enterprise Single
Sign-On
Version 8.2

Provisioning Integration Guide



IBM® Security Access Manager for Enterprise Single
Sign-On
Version 8.2

Provisioning Integration Guide



Note

Before using this information and the product it supports, read the information in “Notices” on page 75.

Edition notice

Note: This edition applies to version 8.2 of IBM Security Access Manager for Enterprise Single Sign-On, (product number 5724–V67) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2002, 2012.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this publication	v
Intended audience	v
What this publication contains	v
Publications	vi
IBM Security Access Manager for Enterprise	
Single Sign-On library	vi
Accessing terminology online	viii
Accessing publications online	viii
Ordering publications	viii
Tivoli user groups	viii
Accessibility	viii
Tivoli technical training	ix
Support information	ix
Conventions used in this publication	ix
Typeface conventions	ix
Operating system-dependent variables and paths	x

Chapter 1. Security Access Manager for Enterprise Single Sign-On Provisioning . 1

Key features and benefits of provisioning.	1
Integration solutions.	3
Configuring the IMS Server for provisioning.	3

Chapter 2. Provisioning API . 5

About Provisioning API	5
IBM Security Access Manager for Enterprise Single Sign-On Provisioning overview	6
Provisioning API system requirements.	6
Available API types for Provisioning API.	6
Java API for provisioning	7
Certificate store configuration for the IMS Bridge (Java Provisioning API).	7
IMS Bridge sample configuration file (Java Provisioning API).	7
IMS Bridge configuration parameters (Java Provisioning API).	8
Developing an integration module by using the IMS Bridge (Java Provisioning API).	9
Using a command-line tool (CLT) to invoke the IMS Bridge API (Java Provisioning API).	10
Java API classes	12
SOAP API for provisioning	28
Usage types (SOAP API Provisioning)	29

SOAP client development	30
IMS Server configuration for SOAP API	31
SOAP API data types for Provisioning API	32
SOAP API operations	35
Using Provisioning API for account setup and maintenance	50
Provisioning a new user by using Provisioning API	50
Adding an application account by using Provisioning API	51
Resetting an application password by using Provisioning API	51
Deleting an application account by using Provisioning API	51
De-provisioning users by using Provisioning API	51

Chapter 3. Provisioning Agent. . 53

Provisioning Agent features	54
Installing and configuring the Provisioning Agent	55
Installing the Provisioning Agent	55
Configuring the Provisioning Agent	56
Setting up the Provisioning Agent with Active Directory Application Mode (ADAM)	56
Starting the Provisioning Agent.	57
Provisioning Agent configuration parameters	57

Appendix A. WSDL for server authentication . 61

Appendix B. WSDL for provisioning service. . 63

Appendix C. Using the Keytool for provisioning . 73

Notices . 75

Glossary . 79

Index . 87

About this publication

The IBM® Security Access Manager for Enterprise Single Sign-On provides sign-on and sign-off automation, authentication management, and user tracking to provide a seamless path to strong digital identity. The *IBM Security Access Manager for Enterprise Single Sign-On Provisioning Integration Guide* provides information for configuring, managing, and troubleshooting the provisioning integration solutions for the product.

Intended audience

This publication is for technical users who understand how IBM Security Access Manager for Enterprise Single Sign-On can be enhanced and customized for user provisioning purposes.

This publication is for Administrators and system programmers who need to perform the following tasks:

- Provisioning IMS Server users
- Provisioning maintenance tasks, such as revoking users, checking user status, resetting passwords by using SOAP API
- IMS Server setup for integration with provisioning systems
- Certificate store configuration

Readers need to be familiar with the following topics:

- Using Java™ API or SOAP API
- Web Services Description Language (WSDL)
- Using other identity provisioning systems, such as the Provisioning API and IBM Security Access Manager for Enterprise Single Sign-On Adapter
- Using command-line tools (CLTs) to perform specific tasks, such as invoking the IMS Bridge API

What this publication contains

This publication contains the following sections:

- Chapter 1, "Security Access Manager for Enterprise Single Sign-On Provisioning," on page 1
Provides an overview of how other provisioning solutions can integrate with IBM Security Access Manager for Enterprise Single Sign-On.
- Chapter 2, "Provisioning API," on page 5
Describes the Provisioning API for provisioning and its integration mechanism with third-party identity provisioning systems.
- Chapter 3, "Provisioning Agent," on page 53
Describes how to set up the Provisioning Agent for monitoring de-provisioned accounts on Active Directory without a third-party identity provisioning system.
- Appendix A, "WSDL for server authentication," on page 61
Contains the list of WSDLs used for server authentication when developing a SOAP client.
- Appendix B, "WSDL for provisioning service," on page 63

Contains the list of WSDLs used for provisioning when developing a SOAP client.

- Appendix C, “Using the Keytool for provisioning,” on page 73
Describes how to use the Sun Java Keytool to create keystores or import certificates.

Publications

This section lists publications in the IBM Security Access Manager for Enterprise Single Sign-On library. The section also describes how to access Tivoli® publications online and how to order Tivoli publications.

IBM Security Access Manager for Enterprise Single Sign-On library

The following documents are available in the IBM Security Access Manager for Enterprise Single Sign-On library:

- *IBM Security Access Manager for Enterprise Single Sign-On Quick Start Guide*, CF38DML

Read this guide for a quick start on the main installation and configuration tasks to deploy and use IBM Security Access Manager for Enterprise Single Sign-On.

- *IBM Security Access Manager for Enterprise Single Sign-On Planning and Deployment Guide*, SC23995203

Read this guide before you do any installation or configuration tasks. This guide helps you to plan your deployment and prepare your environment. It provides an overview of the product features and components, the required installation and configuration, and the different deployment scenarios. It also describes how to achieve high availability and disaster recovery.

- *IBM Security Access Manager for Enterprise Single Sign-On Installation Guide*, GI11930901

Read this guide for the detailed procedures on installation, upgrade, or uninstallation of IBM Security Access Manager for Enterprise Single Sign-On.

This guide helps you to install the different product components and their required middleware, and also do the initial configurations required to complete the product deployment. It covers procedures for using virtual appliance, WebSphere® Application Server Base editions, and Network Deployment.

- *IBM Security Access Manager for Enterprise Single Sign-On Configuration Guide*, GC23969201

Read this guide if you want to configure the IMS Server settings, the AccessAgent user interface, and its behavior.

- *IBM Security Access Manager for Enterprise Single Sign-On Administrator Guide*, SC23995103

This guide is intended for the Administrators. It covers the different Administrator tasks. This guide provides procedures for creating and assigning policy templates, editing policy values, generating logs and reports, and backing up the IMS Server and its database. Use this guide together with the IBM Security Access Manager for Enterprise Single Sign-On Policies Definition Guide.

- *IBM Security Access Manager for Enterprise Single Sign-On Help Desk Guide*, SC23995303

This guide is intended for Help desk officers. The guide helps Help desk officers to manage queries and requests from users usually about their authentication

factors. Use this guide together with the IBM Security Access Manager for Enterprise Single Sign-On Policies Definition Guide.

- *IBM Security Access Manager for Enterprise Single Sign-On Policies Definition Guide*, SC23969401

Read this guide for the detailed descriptions of the different user, machine, and system policies that Administrators can configure in AccessAdmin. Use this guide along with the IBM Security Access Manager for Enterprise Single Sign-On Administrator Guide.

- *IBM Security Access Manager for Enterprise Single Sign-On Troubleshooting and Support Guide*, GC23969301

Read this guide if you have any issues with regards to installation, upgrade, and product usage. This guide covers the known issues and limitations of the product. It helps you determine the symptoms and workaround for the problem. It also provides information about fixes, knowledge bases, and support.

- *IBM Security Access Manager for Enterprise Single Sign-On AccessStudio Guide*, SC23995603

Read this guide if you want to create or edit profiles. This guide provides procedures for creating and editing standard and advanced AccessProfiles for different application types. It also covers information about managing authentication services and application objects, and information about other functions and features of AccessStudio.

- *IBM Security Access Manager for Enterprise Single Sign-On Provisioning Integration Guide*, SC23995703

Read this guide for information about the different Java and SOAP API for provisioning. It also covers procedures for installing and configuring the Provisioning Agent.

- *IBM Security Access Manager for Enterprise Single Sign-On Web API for Credential Management Guide*, SC14764600

Read this guide if you want to install and configure the Web API for credential management.

- *IBM Security Access Manager for Enterprise Single Sign-On Lightweight AccessAgent mode on Terminal Server SDK Guide*, SC14765700

Read this guide for the details on how to develop a virtual channel connector that integrates AccessAgent with Terminal Services applications.

- *IBM Security Access Manager for Enterprise Single Sign-On Serial ID SPI Guide*, SC14762600

IBM Security Access Manager for Enterprise Single Sign-On has a Service Provider Interface (SPI) for devices that contain serial numbers, such as RFID. See this guide to know how to integrate any device with serial numbers and use it as a second authentication factor with AccessAgent.

- *IBM Security Access Manager for Enterprise Single Sign-On Context Management Integration Guide*, SC23995403

Read this guide if you want to install and configure the Context Management solution.

- *IBM Security Access Manager for Enterprise Single Sign-On User Guide*, SC23995003

This guide is intended for the end users. This guide provides instructions for using AccessAgent and Web Workplace.

- *IBM Security Access Manager for Enterprise Single Sign-On Error Message Reference Guide*, GC14762400

This guide describes all the informational, warning, and error messages associated with IBM Security Access Manager for Enterprise Single Sign-On.

Accessing terminology online

The IBM Terminology Web site consolidates the terminology from IBM product libraries in one convenient location. You can access the Terminology Web site at the following Web address:

<http://www.ibm.com/software/globalization/terminology>

Accessing publications online

IBM posts publications for this and all other Tivoli products, as they become available and whenever they are updated, to the Tivoli Information Center Web site at <http://www.ibm.com/tivoli/documentation>.

Note: If you print PDF documents on other than letter-sized paper, set the option in the **File > Print** window that allows Adobe Reader to print letter-sized pages on your local paper.

Ordering publications

You can order many Tivoli publications online at <http://www.elink.ibm.link.ibm.com/publications/servlet/pbi.wss>.

You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968

In other countries, contact your software account representative to order Tivoli publications. To locate the telephone number of your local representative, perform the following steps:

1. Go to <http://www.elink.ibm.link.ibm.com/publications/servlet/pbi.wss>.
2. Select your country from the list and click **Go**.
3. Click **About this site** in the main panel to see an information page that includes the telephone number of your local representative.

Tivoli user groups

Tivoli user groups are independent, user-run membership organizations that provide Tivoli users with information to assist them in the implementation of Tivoli Software solutions. Through these groups, members can share information and learn from the knowledge and experience of other Tivoli users. Tivoli user groups include the following members and groups:

- 23,000+ members
- 144+ groups

Access the link for the Tivoli Users Group at www.tivoli-ug.org.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

For additional information, see "Accessibility features" in the *IBM Security Access Manager for Enterprise Single Sign-On Planning and Deployment Guide*.

Tivoli technical training

For Tivoli technical training information, see the following IBM Tivoli Education Web site at <http://www.ibm.com/software/tivoli/education>.

Support information

If you have a problem with your IBM software, you want to resolve it quickly. IBM provides the following ways for you to obtain the support you need:

Online

Go to the IBM Software Support site at <http://www.ibm.com/software/support/probsub.html> and follow the instructions.

IBM Support Assistant

The IBM Support Assistant is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The IBM Support Assistant provides quick access to support-related information and serviceability tools for problem determination. To install the IBM Support Assistant software, go to <http://www.ibm.com/software/support/isa>.

Troubleshooting Guide

For more information about resolving problems, see the *IBM Security Access Manager for Enterprise Single Sign-On Troubleshooting and Support Guide*.

Conventions used in this publication

This publication uses several conventions for special terms and actions, operating system-dependent commands and paths, and margin graphics.

Typeface conventions

This publication uses the following typeface conventions:

Bold

- Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
- Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes, multicolumn lists, containers, menu choices, menu names, tabs, property sheets) and labels (such as **Tip:** and **Operating system considerations:**)
- Keywords and parameters in text

Italic

- Citations (examples: titles of publications, diskettes, and CDs)
- Words defined in text (example: a nonswitched line is called a *point-to-point line*)
- Emphasis of words and letters (words as words example: "Use the word *that* to introduce a restrictive clause."; letters as letters example: "The LUN address must start with the letter *L*.")
- New terms in text (except in a definition list): a *view* is a frame in a workspace that contains data.
- Variables and values you must provide: ... where *myname* represents....

Monospace

- Examples and code examples

- File names, programming keywords, and other elements that are difficult to distinguish from surrounding text
- Message text and prompts addressed to the user
- Text that the user must type
- Values for arguments or command options

Operating system-dependent variables and paths

This publication uses the UNIX convention for specifying environment variables and for directory notation.

When using the Windows command line, replace *\$variable* with *% variable%* for environment variables and replace each forward slash (/) with a backslash (\) in directory paths. The names of environment variables are not always the same in the Windows and UNIX environments. For example, %TEMP% in Windows environments is equivalent to \$TMPDIR in UNIX environments.

Note: You can use the UNIX conventions if you are using the bash shell on a Windows system.

Chapter 1. Security Access Manager for Enterprise Single Sign-On Provisioning

A provisioning system provides identity lifecycle management for application users in enterprises and manages their credentials.

Provisioning systems have become critical components of enterprise identity and access management strategies. IBM Security Access Manager for Enterprise Single Sign-On provides real-time implementation of access security policies for users and applications.

Integrating IBM Security Access Manager for Enterprise Single Sign-On with a provisioning system results in a complete identity and access management solution.

A complete solution provides automatic application account provisioning, centralized views of all application accounts, sign-on or sign-off automation, authentication management, user-centric audit logs and report generation, and centralized de-provisioning for all accounts.

Before you use automated provisioning, you must configure the IMS Server provisioning bridge. The provisioning bridge automates the IMS Server credential distribution process with third party provisioning systems by using API libraries with a SOAP connection.

If you are:

- Using a user provisioning system like Tivoli Identity Manager, you must configure the IMS Server provisioning bridge before you integrate IBM Security Access Manager for Enterprise Single Sign-On with the provisioning system.
- Not using a user provisioning system, but use Active Directory, you must configure the IMS Server provisioning bridge before you install and configure the Provisioning Agent. Provisioning Agent monitors Active Directory for de-provisioned accounts.
- When the Provisioning Agent detects de-provisioned accounts in Active Directory, the Provisioning Agent de-provisions IMS Server users from the IMS Server automatically.

Note: Provisioning Agent does not provision new accounts in Active Directory.

See the following topics for more information about IBM Security Access Manager for Enterprise Single Sign-On and provisioning.

- “Key features and benefits of provisioning”
- “Integration solutions” on page 3

Key features and benefits of provisioning

An identity provisioning system helps enterprises ensure that the right users have access to the right applications and infrastructure.

It provides a secure, automated, and policy-based identity lifecycle management solution that helps enterprises automate provisioning and de-provisioning of all user accounts. An identity provisioning system also provides a centralized view of all application credentials.

The embedded provisioning engine creates user credentials based on policies and defines the entitlements of the user accounts. It quickly connects users to appropriate enterprise resources while reducing administrative workload.

Greater security through strict policy definition and enforcement

IBM Security Access Manager for Enterprise Single Sign-On enforces strict, user-defined access security policies, and ensures that all users meet these policies by managing their access privileges. Security is not compromised in any way, because Administrators do not see the passwords of the users, they manage user access rights only.

This process significantly strengthens the security culture within an enterprise. IMS Connectors provide seamless enforcement of password change and fortification without user inconvenience.

Enforcement of regulatory compliance requirements

IBM Security Access Manager for Enterprise Single Sign-On strengthens information access and provides user-centric audit logs, while provisioning systems enforce access control for sensitive data. The Provisioning API and the selected provisioning system provide enterprises with a way to comply with regulations, such as Sarbanes-Oxley, the Gramm-Leach-Bliley Act, HIPAA, and the California SB 1386.

Improved employee productivity

IBM Security Access Manager for Enterprise Single Sign-On assists users in managing their credentials. When users launch an application, AccessAgent auto-fills their credentials and authenticates the users to the application server.

Integrating with another provisioning system provides convenience to users by consolidating credential management, so that Administrators can centrally manage and administer the consolidated accounts of users. Users do not have to remember any application passwords.

Streamlined user-centric administration

The integration eliminates the need for Administrators to manage too many application accounts. Administrators can automatically provision and administer multiple application accounts from one centralized system.

Integration also eliminates the need to update users on changes to application credentials since application credential changes are automatically updated to the Wallets. Users must know how to log on to the Wallet, and any new application credential automatically populates the Wallet.

Fast return on investment

The integration of Provisioning API with another provisioning system makes it possible for Administrators to manage multiple user accounts from one location.

The centralized process renders manual updates on individual accounts obsolete. The result is a significant reduction of time and costs associated with user account management.

Consequently, the time and costs saved can be channeled to other areas of network administration.

Integration solutions

IBM Security Access Manager for Enterprise Single Sign-On can integrate with various identity provisioning systems to provide a complete identity and access management solution.

Provisioning systems can integrate with IBM Security Access Manager for Enterprise Single Sign-On. For more information, see the Chapter 2, “Provisioning API,” on page 5.

Configuring the IMS Server for provisioning

Follow these steps to configure the IMS Server for provisioning.

Procedure

1. Use the IMS Configuration Utility (**Authentication Services**) to add the authentication services of the applications to be provisioned.
2. Use AccessAdmin (under **Authentication service policies**) to configure each authentication service to use the appropriate authentication modes. For example: Password
3. Configure the provisioning agent and facilitate authentication with the IMS Server before calling the provisioning services. Complete one of the following methods:

- Use the IMS Configuration Utility (**IMS Bridges**).

This authentication is done through using a shared secret between the provisioning agent and the IMS Server.

- Alternatively, configure the shared secret in the IMS Server configuration file <WAS profile directory>\config\tamesso\config\ims.xml with the following entries:

```
<auth.server.agent.clientIp>
<value xml:lang="en">192.0.2.1</value>
</auth.server.agent.clientIp>
<auth.server.agent.password>
<value xml:lang="en">sharedsecret</value>
</auth.server.agent.password>
```

The preceding configuration means that the provisioning agent with server ID agent is deployed on a machine with an IP address of 192.0.2.1.

It can log on to the IMS Server with server ID agent and password sharedsecret.

Important: Restart the IMS Server after completing the configuration. When the IMS Server starts, it encrypts the password and replaces the clear text password with the encrypted password in the configuration file.

Chapter 2. Provisioning API

These topics provide a reference for SOAP API, developing the SOAP client, and configuring the IMS Server.

These topics can also serve as a reference for the Java API, developing the integration module, and configuring the IMS Server.

See the following topics for more information:

- “About Provisioning API”
- “IBM Security Access Manager for Enterprise Single Sign-On Provisioning overview” on page 6
- “Provisioning API system requirements” on page 6
- “Available API types for Provisioning API” on page 6
- “Java API for provisioning” on page 7
- “SOAP API for provisioning” on page 28
- “Using Provisioning API for account setup and maintenance” on page 50

About Provisioning API

While the identity provisioning system provides the identity lifecycle management for application users, IBM Security Access Manager for Enterprise Single Sign-On provides real-time implementation of access security policies for users and applications.

The integrated solution delivers seamless identity and access management. This solution provides application account provisioning, a centralized view of all application accounts, sign-on and sign-off automation, authentication management, user-centric audit logs and reporting, and centralized de-provisioning of all accounts.

The identity provisioning system needs to communicate with the IMS Server to populate and manage credentials in the Wallet.

The IMS Bridge offers a Java interface to the identity provisioning system to communicate with the IMS Server. If it is not possible to use the Java interface, an IMS Bridge that communicates directly with the IMS Server by using SOAP must be developed.

Communications between the identity provisioning system and the IMS Server are done by using SOAP over HTTPS. When the identity provisioning system provisions new users or new accounts for a user application, resets application passwords for users, de-provisions an enterprise user or an application account, it makes appropriate SOAP calls to the IMS Server with relevant account data information.

The IMS Server accordingly creates a user, populates the Wallet of a user with the new account data, updates credentials, deletes accounts, or revokes a user.

With this integration, users can enjoy single sign-on to provisioned accounts immediately upon sign-on to IMS Server. Administrators can also revoke application access centrally by automatically removing credentials from the Wallet.

IBM Security Access Manager for Enterprise Single Sign-On Provisioning overview

After installing IBM Security Access Manager for Enterprise Single Sign-On, use the provisioning feature to manage users and accounts.

What to do	Where to find information
Learn about IBM Security Access Manager for Enterprise Single Sign-On provisioning features, benefits, and integration solutions.	Chapter 1, “Security Access Manager for Enterprise Single Sign-On Provisioning,” on page 1
Verify that the application programming interface (API) requirements are met.	“Provisioning API system requirements”
Use the keytool utility to create keystores or import certificates.	Appendix C, “Using the Keytool for provisioning,” on page 73
Choose one of the following API types for provisioning: • Java API for Provisioning • SOAP API for Provisioning	• “Java API for provisioning” on page 7 • “SOAP API for provisioning” on page 28
Perform basic tasks by using Provisioning API, such as adding application accounts, resetting application passwords, and de-provisioning users.	“Using Provisioning API for account setup and maintenance” on page 50

Provisioning API system requirements

View the system requirements for Provisioning API.

- IMS Server: 8.2
- (Java API only) IMS Bridge must be deployed with IBM JVM version 1.5.x

Available API types for Provisioning API

The developer can choose between two sets of application programming interfaces (APIs) to integrate an identity provisioning system with IBM Security Access Manager for Enterprise Single Sign-On.

- **Java API for Provisioning**

This API is for identity provisioning systems that support Java-based connectors for integration with third-party systems. The Java API provides a wrapper around the SOAP API to simplify its operations. For example, encryption of application passwords is performed by the provided IMS Bridge, and is then transparent to the developer.

- **SOAP API for Provisioning**

If the Java API cannot be used, the developer can choose to use the SOAP API instead. The advantage of the SOAP API is that it is independent of any programming language. The Provisioning Agent can potentially be written in any programming language native to the identity provisioning system.

Java API for provisioning

IBM Security Access Manager for Enterprise Single Sign-On provides a set of Java APIs for integration with identity provisioning systems.

The standard distribution of provisioning bridge contains the following directories:

Directory	Description
/bin	Contains binary executable files or scripts to invoke functions provided by the IMS Bridge
/lib	Contains libraries for IMS Bridge
/docs	Contains the configuration and deployment guides
/config	Contains sample configuration files for the IMS Bridge

See the following topics for more information about using the Java API with Provisioning API.

- “Certificate store configuration for the IMS Bridge (Java Provisioning API)”
- “IMS Bridge sample configuration file (Java Provisioning API)”
- “IMS Bridge configuration parameters (Java Provisioning API)” on page 8
- “Developing an integration module by using the IMS Bridge (Java Provisioning API)” on page 9
- “Using a command-line tool (CLT) to invoke the IMS Bridge API (Java Provisioning API)” on page 10
- “Java API classes” on page 12

Certificate store configuration for the IMS Bridge (Java Provisioning API)

The IMS Bridge communicates with the IMS Server by using one-way SSL. This means that the IMS Bridge needs to trust the IMS Server SSL certificate.

Import the IMS Server SSL certificate into the keystore as one trusted CA entry if you are deploying the IMS Bridge on an application server with one common truststore shared by different applications.

You can also create one keystore by using the Java key tool utility. Then, configure the IMS Bridge to use the previous trusted store.

Complete the steps in “Configuring the IMS Server for provisioning” on page 3.

IMS Bridge sample configuration file (Java Provisioning API)

This topic shows a sample configuration file for an IMS Bridge.

The IMS Bridge is packaged with a sample configuration file as follows:

```
#####  
# IMS connection information  
#####  
  
#Fully qualified IMS Server Name
```

```

ims.serverName=TARGET-IMS-SERVER

#IMS SSL port number
ims.httpsPort=443

# IMS HTTP port number
ims.httpPort=80

# IMS RUNTIME path
ims.servicePath=/ims/services

#####
# Misc. configs
#####

# PB truststore
provisioningbridge.trustStore=C:/PATH/TO/truststore.jks

# PB truststore password
provisioningbridge.trustStorePassword=password

# PB JVM initializer
provisioningbridge.jvm.environment.initializer=
    encentuate.bridges.provisioning.jvm.GenericJvmEnvironmentInitializer

# PB authservice mappings
provisioningbridge.authenticationService.mapping=lotusNotes:dir_notes

```

IMS Bridge configuration parameters (Java Provisioning API)

This topic contains the descriptions of the parameters. Some parameters are optional.

ims.serverName

The DNS name of the IMS Server.

ims.httpsPort

(Optional) The port IMS Server listens to for HTTPS request. The default is 443.

ims.httpPort

(Optional) The port IMS Server listens to for HTTP request. The default is 80.

ims.servicePath

(Optional) The root path of IMS Server services. The default is /ims/services/. Start the value with /.

provisioningbridge.trustStore

(Mandatory for CLT only) The truststore used by the IMS Bridge. For example: C:\path\to\truststore.

This configuration does not take effect if there is already one system property set for **javax.net.ssl.trustStore**.

provisioningbridge.trustStorePassword

(Mandatory for CLT only) Password of the truststore used by IMS Bridge.

This configuration does not take effect if there is already one system property set for **javax.net.ssl.trustStorePassword**.

provisioningbridge.authenticationService.mapping

(Optional) The mapping of application IDs on the host provisioning system to the representation of the IMS Server. The format of each value of this configuration key follows the format: prov_system_app_ID:IMS_server_app_ID.

For example, you have configured an authentication service for Active Directory in IMS Server called `dir_ibm.com`.

However, the internal representation for the same authentication service in your provisioning system is IBM Security Access Manager for Enterprise Single Sign-On. Include the following configuration key:

```
provisioningbridge.authenticationService.mapping = ISAMESS0:dir_ibm.com
```

provisioningbridge.jvm.environment.initializer

(Optional) Name of a class that implements the `JvmEnvironmentInitializer` interface that sets up the JVM environment such as Java system properties before IMS Bridge starts to run.

The default is:

```
encentuate.bridges.provisioning.jvm.GenericJvmEnvironmentInitializer
```

Developing an integration module by using the IMS Bridge (Java Provisioning API)

Use the IMS Bridge to develop an integration module.

About this task

Complete this procedure to integrate with the IMS Bridge after installing and configuring the IMS Bridge.

Procedure

1. Create an integration module by using the IMS Bridge Java API.
2. Compile the module.
3. Place all necessary classes in the JRE class path.

Example

The following sample code uses the IMS Bridge Java API. It creates and revokes a user with user name `james` in a deployment that does not have multiple Active Directory (AD) domains.

```
import encentuate.bridges.provisioning;
public class IntegrationModule {
    int userStatus;
    List appAccounts = new List();

    // Instantiates the Provisioning Bridge
    ProvisioningBridge bridge = new ProvisioningBridge("C:\\provisioningBridge.properties");

    // Login to the IMS Bridge using the shared secret.bridge.login
    ("provisioning_system", "xyz123 ");

    // Creates an account on the IMS Server for this user if he doesn't
    already have an one.
    bridge.createImsAccount("james", "abcabc");

    // Get registration status of the user account on the IMS Server.
    userStatus = bridge.getRegistrationStatus("james");

    // Once the IMS account is created, the provisioning system can app
    application account
    data to the credential Wallet of this IMS account.

    // Adds a "SQL DB" account with user name "james.h" to the user's
    credential Wallet.
    bridge.createAppAccountData("james", "SQL DB", "james.h", "abcdef");

    // Get a list of accounts in the user's credential Wallet.
    appAccounts = bridge.getUserAccounts("james");
```

```
// Deletes an application account data from the user's credential Wallet
bridge.deleteAppAccountData("james", "SQL DB", "james.h");

// Revokes the IMS user and invalidates his credential Wallet
bridge.revokeImsAccount("james");

// Logout from the IMS Bridge
bridge.logout();
```

Using a command-line tool (CLT) to invoke the IMS Bridge API (Java Provisioning API)

The IMS Bridge APIs can be invoked by using CLTs. To use a CLT, use a utility called **commons-launcher** in the /bin folder.

The following properties must be set for CLT use:

- **provisioningbridge.trustStore**
- **provisioningbridge.trustStorePassword**

For information about the configuration file parameters, see “IMS Bridge configuration parameters (Java Provisioning API)” on page 8.

For information about the command-line tool parameters in each of the following command-line examples, see “Parameters” on page 11.

To provision a new IMS Server user, you can issue a command in the command prompt as follows:

```
java -cp C:\provisioningbridge\bin LauncherBootstrap run \
--configFile C:\provisioningbridge\config\provisioningBridge.properties \
--loginId bridge --password password --task addImsUser \
--imsUserId test.example.com\user --imsUserPassword password \
--userPrincipalName user@test.example.com --samAccountName user \
--domainDnsName test.example.com
```

To list all active user enterprise IDs, you can issue a command in the command prompt as follows:

```
java -cp D:\provisioningbridge\bin LauncherBootstrap run \
--configFile D:\provisioningbridge\config\provisioningBridge.properties \
--loginId bridge --password password --task getAllActiveUserIds \
[--imsUserId domain*\user*]
```

To add an application account to the Wallet of an IMS Server user, you can issue a command in the command prompt as follows:

```
java -cp C:\provisioningbridge\bin LauncherBootstrap run --configFile \
C:\provisioningbridge\config\provisioningBridge.properties --loginId bridge \
--password password --task add --imsUserId test.example.com\user \
--appId dir_alpha.test.example.com --appUserId user \
--appPassword password
```

To reset the password of a user, you can issue a command in the command prompt as follows:

```
java -cp D:\provisioningbridge\bin LauncherBootstrap run \
--configFile D:\provisioningbridge\config\provisioningBridge.properties \
--loginId bridge --password password --task resetPassword \
--imsUserId test.example.com\prov5 --imsUserPassword new_p@ssw0rd
```

Note: Provisioning API resets only the ISAM ESSO password but not the Active Directory password.

To update an application account in the Wallet of an IMS Server user, you can issue a command in the command prompt as follows:

```
java -cp C:\provisioningbridge\bin LauncherBootstrap run \  
--configFile C:\provisioningbridge\config\provisioningBridge.properties \  
--loginId bridge --password password --task set \  
--imsUserId test.example.com\user \  
--appId dir_alpha.test.example.com --appUserId user \  
--appPassword password
```

To delete an application account that is stored in the Wallet of an IMS Server user, you can issue a command in the command prompt as follows:

```
java -cp C:\provisioningbridge\bin LauncherBootstrap run \  
--configFile C:\provisioningbridge\config\provisioningBridge.properties \  
--loginId bridge --password password --task delete \  
--imsUserId test.example.com\user \  
--appId dir_alpha.test.example.com --appUserId user
```

To check the registration status of a user, you can issue a command in the command prompt as follows:

```
java -cp C:\provisioningbridge\bin LauncherBootstrap run \  
--configFile C:\provisioningbridge\config\provisioningBridge.properties \  
--loginId bridge --password password --task registrationStatus \  
--imsUserId test.example.com\user
```

To get the Wallet contents of a user, you can issue a command in the command prompt as follows:

```
java -cp C:\provisioningbridge\bin LauncherBootstrap run \  
--configFile C:\provisioningbridge\config\provisioningBridge.properties \  
--loginId bridge --password password --task walletAccountInfo \  
--imsUserId test.example.com\user
```

To unlock a user account, you can issue a command in the command prompt as follows:

```
java -cp C:\provisioningbridge\bin LauncherBootstrap run \  
--configFile C:\provisioningbridge\config\provisioningBridge.properties \  
--loginId bridge --password password --task unlockAccount \  
--imsUserId test.example.com\user [--unlockEntDirAcc true/false]
```

To lock a user with the IMS Server user ID, you can issue a command in the command prompt as follows:

```
java -cp C:\provisioningbridge\bin LauncherBootstrap run \  
--configFile C:\provisioningbridge\config\provisioningBridge.properties \  
--loginId bridge --password password --task lockAccount \  
--imsUserId test.example.com\user
```

Note: IMS Bridge sets the specified IMS Server user attributes `nonCertAuthentication` to Disabled. IMS Bridge does not lock the enterprise directory account.

Parameters

The following are the different IMS Bridge command-line tool parameters with corresponding descriptions:

imsUserId

The user name for the account on the IMS Server. Use the same name as the primary enterprise identity of the user. In a deployment with multiple Active Directory domains, the format of the user name is of the form `dnsDomain\sAMAccountName`. For example: `test.example.com\user`

imsUserPassword

The password for the account on the IMS Server. For example: password

userPrincipalName

This parameter is the UPN of the user account and is only useful when enterprise directory is Active Directory. This parameter is optional.

Note: If the user account does not have an Active Directory **userPrincipalName** attribute, do not specify the **userPrincipalName** parameter.

samAccountName

This parameter is the **sAMAccountName** of the Active Directory user account and is only useful when enterprise directory is Active Directory. For example: user

domainDnsName

This parameter is the DNS domain name of the enterprise directory. For example: test.example.com

appId The unique identifier of the provisioning system for the application that this account data belongs. For example: dir_alpha.test.example.com

appUserId

The user name of the application account data to add. For example: user

appPassword

The password of the application account data to add. For example: password

loginId

The IMS Bridge login user name. For example: test.example.com\user

password

The password for the IMS Bridge credential. For example: password

task The provisioning operation you want to perform. For example: addImsUser

unlockEntDirAcc

Specifies whether to unlock the enterprise directory account. If the value is true, the IMS Bridge API attempts to unlock the enterprise directory account.

If the parameter is not specified, the IMS Bridge API looks up the password sync policy. If password sync is enabled, the IMS Bridge API unlocks the enterprise directory account along with the IMS Server user account. If password sync is disabled, only the IMS Server user account is unlocked. This parameter is optional.

Java API classes

The IMS Bridge interfaces with third-party identity provisioning systems through the **encentuate.bridges.provisioning.ProvisioningBridge** class.

ProvisioningBridge (Constructor)

Constructor of the class.

```
public ProvisioningBridge(String configFile)
throws ProvisioningBridgeException
```

- Parameters:

configFile

The full path name of the IMS Server configuration file.

- Throws:

ProvisioningBridgeException

If there are errors while reading the configuration file, this exception is thrown with an error code BAD_CONFIGURATION.

login

Authenticates the IMS Bridge to the IMS Server by using a pre-configured shared secret. This method must be called first before other methods in the API can be used.

```
public void login(String serverId, String serverPassword)
throws ProvisioningBridgeException, IllegalArgumentException
```

- Parameters:

serverId

The ID of the server (not case sensitive) on which the IMS Bridge is running. This parameter is pre-configured in the IMS Server as part of the shared secret for the IMS Bridge to log on to the IMS Server.

serverPassword

The corresponding case sensitive password for this shared secret.

- Throws:

ProvisioningBridgeException

If logon fails, with these possible error codes:

- INVALID_LOGIN

If the **serverId** and **serverPassword** pair does not match any pre-configured shared secret, or if the IP of this server does not match that pre-configured on the IMS Server.

- ACCESS_DENIED

If the logon account has no access to the IMS Server provisioning services.

- REMOTE_EXCEPTION

If there are exceptions when calling the IMS Server SOAP services.

logout

Logs out from the IMS Server at the end of the session.

```
public void logout()
```

- Parameters:

None

createImsAccount

This method creates an account on the IMS Server by using the given user name and initial password. User attributes are optional, depending on whether the deployment has multiple Active Directory domains. An IMS Server account is necessary for all the other provisioning tasks.

```
public void createImsAccount(String imsUserId, String initialPassword)
throws ProvisioningBridgeException
public void createImsAccount(String imsUserId, String initialPassword,
Hashtable userAttrs)
throws ProvisioningBridgeException
```

- Parameters:

imsUserId

The user name for the new account on the IMS Server. Use the same name as the primary enterprise identity of the user. In a deployment with multiple AD domains, the format of the user name is of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

initialPassword

The initial password to be set on the IMS Server for the new account.

userAttrs

A hash table that contains user attributes in name-value pairs. This parameter is to be supplied if the deployment has multiple AD domains. Currently, the following user attributes are supported:

- **userPrincipalName** (optional)
This parameter is the UPN of the user account and is only useful when enterprise directory is AD.
- **domainDnsName**
This parameter is the DNS domain name of the enterprise directory.
- **sAMAccountName**
This parameter is the **sAMAccountName** of the AD user account and is only useful when enterprise directory is AD.
- **directoryId** (optional)
This parameter is the ID of enterprise directory that holds the user account. This parameter is optional if there is only one enterprise directory configured in the IMS Server.

- Throws:

ProvisioningBridgeException

If a new account cannot be created on the IMS Server, with these possible error codes.

- **AUTHENTICATION_NEEDED**
If this method is called before the IMS Bridge has logged on to the IMS Server.
- **USER_IS_REGISTERED**
If **imsUserId** exists on the IMS Server.
- **USER_IS_REVOKED**
If **imsUserId** is registered but is already revoked.

deleteImsAccount

This method deletes a user account from the IMS Server and deletes the credential Wallet of the user. An account of the same user name can be created subsequent to this call.

```
public void deleteImsAccount(String imsUserId)
throws ProvisioningBridgeException
```

- Parameters:

imsUserId

The user name for the new account on the IMS Server. Use the same name as the primary enterprise identity of the user. In a deployment with multiple AD domains, the format of the user name is of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

- Throws:

ProvisioningBridgeException

If deletion fails, with these possible error codes:

- AUTHENTICATION_NEEDED
If this method is called before the IMS Bridge has logged on to the IMS Server.
- REMOTE_EXCEPTION
If invocation of the IMS Server service fails.
- USER_IS_REVOKED
If **imsUserId** has been revoked on the IMS Server.

revokeImsAccount

This method revokes a user account from the IMS Server and deletes the credential Wallet of the user. An IMS Server user account cannot be created with the same user name as a revoked account.

```
public void revokeImsAccount(String imsUserId)
throws ProvisioningBridgeException
```

- Parameters

imsUserId

The user name for the new account on the IMS Server. Use the same name as the primary enterprise identity of the user. In a deployment with multiple AD domains, the format of the user name is of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

- Throws:

ProvisioningBridgeException

If revocation fails, with these possible error codes:

- AUTHENTICATION_NEEDED
If this method is called before the IMS Bridge has logged on to the IMS Server.
- REMOTE_EXCEPTION
If invocation of the IMS Server service fails.

getRegistrationStatus

This method returns the registration status of a user account on the IMS Server.

```
public int getRegistrationStatus(String imsUserId)
throws ProvisioningBridgeExceptions
```

- Parameters:

imsUserId

The user name for the new account on the IMS Server. Use the same name as the primary enterprise identity of the user. In a deployment with multiple AD domains, the format of the user name is of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

- Returns:

`ResultCode.USER_NOT_REGISTERED`

User is not found on the IMS Server.

`ResultCode.USER_IS_REGISTERED`

User is found on the IMS Server and is registered.

`ResultCode.USER_IS_REVOKED`

User is found on the IMS Server but is revoked.

`ResultCode.DATASTORE_EXCP`

A database access error has occurred.

`ResultCode.NOT_OK`

The method could not return the registration status due to some error.

- Throws:

- **ProvisioningBridgeException**

- If the call fails, with these possible error codes:

- `AUTHENTICATION_NEEDED`

- If this method is called before the IMS Bridge has logged on to the IMS Server.

- `ProvisioningBridgeErrorCode.REMOTE_EXCEPTION`

- If there was an error on the server that prevented this call from succeeding.

addAppAccountData

This method adds application account data to the credential Wallet of the user. The account must not exist in the Wallet.

```
public void addAppAccountData(String imsUserId, String appId, String appIdUserid,
String appIdPassword)
throws ProvisioningBridgeException
```

- Parameters:

- **imsUserId**

- The user name for the new account on the IMS Server. Use the same name as the primary enterprise identity of the user. In a deployment with multiple AD domains, the format of the user name is of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

- **appId**

- The unique identifier of the provisioning system for the application to which this account data belongs.

- **appIdUserid**

- The user name of the application account data to add.

- **appIdPassword**

- The password of the application account data to add.

- Throws:

- **ProvisioningBridgeException**

- If the application account data cannot be set in the Wallet of the user, with these possible error codes:

- `AUTHENTICATION_NEEDED`

- If this method is called before the IMS Bridge has logged on to the IMS Server.

- `USER_NOT_REGISTERED`

- If **imsUserId** is not registered on the IMS Server.

- `USER_IS_REVOKED`

- If **imsUserId** has been revoked on the IMS Server.

- `ACCOUNT_ALREADY_EXISTS`

- If the account data exists.

- `UNKNOWN_AUTH_SERVICE`

- If **appId** did not map to a valid application identifier on the IMS Server.
- `ProvisioningBridgeErrorCode.REMOTE_EXCEPTION`
If there was an error on the server that prevented this call from succeeding.

updateAppAccountData

This method updates application account data on the credential Wallet of the user. The account must exist in the Wallet.

```
public void updateAppAccountData(String imsUserId, String appId,
String appUserId,String appPassword)throws ProvisioningBridgeException
```

- Parameters:

imsUserId

The user name for the new account on the IMS Server. Use the same name as the primary enterprise identity of the user. In a deployment with multiple AD domains, the format of the user name is of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

appId

The unique identifier of the provisioning system for the application to which this account data belongs.

appUserId

The user name of the application account data to update.

appPassword

The password of the application account data to update.

- Throws:

ProvisioningBridgeException

If the application account data cannot be set in the Wallet of the user, with these possible error codes:

- `AUTHENTICATION_NEEDED`
If this method is called before the IMS Bridge has logged on to the IMS Server.
- `USER_NOT_REGISTERED`
If **imsUserId** is not registered on the IMS Server.
- `USER_IS_REVOKED`
If **imsUserId** has been revoked on the IMS Server.
- `ACCOUNT_NOT_FOUND`
If the account data does not exist.
- `UNKNOWN_AUTH_SERVICE`
If **appId** did not map to a valid application identifier on the IMS Server.
- `ProvisioningBridgeErrorCode.REMOTE_EXCEPTION`
If there was an error on the server that prevented this call from succeeding.

deleteAppAccountData

This method deletes application account data from the credential Wallet of the user on the IMS Server.

```
public void deleteAppAccountData(String imsUserId, String appId,
String appUserId)
throws ProvisioningBridgeException
```

- Parameters:

imsUserId

The user name for the new account on the IMS Server. Use the same name as the primary enterprise identity of the user. In a deployment with multiple AD domains, the format of the user name is of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

appId

The unique identifier of the provisioning system for the application to which this account data belongs.

appUserId

The user name of the application account data to delete.

- Throws:

ProvisioningBridgeException

If the application account data cannot be set in the Wallet of the user, with these possible error codes:

- `AUTHENTICATION_NEEDED`
If this method is called before the IMS Bridge has logged on to the IMS Server.
- `USER_NOT_REGISTERED`
If **imsUserId** is not registered on the IMS Server.
- `USER_IS_REVOKED`
If **imsUserId** has been revoked on the IMS Server.
- `UNKNOWN_AUTH_SERVICE`
If **appId** did not map to a valid application identifier on the IMS Server.
- `ProvisioningBridgeErrorCode.REMOTE_EXCEPTION`
If there was an error on the server that prevented this call from succeeding.

getUserAccounts

This method returns a list of accounts in the credential Wallet of the user. Password accounts and accounts that are provisioned are returned.

```
public List getUserAccounts(String imsUserId)
throws ProvisioningBridgeException
```

- Parameters:

imsUserId

The user name for the new account on the IMS Server. Use the same name as the primary enterprise identity of the user. In a deployment with multiple AD domains, the format of the user name is of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

appId

The unique identifier of the provisioning system for the application to which this account data belongs.

appUserId

The user name of the application account data to delete.

- Returns:

Returns a list of accounts. If the user does not have any accounts, the list is empty. Otherwise, each entry in the list is an instance of Map object. The Map object contains selected information for one account and the following keys are available:

- `USER_ID`

User name of the account.

- **AUTHSVC_ID**

The authentication service ID of the account.

- Throws:

- ProvisioningBridgeException**

If the call fails, with these possible error codes:

- AUTHENTICATION_NEEDED

If this method is called before the IMS Bridge has logged on to the IMS Server.

- ProvisioningBridgeErrorCode.REMOTE_EXCEPTION

If there was an error on the server that prevented this call from succeeding.

getAllActiveUserIDs

This method returns a list of all active user enterprise IDs.

```
public List<String> getAllActiveUserIDs(String filter)
throws ProvisioningBridgeException
```

- Parameters:

- filter**

If set to asterisk (*), this parameter returns all users.

Note: An asterisk (*) can also be used to represent any number and set of characters. For example, setting the filter parameter to a*h, returns all names beginning with 'a' and ending with 'h', namely, Ashish, Amitabh, Ajitabh, and so on.

- Returns:

A list of active user enterprise IDs.

- **List<String>**

List of active user enterprise IDs.

- Throws:

- ProvisioningBridgeException**

If the call fails, with these possible error codes:

- AUTHENTICATION_NEEDED

If this method is called before the IMS Bridge has logged on to the IMS Server.

resetPassword

This method resets the password of the given IMS Server user ID.

Note: Provisioning API resets only the SAM E-SSO Password but not the AD password.

```
public int resetPassword(String imsUserId, String newPassword)
throws ProvisioningBridgeException
```

- Prerequisite:

Before you call this method, navigate to **AccessAdmin>System Policies>Sign up Policies>Option for specifying secret**. Choose the value **Secret not required**.

- Parameters:

- imsUserId**

The user name for the new account on the IMS Server. Use the same name as the primary enterprise identity of the user. In a deployment with multiple AD domains, the format of the user name is of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

newPassword

New password of the given user.

- Returns:

`ResultCode.OK`

If the password was reset successfully.

`ResultCode.ILLEGAL_ARGUMENT`

If any argument is null or empty.

`ResultCode.INVALID_ENCRYPTED_PASSWORD`

If the password is not in accordance with the password policy.

`ResultCode.SECRET_NOT_REQUIRED_DISABLED`

If the IMS Server setup policy "Option for specifying secret" is not set to "Secret not required" (`pid_secret_option=0`).

`ResultCode.NO_AUTHENTICATOR_SOCI`

If the system is unable to get Authenticator Soci from data store.

`ResultCode.UNABLE_TO_DECRYPT_RANDOMIZED_SECRET`

If the system is unable to decrypt randomized secret.

`ResultCode.GET_CIPHER_DATA_ERROR`

If the system is unable to get raw encrypted CSK and algorithm.

`ResultCode.UNABLE_TO_DECRYPT_CSK`

If the system is unable to decrypt CSK.

`ResultCode.UNABLE_TO_ENCRYPT_CSK`

If the system is unable to encrypt CSK

`ResultCode.USER_NOT_REGISTERED`

If the given user does not exist in system records.

`ResultCode.DATASTORE_EXCP`

If any data store exception results.

- Throws:

ProvisioningBridgeException

If the call fails, with these possible error codes:

- `AUTHENTICATION_NEEDED`

If this method is called before the IMS Bridge has logged on to the IMS Server.

addAccountCredentialWithAccountFields

This method adds the provisioned account credentials with multiple fields.

```
public int addAccountCredentialWithAccountFields(String sessionId,
        String enterpriseId, String authId, String accountType,
        NameValue[] accountFields)
```

- Parameters:

sessionId

The session ID of the provisioning agent.

enterpriseID

The enterprise ID or the IBM Security Access Manager for Enterprise Single Sign-On user name of the user to be provisioned. In a deployment with multiple Active Directory domains, the format of the user name must be of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

authID

The authentication service ID of the user account to be provisioned.

accountType

The type of account to be provisioned: Password, OTP, MAC, or Certificate.

accountFields

The account data item template ID to apply. The account data item template defines the properties of individual account data items. For example: `aditi_cipwd`. For more information about account data item templates, see the *AccessStudio Guide*.

- Returns:

`DATASTORE_EXCEPTION`

If there are database errors during loading of the requested authentication service.

`UNKNOWN_AUTH_SERVICE`

If the requested authentication service does not exist.

`INVALID_AUTH_SERVICE`

If the requested authentication service in IMS DB cannot be parsed.

`INVALID_ACCOUNT_DATA_TEMPLATE`

If the account data template specified for the `authId` does not match the fields given as inputs.

`BAD_INPUT_PARAMETERS`

If an input is null or empty.

`USER_NOT_REGISTERED`

If the user is not yet registered.

`UNSUPPORTED_AUTH_MECHANISM`

If authentication service does not support password authentication.

`UNKNOWN_DATA_STORAGE_TEMPLATE`

If there is an error obtaining the storage template for an authentication service.

`UNKNOWN_ACCOUNT_DATA_TEMPLATE`

If there is an error obtaining the account data template for an authentication service.

`ACCOUNT_ALREADY_EXISTS`

If the account being added exists for a user.

`NOT_OK`

If some error occurs when setting the account data.

addAccountCredentialWithSecondKey

This method adds an account credential with three fields: username, password, and domain.

```
public int addAccountCredentialWithAccountFields(String sessionId,
        String enterpriseId, String authId, String accountType,
        NameValue[] accountFields)
```

- Parameters:

sessionID

The session ID of the provisioning agent.

enterpriseID

The enterprise ID or the IBM Security Access Manager for Enterprise Single Sign-On user name of the user to be provisioned. In a deployment with multiple Active Directory domains, the format of the user name must be of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

adminEntId

The Enterprise ID of the admin who created the user.

authID

The authentication service ID of the user account to be provisioned.

accountType

The type of account to be provisioned: Password, OTP, MAC, or Certificate.

username

The user name of the account to be added.

domain

The domain of the account to be added.

accountFields

The account data item template ID to associate. The account data item template defines the properties of individual account data items. For example: `aditi_cipwd`. For more information about account data item templates, see the *AccessStudio Guide*.

- Returns:

`DATASTORE_EXCEPTION`

If there are database errors during loading of the requested authentication service.

`UNKNOWN_AUTH_SERVICE`

If the requested authentication service does not exist.

`INVALID_AUTH_SERVICE`

If the requested authentication service in IMS DB cannot be parsed.

`INVALID_ACCOUNT_DATA_TEMPLATE`

If the account data template specified for the `authId` does not match the fields given as inputs.

`BAD_INPUT_PARAMETERS`

If an input is null or empty.

`USER_NOT_REGISTERED`

If the user is not yet registered.

`UNSUPPORTED_AUTH_MECHANISM`

If authentication service does not support password authentication.

`UNKNOWN_DATA_STORAGE_TEMPLATE`

If there is an error obtaining the storage template for an authentication service.

`UNKNOWN_ACCOUNT_DATA_TEMPLATE`

If there is an error obtaining the account data template for an authentication service.

`ACCOUNT_ALREADY_EXISTS`

If the account being added exists for a user.

`NOT_OK`

If some error occurs when setting the account data.

removeAccountCredentialWithAccountFields

This method removes provisioned account credentials with multiple fields.

```
public int removeAccountCredentialWithAccountFields(String sessionId,
    String enterpriseId, String authId, String accountType,
    NameValue[] accountFields)
```

- Parameters:

sessionId

The session ID of the provisioning agent.

enterpriseId

The enterprise ID or the IBM Security Access Manager for Enterprise Single Sign-On user name of the user to be provisioned. In a deployment with multiple Active Directory domains, the format of the user name must be of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

authId

The authentication service ID of the user account to be provisioned.

accountType

The type of account to be provisioned: Password, OTP, MAC, or Certificate.

accountFields

The account data item template ID. The account data item template defines the properties of individual account data items. For example: `aditi_cipwd`. For more information about account data item templates, see the *AccessStudio Guide*.

- Returns:

DATASTORE_EXCEPTION

If there are database errors during loading of the requested authentication service.

UNKNOWN_AUTH_SERVICE

If the requested authentication service does not exist.

INVALID_AUTH_SERVICE

If the requested authentication service in IMS DB cannot be parsed.

INVALID_ACCOUNT_DATA_TEMPLATE

If the account data template specified for the `authId` does not match the fields given as inputs.

BAD_INPUT_PARAMETERS

If an input is null or empty.

USER_NOT_REGISTERED

If the user is not yet registered.

UNSUPPORTED_AUTH_MECHANISM

If authentication service does not support password authentication.

UNKNOWN_DATA_STORAGE_TEMPLATE

If there is an error obtaining the storage template for an authentication service.

UNKNOWN_ACCOUNT_DATA_TEMPLATE

If there is an error obtaining the account data template for an authentication service.

ACCOUNT_ALREADY_EXISTS

If the account being added exists for a user.

NOT_OK

If some error occurs when setting the account data.

removeAccountCredentialWithSecondKey

This method removes an account credential with three fields: username, password, and domain. The account must exist.

```
public int removeAccountCredentialWithSecondKey(String sessionId,
        String enterpriseId, String authId, String accountType,
        String username, String secondKey)
```

- Parameters:

sessionId

The session ID of the provisioning agent.

enterpriseId

The enterprise ID or the IBM Security Access Manager for Enterprise Single Sign-On user name of the user to be provisioned. In a deployment with multiple Active Directory domains, the format of the user name must be of the form dnsDomain\sAMAccountName. For example: example.com\user1.

authId

The authentication service ID of the user account to be provisioned.

accountType

The type of account to be provisioned: Password, OTP, MAC, or Certificate.

username

The user name of the account to be added.

domain

The domain of the account to be added.

- Returns:

DATASTORE_EXCEPTION

If there are database errors during loading of the requested authentication service.

UNKNOWN_AUTH_SERVICE

If the requested authentication service does not exist.

INVALID_AUTH_SERVICE

If the requested authentication service in IMS DB cannot be parsed.

INVALID_ACCOUNT_DATA_TEMPLATE

If the account data template specified for the authId does not match the fields given as inputs.

BAD_INPUT_PARAMETERS

If an input is null or empty.

USER_NOT_REGISTERED

If the user is not yet registered.

UNSUPPORTED_AUTH_MECHANISM

If authentication service does not support password authentication.

UNKNOWN_DATA_STORAGE_TEMPLATE

If there is an error obtaining the storage template for an authentication service.

UNKNOWN_ACCOUNT_DATA_TEMPLATE

If there is an error obtaining the account data template for an authentication service.

ACCOUNT_ALREADY_EXISTS

If the account being added exists for a user.

NOT_OK

If some error occurs when setting the account data.

setAccountCredentialWithAccountFields

This method sets provisioned account credentials with multiple fields.

```
public int setAccountCredentialWithAccountFields(String sessionId,
        String enterpriseId, String authId, String accountType,
        NameValue[] accountFields)
```

- Parameters:

sessionId

The session ID of the provisioning agent.

enterpriseId

The enterprise ID or the IBM Security Access Manager for Enterprise Single Sign-On user name of the user to be provisioned. In a deployment with multiple Active Directory domains, the format of the user name must be of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

authId

The authentication service ID of the user account to be provisioned.

accountType

The type of account to be provisioned: Password, OTP, MAC, or Certificate.

accountFields

The account data item template ID. The account data item template defines the properties of individual account data items. For example: `aditi_cipwd`. For more information about account data item templates, see the section on “account data item templates” in the *AccessStudio Guide*.

- Returns:

DATASTORE_EXCEPTION

If there are database errors during loading of the requested authentication service.

UNKNOWN_AUTH_SERVICE

If the requested authentication service does not exist.

INVALID_AUTH_SERVICE

If the requested authentication service in IMS DB cannot be parsed.

INVALID_ACCOUNT_DATA_TEMPLATE

If the account data template specified for the `authId` does not match the fields given as inputs.

BAD_INPUT_PARAMETERS

If an input is null or empty.

USER_NOT_REGISTERED

If the user is not yet registered.

UNSUPPORTED_AUTH_MECHANISM

If authentication service does not support password authentication.

UNKNOWN_DATA_STORAGE_TEMPLATE

If there is an error obtaining the storage template for an authentication service.

UNKNOWN_ACCOUNT_DATA_TEMPLATE

If there is an error obtaining the account data template for an authentication service.

ACCOUNT_ALREADY_EXISTS

If the account being added exists for a user.

NOT_OK

If some error occurs when setting the account data.

setAccountCredentialWithSecondKey

This method updates an account credential with three fields: username, password, and domain. The account must exist.

```
public int setAccountCredentialWithSecondKey(String sessionId,
      String enterpriseId, String authId, String accountType,
      String username, String password, String secondKey)
```

- Parameters:

sessionId

The session ID of the provisioning agent.

enterpriseId

The enterprise ID or the IBM Security Access Manager for Enterprise Single Sign-On user name of the user to be provisioned. In a deployment with multiple Active Directory domains, the format of the user name must be of the form dnsDomain\sAMAccountName. For example: example.com\user1.

adminEntId

The Enterprise ID of the admin who created the user.

authId

The authentication service ID of the user account to be provisioned.

accountType

The type of account to be provisioned: Password, OTP, MAC, or Certificate.

username

The user name of the account to be added.

domain

The domain of the account to be added.

- Returns:

DATASTORE_EXCEPTION

If there are database errors during loading of the requested authentication service.

UNKNOWN_AUTH_SERVICE

If the requested authentication service does not exist.

INVALID_AUTH_SERVICE

If the requested authentication service in IMS DB cannot be parsed.

INVALID_ACCOUNT_DATA_TEMPLATE

If the account data template specified for the authId does not match the fields given as inputs.

BAD_INPUT_PARAMETERS

If an input is null or empty.

USER_NOT_REGISTERED

If the user is not yet registered.

UNSUPPORTED_AUTH_MECHANISM

If authentication service does not support password authentication.

UNKNOWN_DATA_STORAGE_TEMPLATE

If there is an error obtaining the storage template for an authentication service.

UNKNOWN_ACCOUNT_DATA_TEMPLATE

If there is an error obtaining the account data template for an authentication service.

ACCOUNT_ALREADY_EXISTS

If the account being added exists for a user.

NOT_OK

If some error occurs when setting the account data.

unlockAccount

This method unlocks the user with a given IMS Server user ID.

```
public int unlockAccount (String sessionId, String enterpriseId,  
                        String adminEntId, String unlockEntDirAccount)
```

- Parameters:

sessionId

The session ID of the provisioning agent.

enterpriseID

The enterprise ID or the IBM Security Access Manager for Enterprise Single Sign-On user name of the user to be provisioned. In a deployment with multiple Active Directory domains, the format of the user name must be of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

adminEntId

The Enterprise ID of the admin who created the user.

- Returns:

`ResultCode.OK`

If the password was reset successfully.

`ResultCode.UNLOCK_WALLET`

Account unlocked successfully, but unlocking enterprise directory account failed.

`ResultCode.UNLOCK_ENTERPRISE`

Unlocking enterprise directory account succeeded, but unlocking account failed.

lockAccount

This method locks the user with a given IMS Server user ID.

```
public int lockAccount (String sessionId, String enterpriseId, String adminEntId)
```

- Parameters:

sessionId

The session ID of the provisioning agent.

enterpriseID

The enterprise ID or the IBM Security Access Manager for Enterprise Single Sign-On user name of the user to be provisioned. In a deployment with multiple Active Directory domains, the format of the user name must be of the form `dnsDomain\sAMAccountName`. For example: `example.com\user1`.

adminEntId

The Enterprise ID of the admin who created the user.

- Returns:
 `ResultCode.OK`
 If the lock is successful.
 `ResultCode.NOT_OK`
 If the lock fails.

SOAP API for provisioning

Simple Object Access Protocol or SOAP is a protocol for exchanging XML-based messages over a computer network, normally by using HTTP. SOAP forms the foundation layer of the Web services stack, providing a basic messaging framework that more abstract layers can build on.

SOAP services are defined by using Web Services Definition Language (WSDL) and are accessible by using a URL that is known as a SOAP endpoint.

IBM Security Access Manager for Enterprise Single Sign-On provides a SOAP API for identity provisioning systems to communicate with the IMS Server to perform provisioning. With the SOAP API, the request interface is an object in the native programming language of your application.

A third-party SOAP client can be used to generate business-object interfaces and network stubs from a WSDL document that specifies the IMS Server message schema, the service address, and other information.

The SOAP client handles the details of building the SOAP request and sending it to the IMS Server. Your application works with data in the form of object properties, and it sends and receives the data by calling object methods.

Identity provisioning for IBM Security Access Manager for Enterprise Single Sign-On requires the use of two sets of SOAP APIs:

- **API for server authentication**

For the provisioning agent to log on and log off the IMS Server. Provisioning agents must log on to the IMS Server before invoking other API operations.

- **API for provisioning service**

For creating, deleting, or revoking IBM Security Access Manager for Enterprise Single Sign-On users, and adding, setting, or removing application account credentials.

A typical identity provisioning system contains provisioning agents for provisioning users and applications on third-party systems. It is assumed that the provisioning agent would be using the SOAP API to integrate with the IMS Server.

The provisioning agent first sets up an IMS Server session by logging on to the IMS Server. It provisions IBM Security Access Manager for Enterprise Single Sign-On users by specifying their user names and initial passwords. It can also provision application credentials by specifying the application user names and passwords.

When necessary, the provisioning agent can also call the appropriate operations to reset application passwords, remove application credentials, and delete or revoke IBM Security Access Manager for Enterprise Single Sign-On users. Finally, the provisioning agent terminates the session by logging off the IMS Server.

See the following topics for more information.

- “Usage types (SOAP API Provisioning)”
- “SOAP client development” on page 30
- “IMS Server configuration for SOAP API” on page 31
- “SOAP API data types for Provisioning API” on page 32
- “SOAP API operations” on page 35

Usage types (SOAP API Provisioning)

See these usage types for SOAP API Provisioning.

The following are the usage types detailed in this topic.

- “Provisioning a typical user (SOAP API Provisioning)”
- “Resetting an application password for a user (SOAP API Provisioning)”
- “Removing application account credentials for a user (SOAP API Provisioning)”
- “Deleting or revoking a user (SOAP API Provisioning)” on page 30
- “Checking the status of a user and obtaining the list of accounts in the credential Wallet (SOAP API Provisioning)” on page 30
- “Getting all active user enterprise IDs (SOAP API Provisioning)” on page 30
- “Resetting password (SOAP API Provisioning)” on page 30

Provisioning a typical user (SOAP API Provisioning)

Follow these steps to provision a typical user.

Procedure

1. Call **loginByPassword** to log on to the IMS Server.
2. Call **preProvisionImsUser** to provision an IBM Security Access Manager for Enterprise Single Sign-On user.
3. Call **createWallet** to create a Wallet for the IBM Security Access Manager for Enterprise Single Sign-On user.
4. Call **getProvisioningCert** to obtain the provisioning certificate of the user that is used to encrypt application passwords.
5. Call **addAccountCredential** for each application to be provisioned for the user.
6. Call **terminateSession** to log off from the IMS Server.

Resetting an application password for a user (SOAP API Provisioning)

Follow these steps to reset an application password for a user.

Procedure

1. Call **loginByPassword** to log on to the IMS Server.
2. Call **getProvisioningCert** to obtain the provisioning certificate of the user that is used to encrypt application passwords.
3. Call **setAccountCredential** for the application.
4. Call **terminateSession** to log off from the IMS Server.

Removing application account credentials for a user (SOAP API Provisioning)

Follow these steps to remove application account credentials for a user.

Procedure

1. Call **loginByPassword** to log on to the IMS Server.
2. Call **removeAccountCredential** for the application.
3. Call **terminateSession** to log off from the IMS Server.

Deleting or revoking a user (SOAP API Provisioning)

Follow these steps to delete or revoke a user.

Procedure

1. Call **loginByPassword** to log on to the IMS Server.
2. Call **deleteImsAccount** or **revokeImsAccount** to delete or revoke the IBM Security Access Manager for Enterprise Single Sign-On user.
3. Call **terminateSession** to log off from the IMS Server.

Checking the status of a user and obtaining the list of accounts in the credential Wallet (SOAP API Provisioning)

Follow these steps to check the status of a user and obtain the list of accounts in the credential Wallet.

Procedure

1. Call **loginByPassword** to log on to the IMS Server.
2. Call **getRegistrationStatus** to check whether the user is registered. If registered, call **getUserAccounts** to obtain the list of accounts in the credential Wallet.
3. Call **terminateSession** to log off from the IMS Server.

Getting all active user enterprise IDs (SOAP API Provisioning)

Follow these steps to obtain all active user enterprise IDs.

Procedure

1. Call **loginByPassword** to log on to the IMS Server.
2. Call **getAllActiveUserIDs** for the application.
3. Call **terminateSession** to log off from the IMS Server.

Resetting password (SOAP API Provisioning)

Follow these steps to reset the password.

Procedure

1. Call **loginByPassword** to log on to the IMS Server.
2. Call **resetPassword** for the application.
3. Call **terminateSession** to log off from the IMS Server.

SOAP client development

The provisioning agent (IMS Bridge) must be developed as a SOAP client.

The SOAP API has been tested on the following platforms:

- Microsoft .NET with Visual Studio .NET
- Apache Axis

Other SOAP client environments can be used as long as they support standard SOAP messages.

SOAP tools consume WSDL to generate SOAP client code. The WSDL for this API can be obtained from an installed IMS Server at the following URLs. Replace `imsserver` with the host name of your IMS Server.

- `https://imsserver/ims/services/ibme.ims.service.ProvisioningService?wsdl`
- `https://imsserver/ims/services/ibm.ims.service.ServerAuthentication?wsdl`

See Appendix A, “WSDL for server authentication,” on page 61 and Appendix B, “WSDL for provisioning service,” on page 63 for the list of WSDLs.

Developing the SOAP client by using Visual Studio .NET

Follow these steps to develop the SOAP client by using Visual Studio .Net

Procedure

1. Add a Web Reference, directing it to the WSDL for Server Authentication. Name it `ImserverAuthentication`.
2. Add a Web Reference, directing it to the WSDL for Provisioning Service. Name it `ImprovisioningService`.
3. In the appropriate code, create an `ImserverAuthentication.ServerAuthenticationService` object.
4. In the appropriate code, create an `ImprovisioningService.ProvisioningServiceService` object.
5. When needed, call the methods of the objects.

Developing the SOAP client by using Apache Axis

Follow these steps to develop the SOAP client by using Apache Axis.

Procedure

1. Use **WSDL2Java** to automatically create Java stubs and classes.
2. In the appropriate code, create an `EncentuateServerAuthenticationServerAuthenticationSoapBindingStub` object, directing it to the WSDL for Server Authentication.
3. In the appropriate code, create an `EncentuateProvisioningServiceProvisioningServiceSoapBindingStub` object, directing it to the WSDL for Provisioning Service.
4. When needed, call the methods of the objects.

IMS Server configuration for SOAP API

Configure the IMS Server for SOAP API.

To configure the IMS Server for provisioning, see the procedure in “Configuring the IMS Server for provisioning” on page 3.

Take note the following:

- The provisioning agent communicates with the IMS Server by using one-way SSL. Consequently, the provisioning agent must trust the SSL certificate of the IMS Server.

If you are deploying the provisioning agent on an application server, where there is already a common truststore shared by different applications, import the SSL certificate of the IMS Server into the keystore as one trusted certificate authority entry.

On the Java platform, you can create a keystore by using the Java key tool utility. On the Visual Studio .NET platform, you can store the SSL certificate of the IMS Server in the Windows certificate store by using the Certificates snap-in. You can use Internet Explorer to download the SSL certificate of the IMS Server into the Windows certificate store.

- Visit <https://imsserver/> where imsserver is the host name of the IMS Server.
- Click **View Certificate** and install the certificate in the **Local Computer** certificate store.

Select **Install Certificate > Next > Place all certificates in the following store > Browse > Trusted Root Certification Authorities > Show physical stores > Local Computer > OK > Next > Finish**.

- Ensure that the SOAP client, or the provisioning agent, specifies the correct provisioning agent name, shared secret, authentication service IDs, and account types when using the API.

SOAP API data types for Provisioning API

Use this topic to view the data types and values that are used by this API.

resultCode

```
<element name="resultCode" type="xsd:int" />
```

- **Synopsis**

The result code that is returned by an operation to indicate the success of the operation or the reason for failure.

- **Value**

Identifier	Value (Hex)	Description
OK	0x00000000	Success.
NOT_OK	0x50000001	Generic failure.
BAD_CONFIGURATION	0x13000110	Bad IMS Server configuration.
ACCESS_DENIED	0x53000220	Shared secret is not correct or SOAP client IP address is not allowed.
DATASTORE_EXCP	0x23005000	A database error has occurred.
BAD_INPUT_PARAMETERS	0x53000320	An input parameter is null or empty.
GET_CIPHER_DATA_ERROR	0x53008409	The system is unable to get the raw encrypted CSK and algorithm.
ILLEGAL_ARGUMENT	0x53008405	The required argument is null or invalid.
INVALID_ENCENTUATE_PASSWORD	0x53000305	User submits an IBM Security Access Manager for Enterprise Single Sign-On password that does not satisfy relevant policies such as minimum length, and so on.
INVALID_LOGIN_CLIENT_IP_MISSING	0x53000254	Unable to log on because the SOAP client IP address cannot be found.
INVALID_SESSION	0x53000200	Session ID is not valid.

Identifier	Value (Hex)	Description
UNKNOWN_AUTH_SERVICE	0x53008101	The requested authentication service does not exist.
INVALID_AUTH_SERVICE	0x53008150	The requested authentication service has not been correctly defined.
INVALID_ACCOUNT_DATA_TEMPLATE	0x53008151	The account data template specified for the authentication service does not match the fields given as inputs.
SECRET_NOT_REQUIRED_DISABLED	0x53008406	The pid_secret_option is not set to zero.
USER_IS_REGISTERED	0x53000282	The specified user is registered.
USER_IS_REVOKED	0x53000259	The specified user has been revoked.
USER_NOT_REGISTERED	0x53000284	The specified user is not yet registered.
UNSUPPORTED_AUTH_MECHANISM	0x53008251	The authentication service does not support authentication by password.
UNKNOWN_DATA_STORAGE_TEMPLATE	0x53008301	There is an error obtaining the storage template for the authentication service.
ACCOUNT_ALREADY_EXISTS	0x53008402	The account being added exists for the user.
ENTRY_NOT_FOUND	0x23005530	The specified user is not found.
UNEXPECTED_WARNING	0x23000002	An encryption error has occurred.
UNKNOWN_ACCOUNT_DATA_TEMPLATE	0x53008102	There is an error obtaining the account data template for the authentication service.
ACCOUNT_NOT_FOUND	0x53008403	The account to be deleted does not exist.
NO_AUTHENTICATOR_SOCI	0x53008407	No authenticator Soci is found.
UNABLE_TO_DECRYPT_RANDOMIZED_SECRET	0x53008408	The system is unable to decrypt the random secret.
UNABLE_TO_DECRYPT_CSK	0x5300840A	The system is unable to decrypt CSK.
UNABLE_TO_ENCRYPT_CSK	0x5300840B	The system is unable to encrypt to CSK.

resultString

```
<element name="resultString" nillable="true" type="xsd:string"/>
```

- **Synopsis**

The result string that is returned by an operation to indicate the output of the operation.

- **Value**

Depends on the operation.

ResultMessage

```
<complexType name="ResultMessage">
  <sequence>
    <element name="resultCode" type="xsd:int" />
    <element name="resultString" nillable="true" type="xsd:string" />
  </sequence>
</complexType>
<element name="ResultMessage" nillable="true" type="tns1:ResultMessage" />
```

- **Synopsis**

The composite result message that is returned by an operation to indicate the success or failure of the operation, as well as its output.

- **Value**

Depends on the operation.

NameValue

```
<complexType name="NameValue">
  <sequence>
    <element name="name" nillable="true" type="xsd:string"/>
    <element name="value" nillable="true" type="xsd:string"/>
  </sequence>
</complexType><complexType name="ResultMessage">
  <sequence>
    <element name="resultCode" type="xsd:int" />
    <element name="resultString" nillable="true" type="xsd:string" />
  </sequence>
</complexType>
<element name="ResultMessage" nillable="true" type="tns1:ResultMessage" />
```

- **Synopsis**

Name value pair for a user attribute.

- **Value**

For example, to set a mobile phone number of a user to receive Mobile ActiveCodes, the name is to be set to **gsmNumber**. The value is to be set to a mobile phone number of the format 1-617-12345678.

ResultArrayMap

```
<complexType name="ResultArrayMap">
  <sequence>
    <element name="maps" nillable="true" type="impl:ArrayOf_apachesoap_Map"/>
    <element name="resultCode" type="xsd:int"/>
  </sequence>
</complexType>
<element name="ResultArrayMap" nillable="true" type="tns1:ResultArrayMap"/>
<complexType name="ArrayOf_apachesoap_Map">
  <complexContent>
    <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="apachesoap:Map[]" />
    </restriction>
  </complexContent>
</complexType>
<complexType name="Map">
  <sequence>
    <element maxOccurs="unbounded" minOccurs="0" name="item">
      <complexType>
        <all>
          <element name="key" type="xsd:anyType"/>
          <element name="value" type="xsd:anyType"/>
        </all>
      </complexType>
    </element>
  </sequence>
</complexType>
```

```

</complexType>
</element>
</sequence>
</complexType>

```

- **Synopsis**

The composite result message that is returned by an operation to indicate the success or failure of the operation, as well as an array of Maps as its output.

- **Value**

Depends on the operation.

ResultStringList

```

<complexType name="ResultStringList">
  <sequence>
    <element name="errCode" type="xsd:int"/>
    <element name="stringList" nillable="true" type="impl:ArrayOf_xsd_string"/>
  </sequence>
</complexType>

```

- **Synopsis**

The composite result message that is returned by an operation to indicate the success or failure of the operation, and an array of string as its output.

- **Value**

Depends on the operation.

SOAP API operations

Use these topics to review the SOAP API operations for server authentication and provisioning services.

See the following topics for more information.

- “SOAP API operations for server authentication (Provisioning API)”
- “SOAP API operations for provisioning service (Provisioning API)” on page 36

SOAP API operations for server authentication (Provisioning API)

Use this topic to view all the operations that are offered by the API for server authentication.

loginByPassword

```
public ResultMessage loginByPassword(String serverId, String password);
```

- **Synopsis**

To log on to the IMS Server as a provisioning agent.

This operation requests the IMS Server to issue a session ID that can be used in other provisioning operations.

- **Arguments**

Argument name	Description
serverId	The name of the provisioning agent. This name corresponds to the provisioning agent name setting in the IMS Server configuration file. For more information, see “Configuring the IMS Server for provisioning” on page 3.
password	The shared secret between the provisioning agent and the IMS Server.

- **Return Value**

Returns ResultMessage that consists of resultCode and resultString.

Returns session ID in resultString if resultCode is OK.
Returns error message in resultString if resultCode is OK.

- **Remarks**

Use this operation to log on to the IMS Server before other operations of this API can be invoked. The session ID returned by this operation is to be used in other operations of this API.

terminateSession

```
public int terminateSession(String sessionKey);
```

- **Synopsis**

To terminate the session.

This operation requests the IMS Server to terminate the session for the specified **sessionId**.

- **Arguments**

Argument name	Description
sessionKey	The session ID of the session to be terminated.

- **Return Value**

Returns resultCode.

- **Remarks**

Use this operation to terminate the session after all required user provisioning operations have been performed.

SOAP API operations for provisioning service (Provisioning API)

Use this topic to view all the operations that are offered by the API for provisioning service. Before using any of these operations, a session ID must already be obtained through the **loginByPassword** operation of the API for server authentication.

addAccountCredential

```
public int addAccountCredential(String sessionId, String enterpriseId,  
String authId, String accountType, String username, String password);
```

- **Synopsis**

To add provisioned account credentials for a user.

This operation requests the IMS Server to add a set of application account credentials for an existing IBM Security Access Manager for Enterprise Single Sign-On user. For security purposes, the password must be encrypted by using the provisioning certificate of the user.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID or the IBM Security Access Manager for Enterprise Single Sign-On user name of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
authId	The authentication service ID of the user account to be provisioned.
accountType	The type of account to be provisioned: Password, OTP, MAC, or Certificate.

Argument name	Description
username	The application user name of the user account to be provisioned.
password	The password in Base64, encrypted by the provisioning encryption certificate of the user in an XML snippet of the following form: <pre><password transformation="RSA/None/PKCS1Padding/2048/ProvisionKeypair"> QtOpkL0eWD+7vMPLUIVYHJFHijY+2ggvm1C26ra0XMZBURrbQRbg vXeI4SA5tuh7EuBkJWjC/fhivpBqmz2Nm1ersSUFZ4IQxYe0EXD txBmkSF149I/eieUVzhVcyvrzkP276FPX4Y01Miz/S4fq9o4Xs7W 1r33Nu2tKSCVvNWZ1R2/DtRTxmHI5ib0R0Vs3ie7rdGpG75xY61 gwwMUCFeF7VoFZT1T07AXA7yT1Ab0iE60iYHxh12VTASNPo8SegG 1vZqjrxrzIUbi1DloV620C6RhV7D741iXykhZmmxBH/UWvaK3G1I 1xE/Cva39hIE01Uw8m1SPNi1gLqLKw== </password></pre> Note: The password must be encoded in UTF-16LE encoding before it is encrypted. After encryption, the byte stream must be in Little-Endian order before conversion into Base64 encoding.

- **Return Value**

Returns resultCode.

- **Remarks**

Before using this operation, obtain the provisioning certificate of the user through the **getProvisioningCert** operation.

createWallet

```
public ResultMessage createWallet(String sessionId, String enterpriseId,
String walletId);
```

- **Synopsis**

To create a Wallet with the given enterprise ID.

This operation requests the IMS Server to create a Wallet for the provisioned user. A Wallet is only created for the user if the user does not have one.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID or the IBM Security Access Manager for Enterprise Single Sign-On user name of the user to be provisioned. In a deployment with multiple Active Directory (AD) domains, the format of the user name must be of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
walletId	The ID of the Wallet to be created. If this parameter is null or empty, a Wallet ID is generated.

- **Return Value**

Returns ResultMessage that consists of resultCode and resultString.

Returns Wallet ID in resultString if resultCode is OK.

- **Remarks**

Use this operation to create a Wallet after a user has been provisioned.

deleteImsAccount

```
public int deleteImsAccount(String sessionId, String enterpriseId);
```

- **Synopsis**

To delete an IBM Security Access Manager for Enterprise Single Sign-On user. This operation requests the IMS Server to delete an IBM Security Access Manager for Enterprise Single Sign-On user and the corresponding Wallet. If deleted, a user is removed from the IMS Server database. Another user of the same enterprise ID can be provisioned later.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.

- **Return Value**

Returns resultCode.

getProvisioningCert

```
public ResultMessage getProvisioningCert(String sessionId,
String enterpriseId);
```

- **Synopsis**

To get the specified X.509 provisioning certificate of the user.

This operation requests the IMS Server to return the X.509 certificate that can be used to encrypt passwords for provisioned account credentials.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.

- **Return Value**

Returns ResultMessage that consists of resultCode and resultString.

Returns the provisioning encryption certificate of the user in resultString if resultCode is OK. The certificate is a Base64-encoded X.509 (.CER) certificate.

- **Remarks**

Use this operation to obtain a provisioning certificate of the user before operations for setting account credentials are used: **addAccountCredential** and **setAccountCredential**.

getRegistrationStatus

```
public int getRegistrationStatus(String sessionId, String enterpriseId);
```

- **Synopsis**

To get the registration status of a user account on the IMS Server.

This operation requests the IMS Server to return the registration status of a user account that might not be found or might be revoked.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.

- **Return Value**

Returns resultCode.

getUserAccounts

```
public ResultArrayMap getUserAccounts(String sessionId, String enterpriseId);
```

- **Synopsis**

To get the list of accounts in the credential Wallet of a user.

This operation requests the IMS Server to return the list of accounts in the credential Wallet of a user. Password accounts and accounts that are provisioned are returned.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.

- **Return Value**

Returns ResultArrayMaps that consists of resultCode and an array of Maps. If the user does not have any accounts, the array is empty. Otherwise, each entry in the array is an instance of Map object. The Map object contains selected information for one account and the following keys are available:

- **USER_ID**

User name of the account.

- **AUTHSVC_ID**

The authentication service ID of the account.

preProvisionImsUser

```
public int preProvisionImsUser(String sessionId, String enterpriseId, String initialPassword, NameValue[] attributes);
```

- **Synopsis**

To create an IBM Security Access Manager for Enterprise Single Sign-On user.

This operation requests the IMS Server to create an IBM Security Access Manager for Enterprise Single Sign-On user with the specified enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) and initial password.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.

Argument name	Description
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
initialPassword	The initial password for the user.
attributes	Any IMS Server attributes (name value pairs) to be added for the new IBM Security Access Manager for Enterprise Single Sign-On user. null is an acceptable input. The following user attributes are used for a deployment with multiple AD domains: • userPrincipalName (optional): UPN of the user account and is only useful when enterprise directory is AD. • domainDnsName: DNS domain name of the enterprise directory. • samAccountName: name of the Active Directory user account and is only useful when enterprise directory is AD. • directoryId (optional): ID of enterprise directory that holds the user account (optional if there is only one enterprise directory configured in the IMS Server).

- **Return Value**

Returns resultCode.

removeAccountCredential

```
public int removeAccountCredential(String sessionId, String enterpriseId,
String authId, String accountType, String username);
```

- **Synopsis**

To remove provisioned account credentials for a user.

This operation requests the IMS Server to remove the specified account credentials for a user. The account must exist for the user.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
authId	The authentication service ID of the user account to be provisioned.
accountType	The type of account to be provisioned: Password, OTP, MAC, or Certificate.
username	The application user name of the user account to be removed.

- **Return Value**

Returns resultCode.

revokeImsAccount

```
public int revokeImsAccount(String sessionId, String enterpriseId);
```

- **Synopsis**

To revoke an IBM Security Access Manager for Enterprise Single Sign-On user.

This operation requests the IMS Server to revoke an IBM Security Access Manager for Enterprise Single Sign-On user and the authentication factors. If

revoked, a user is marked as revoked in the IMS Server database. New users are not allowed to reuse the enterprise ID of the revoked user.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.

- **Return Value**

Returns resultCode.

setAccountCredential

```
public int setAccountCredential(String sessionId, String enterpriseId,
String authId, String accountType, String username, String password);
```

- **Synopsis**

To set provisioned account credentials for a user.

This operation requests the IMS Server to set the application password of an existing user account. This operation fails if the account data does not exist or the account is not an account of Password type.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
authId	The authentication service ID of the user account to be set.
accountType	The type of account to be set: Password, OTP, MAC, or Certificate.
username	The application user name of the user account to be set.
password	The password in Base64, encrypted by the provisioning encryption certificate of the user in an XML snippet of the following form: <pre><password transformation="RSA/None/PKCS1Padding/2048/ ProvisionKeypair"> Qt0pkL0eWD+7vMPLUIVYHJFHijY+ 2ggvm1C26raOXMZBURrbQRbgvXeI4SA5t uh7EuBkJWjC/fhivpBqmqz2Nm1ersSUFZ4IQxYe0EXDtXBmkSF149I/ eieUVz hVcyvrzkP276FPX4Y01Miz/S4fq9o4Xs7W1r33Nu2tKSCVwvNWZ1R2/ DtRTxm HI5ib0R0Vs3ie7rdGpG75xY61gwwMUCFeF7VoFZT1T07 AXA7yT1Ab0iE60iYH xh12VTASNPo8Seg1vZqjrxrzIUbi1D1oV620 C6RhV7D741iXykhZmmxBH/UW vaK3G1I1xE/Cva39hIE01Uw8m1SPNi1gLqLKw== </password></pre> Note: The password must be encoded in UTF-16LE encoding before it is encrypted. After encryption, the byte stream must be in Little-Endian order before conversion into Base64 encoding.

- **Return Value**

Returns resultCode.

- **Remarks**

Before using this operation, obtain the provisioning certificate of the user through the **getProvisioningCert** operation.

getAllActiveUserIDs

```
public ResultStringList getAllActiveUserIDs(String sessionId,
String filter)
```

- **Synopsis**

To get a list of all active user enterprise IDs.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
filter	If set to asterisk (*), this argument returns all users. An asterisk (*) can also be used to represent any number and set of characters. For example, setting the filter parameter to a*h, returns all names beginning with a; and ending with h, namely, Ashish, Amitabh, Ajitabh, and so on.

- **Return Value**

A list of active user enterprise IDs.

- **Remarks**

Before using this operation, obtain the provisioning certificate of the user through the **getProvisioningCert** operation.

resetPassword

```
public int resetPassword(String sessionId, String enterpriseId,
String newPassword)
```

- **Synopsis**

To reset the password of a given IMS Server user ID.

Note: Provisioning API resets only the SAM E-SSO Password but not the Active Directory password.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
newPassword	New password for the given IMS Server user ID.

- **Return Value**

Returns resultCode.

- **Remarks**

Before using this operation, obtain the provisioning certificate of the user through the **getProvisioningCert** operation.

addAccountCredentialWithAccountFields

```
public int addAccountCredentialWithAccountFields(String sessionId,
        String enterpriseId, String authId, String accountType,
        NameValue[] accountFields)
```

- **Synopsis**

To add provisioned account credentials with multiple fields.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
authId	The authentication service ID of the user account to be provisioned.
accountType	The type of account to be provisioned: Password, OTP, MAC, or Certificate.
accountFields	The account data item template ID. The account data item template defines the properties of individual account data items. For example: aditi_cipwd. For more information about account data item templates, see the <i>AccessStudio Guide</i> .

- **Return Value**

Returns resultCode.

- **DATASTORE_EXCEPTION**

If there are database errors during loading of the requested authentication service.

- **UNKNOWN_AUTH_SERVICE**

If the requested authentication service does not exist.

- **INVALID_AUTH_SERVICE**

If the requested authentication service in IMS DB cannot be parsed.

- **INVALID_ACCOUNT_DATA_TEMPLATE**

If the account data template specified for the authId does not match the fields given as inputs.

- **BAD_INPUT_PARAMETERS**

If an input is null or empty.

- **USER_NOT_REGISTERED**

If the user is not yet registered.

- **UNSUPPORTED_AUTH_MECHANISM**

If authentication service does not support password authentication.

- **UNKNOWN_DATA_STORAGE_TEMPLATE**

If there is an error obtaining the storage template for an authentication service.

- **UNKNOWN_ACCOUNT_DATA_TEMPLATE**

If there is an error obtaining the account data template for an authentication service.

- **ACCOUNT_ALREADY_EXISTS**

If the account being added exists for a user.

- **NOT_OK**

If some error occurs when setting the account data.

addAccountCredentialWithSecondKey

```
public int addAccountCredentialWithSecondKey(String sessionId,
      String enterpriseId, String authId, String accountType,
      String username, String password, String secondKey)
```

- **Synopsis**

To update an account credential with three fields: username, password, and domain. The account must exist.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
adminEntId	The Enterprise ID of the admin who created the user.
authID	The authentication service ID of the user account to be provisioned.
accountType	The type of account to be provisioned: Password, OTP, MAC, or Certificate.
username	The user name of the account to be added.
domain	The domain of the account to be added.

- **Return Value**

Returns resultCode.

- **DATASTORE_EXCEPTION**

If there are database errors during loading of the requested authentication service.

- **UNKNOWN_AUTH_SERVICE**

If the requested authentication service does not exist.

- **INVALID_AUTH_SERVICE**

If the requested authentication service in IMS DB cannot be parsed.

- **INVALID_ACCOUNT_DATA_TEMPLATE**

If the account data template specified for the authId does not match the fields given as inputs.

- **BAD_INPUT_PARAMETERS**

If an input is null or empty.

- **USER_NOT_REGISTERED**

If the user is not yet registered.

- **UNSUPPORTED_AUTH_MECHANISM**

If authentication service does not support password authentication.

- **UNKNOWN_DATA_STORAGE_TEMPLATE**

If there is an error obtaining the storage template for an authentication service.

- **UNKNOWN_ACCOUNT_DATA_TEMPLATE**

If there is an error obtaining the account data template for an authentication service.

- **ACCOUNT_ALREADY_EXISTS**
If the account being added exists for a user.
 - **NOT_OK**
If some error occurs when setting the account data.
- the account data.

removeAccountCredentialWithAccountFields

```
public int removeAccountCredentialWithAccountFields(String sessionId,
    String enterpriseId, String authId, String accountType,
    NameValue[] accountFields)
```

- **Synopsis**
To remove provisioned account credentials with multiple fields.
- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
authID	The authentication service ID of the user account to be provisioned.
accountType	The type of account to be provisioned: Password, OTP, MAC, or Certificate.
accountFields	The account data item template ID. The account data item template defines the properties of individual account data items. For example: aditi_cipwd. For more information about account data item templates, see the <i>AccessStudio Guide</i> .

- **Return Value**
Returns resultCode.
- **DATASTORE_EXCEPTION**
If there are database errors during loading of the requested authentication service.
- **UNKNOWN_AUTH_SERVICE**
If the requested authentication service does not exist.
- **INVALID_AUTH_SERVICE**
If the requested authentication service in IMS DB cannot be parsed.
- **INVALID_ACCOUNT_DATA_TEMPLATE**
If the account data template specified for the authId does not match the fields given as inputs.
- **BAD_INPUT_PARAMETERS**
If an input is null or empty.
- **USER_NOT_REGISTERED**
If the user is not yet registered.
- **UNSUPPORTED_AUTH_MECHANISM**
If authentication service does not support password authentication.
- **UNKNOWN_DATA_STORAGE_TEMPLATE**
If there is an error obtaining the storage template for an authentication service.

- **UNKNOWN_ACCOUNT_DATA_TEMPLATE**
If there is an error obtaining the account data template for an authentication service.
- **ACCOUNT_ALREADY_EXISTS**
If the account being added exists for a user.
- **NOT_OK**
If some error occurs when setting the account data.

removeAccountCredentialWithSecondKey

```
public int removeAccountCredentialWithSecondKey(String sessionId,
        String enterpriseId, String authId, String accountType,
        String username, String secondKey)
```

- **Synopsis**

To remove an account credential with three fields: username, password, and domain. The account must exist.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
authID	The authentication service ID of the user account to be provisioned.
accountType	The type of account to be provisioned: Password, OTP, MAC, or Certificate.
username	The user name of the account to be added.
domain	The domain of the account to be added.

- **Return Value**

Returns resultCode.

- **DATASTORE_EXCEPTION**
If there are database errors during loading of the requested authentication service.
- **UNKNOWN_AUTH_SERVICE**
If the requested authentication service does not exist.
- **INVALID_AUTH_SERVICE**
If the requested authentication service in IMS DB cannot be parsed.
- **INVALID_ACCOUNT_DATA_TEMPLATE**
If the account data template specified for the authId does not match the fields given as inputs.
- **BAD_INPUT_PARAMETERS**
If an input is null or empty.
- **USER_NOT_REGISTERED**
If the user is not yet registered.
- **UNSUPPORTED_AUTH_MECHANISM**
If authentication service does not support password authentication.
- **UNKNOWN_DATA_STORAGE_TEMPLATE**

If there is an error obtaining the storage template for an authentication service.

- **UNKNOWN_ACCOUNT_DATA_TEMPLATE**

If there is an error obtaining the account data template for an authentication service.

- **ACCOUNT_ALREADY_EXISTS**

If the account being added exists for a user.

- **NOT_OK**

If some error occurs when setting the account data.

setAccountCredentialWithAccountFields

```
public int setAccountCredentialWithAccountFields(String sessionId,
        String enterpriseId, String authId, String accountType,
        NameValue[] accountFields)
```

- **Synopsis**

To set provisioned account credentials with multiple fields.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
authID	The authentication service ID of the user account to be provisioned.
accountType	The type of account to be provisioned: Password, OTP, MAC, or Certificate.
accountFields	The account data item template ID. The account data item template defines the properties of individual account data items. For example: aditi_cipwd. For more information about account data item templates, see the <i>AccessStudio Guide</i> .

- **Return Value**

Returns resultCode.

- **DATASTORE_EXCEPTION**

If there are database errors during loading of the requested authentication service.

- **UNKNOWN_AUTH_SERVICE**

If the requested authentication service does not exist.

- **INVALID_AUTH_SERVICE**

If the requested authentication service in IMS DB cannot be parsed.

- **INVALID_ACCOUNT_DATA_TEMPLATE**

If the account data template specified for the authId does not match the fields given as inputs.

- **BAD_INPUT_PARAMETERS**

If an input is null or empty.

- **USER_NOT_REGISTERED**

If the user is not yet registered.

- **UNSUPPORTED_AUTH_MECHANISM**

- If authentication service does not support password authentication.
- **UNKNOWN_DATA_STORAGE_TEMPLATE**
If there is an error obtaining the storage template for an authentication service.
- **UNKNOWN_ACCOUNT_DATA_TEMPLATE**
If there is an error obtaining the account data template for an authentication service.
- **ACCOUNT_ALREADY_EXISTS**
If the account being added exists for a user.
- **NOT_OK**
If some error occurs when setting the account data.

setAccountCredentialWithSecondKey

```
public int setAccountCredentialWithSecondKey(String sessionId,
      String enterpriseId, String authId, String accountType,
      String username, String password, String secondKey)
```

- **Synopsis**

To update an account credential with three fields: username, password, and domain. The account must exist.

- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple Active Directory (AD) domains, the format of the user name must be of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
authID	The authentication service ID of the user account to be provisioned.
accountType	The type of account to be provisioned: Password, OTP, MAC, or Certificate.
username	The user name of the account to be added.
domain	The domain of the account to be added.

- **Return Value**

Returns resultCode.

- **DATASTORE_EXCEPTION**
If there are database errors during loading of the requested authentication service.
- **UNKNOWN_AUTH_SERVICE**
If the requested authentication service does not exist.
- **INVALID_AUTH_SERVICE**
If the requested authentication service in IMS DB cannot be parsed.
- **INVALID_ACCOUNT_DATA_TEMPLATE**
If the account data template specified for the authId does not match the fields given as inputs.
- **BAD_INPUT_PARAMETERS**
If an input is null or empty.
- **USER_NOT_REGISTERED**

- If the user is not yet registered.
- **UNSUPPORTED_AUTH_MECHANISM**
If authentication service does not support password authentication.
- **UNKNOWN_DATA_STORAGE_TEMPLATE**
If there is an error obtaining the storage template for an authentication service.
- **UNKNOWN_ACCOUNT_DATA_TEMPLATE**
If there is an error obtaining the account data template for an authentication service.
- **ACCOUNT_ALREADY_EXISTS**
If the account being added exists for a user.
- **NOT_OK**
If some error occurs when setting the account data.

unlockAccount

```
public int unlockAccount (String sessionId, String enterpriseId,
                        String adminEntId, String unlockEntDirAccount)
```

- **Synopsis**
To unlock the user with a given IMS Server user ID.
- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple Active Directory (AD) domains, the format of the user name must be of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
adminEntId	The Enterprise ID of the admin who created the user.

- **Return Value**
Returns resultCode.
- **ResultCode.OK**
If the password is reset successfully.
- **ResultCode.UNLOCK_WALLET**
The account is unlocked successfully, but unlocking enterprise directory account fails.
- **ResultCode.UNLOCK_ENTERPRISE**
Unlocking enterprise directory account successful, but unlocking account fails.

lockAccount

```
public int lockAccount (String sessionId, String enterpriseId, String adminEntId)
```

- **Synopsis**
To lock the user with a given IMS Server user ID.
- **Arguments**

Argument name	Description
sessionId	The session ID of the provisioning agent.

Argument name	Description
enterpriseId	The enterprise ID (IBM Security Access Manager for Enterprise Single Sign-On user name) of the user to be provisioned. In a deployment with multiple AD domains, the format of the user name is of the form dnsDomain\sAMAccountName. For example, ibm.com\user name.
adminEntId	The Enterprise ID of the admin who created the user.

- **Return Value**

Returns resultCode.

- **resultCode.OK**

If the password is reset successfully.

- **resultCode.UNLOCK_WALLET**

The account is unlocked successfully, but unlocking enterprise directory account fails.

- **resultCode.UNLOCK_ENTERPRISE**

Unlocking enterprise directory account successful, but unlocking account fails.

Using Provisioning API for account setup and maintenance

Use these topics to perform basic tasks by using Provisioning API, such as adding application accounts, resetting application passwords, and de-provisioning users.

See these topics for more information.

- “Provisioning a new user by using Provisioning API”
- “Adding an application account by using Provisioning API” on page 51
- “Resetting an application password by using Provisioning API” on page 51
- “Deleting an application account by using Provisioning API” on page 51
- “De-provisioning users by using Provisioning API” on page 51

Provisioning a new user by using Provisioning API

Follow these steps to provision a new user.

Procedure

1. The provisioning system provides the IBM Security Access Manager for Enterprise Single Sign-On user name and application credentials to the IMS Server. The IMS Server then provisions the IBM Security Access Manager for Enterprise Single Sign-On user.
2. The Wallet of the user is also initialized on the IMS Server, so that application account credentials can be added to the Wallet.

Note: Create IBM Security Access Manager for Enterprise Single Sign-On user accounts before other application accounts. Application account credentials cannot be added to the Wallet of the user before the user is provisioned on the IMS Server.

3. The users log on with their SAM E-SSO user names and initial passwords when they use AccessAgent for the first time.

Results

After they are prompted to change their initial password, the Wallet containing the provisioned account credentials is downloaded from the IMS Server.

Adding an application account by using Provisioning API

Follow these steps to add an application account.

Procedure

1. The provisioning system provides the IBM Security Access Manager for Enterprise Single Sign-On user name and application credentials to the IMS Server. The IMS Server then adds the application and other credentials to the Wallet of the user.
2. The next time the user logs on to AccessAgent, the AccessAgent will have the necessary credentials in the Wallet to automate sign-on to the new application. Applications can then be added without informing users of the new credentials. Users log on to AccessAgent.

Resetting an application password by using Provisioning API

Follow these steps to reset an application password.

Procedure

1. The provisioning system provides the IBM Security Access Manager for Enterprise Single Sign-On user name and the new application password to the IMS Server. The IMS Server then updates the application password in the Wallet of the user.
2. The next time the user logs on to AccessAgent, the AccessAgent will have the updated application passwords in the Wallet to automate sign-on to the applications. Administrators can reset application passwords without directly notifying each user. Users sign on to IBM Security Access Manager for Enterprise Single Sign-On to log on to the applications.

Deleting an application account by using Provisioning API

Follow these steps to delete an application account.

Procedure

1. The provisioning system provides the IBM Security Access Manager for Enterprise Single Sign-On user name and the application account name to the IMS Server. The IMS Server then deletes the application account from the Wallet of the user.
2. The next time the user logs on to AccessAgent, it no longer has access to the deleted application credentials in the Wallets and not be able to log on to the application. Applications can be removed centrally and all access can be terminated automatically.

De-provisioning users by using Provisioning API

Follow these steps to de-provision users.

Procedure

1. The provisioning system provides the IBM Security Access Manager for Enterprise Single Sign-On user name to be de-provisioned to the IMS Server. The IMS Server then revokes the user. This revocation removes the accounts and the Wallet of the IBM Security Access Manager for Enterprise Single Sign-On user from the server.
2. If the user attempts to log on using AccessAgent, the logon fails, and Wallets that have been cached locally by AccessAgent are revoked.

Chapter 3. Provisioning Agent

The Provisioning Agent is an application that monitors Active Directory periodically for deletion of users before attempting to delete or revoke the corresponding IMS Server users on the IMS Server.

The Provisioning Agent is installed as a separate entity. Provisioning Agent uses the IMS Server provisioning APIs to perform de-provisioning tasks when a third party user provisioning system is not used. You can configure the Provisioning Agent to monitor Active Directory or a host with ADAM services to determine user account status and various user attributes.

An enterprise with Active Directory typically uses the Active Directory management console to manage its users. User management activities include setting of user attributes, disabling accounts and de-provisioning accounts.

After deploying IBM Security Access Manager for Enterprise Single Sign-On, the enterprise must manage users with the AccessAdmin application. IMS Server manages the Wallets containing all application credentials, audit logs, and user policies.

When a user must be de-provisioned, the Administrator must deprovision the user from both the IMS Server and the Active Directory.

To reduce administrative effort, the Administrator can use the IMS Server as the central administration server.

When an IMS Server user is deprovisioned through AccessAdmin, the Active Directory account of the user can be deleted from the Active Directory by using a connector.

However, some enterprises might not want to change their existing business and Help desk processes of de-provisioning users through the Active Directory management console. In such cases, the Provisioning Agent is used.

With the Provisioning Agent, the Administrator or Help desk user can deprovision users from the Active Directory management console. The Provisioning Agent then automatically de-provisions the corresponding IMS Server users from the IMS Server.

Note:

- There are no policy settings for the Provisioning Agent.
- The Provisioning Agent can support up to a maximum number of 2000 users. Performance tests must be done before deploying Provisioning Agent for more than 2000 users.
- The Provisioning Agent can deprovision IMS Server users only when the users have been deprovisioned in the Active Directory.

See the following topics for more information.

- “Provisioning Agent features” on page 54
- “Prerequisites for Provisioning Agent” on page 54
- “How provisioning agent works” on page 54

- “Installing and configuring the Provisioning Agent” on page 55

Provisioning Agent features

This topic describes the high-level specifications of the IBM Security Access Manager for Enterprise Single Sign-On provisioning Active Directory agent, as well as the various deployment options.

No administrative costs for Active Directory-based user de-provisioning

The solution does not add administrative costs for user de-provisioning through the Active Directory management console.

No modification to existing Active Directory and provisioning infrastructure

The solution does not require modification to existing Active Directory and provisioning infrastructure. This feature eliminates the need to obtain approvals for infrastructure modifications, which are typically tedious and might take too long.

Complete de-provisioning of accounts

When users are de-provisioned on Active Directory, the solution performs a complete de-provisioning of the corresponding IMS Server users. Complete de-provisioning includes revoking the authentication factors of users, disabling user Wallets and creation of audit logs. It ensures that de-provisioning complies with relevant legislations, such as SOX.

IMS Server users can be either revoked or deleted.

Note: To retain audit log information for compliance initiatives, when de-provisioning, you can revoke an account instead of deleting it. IBM Security Access Manager for Enterprise Single Sign-On accounts that you revoke cannot be reactivated.

If a customer has an ADAM server, install the Provisioning Agent on the same machine as the ADAM. Search and directory lookup operations can complete faster because the ADAM host has its own cached copy of the user directory. However, the Provisioning Agent can also be configured to communicate directly with Active Directory.

An enterprise might have one or more ADAMs. If there are multiple ADAMs supporting multiple domains, each ADAM machine hosts one Provisioning Agent.

Prerequisites for Provisioning Agent

In the Provisioning Agent solution:

- the Active Directory can be the enterprise directory
- the Active Directory management console is used for user management; to set user attributes, disable accounts, and deprovision accounts

How provisioning agent works

Provisioning Agent with IBM Security Access Manager for Enterprise Single Sign-On supports only de-provisioning IMS Server user when the Active Directory account is de-provisioned.

When the administrator or help desk employee de-provisions a user in the Active Directory management console, the following process occurs:

1. In the Active Directory management console, the user is deprovisioned.

2. The Provisioning Agent detects (through periodic polling) that a user has been de-provisioned on Active Directory.
3. The Provisioning Agent invokes the provisioning API of the IMS Server to deprovision the IMS Server user.
4. The authentication factors of the user are automatically revoked.
5. When the user attempts to log on with AccessAgent, the user is informed that the account has been revoked.

Installing and configuring the Provisioning Agent

The Provisioning Agent monitors Active Directory server periodically for deletion or revocation of users and triggers the corresponding action on the IMS Server. The IBM Security Access Manager for Enterprise Single Sign-On Provisioning Agent is distributed as a compressed (.ZIP) archive.

Use the Provisioning Agent when a user provisioning system like Tivoli Identity Manager is not deployed. The Provisioning Agent saves administrators the effort of separately revoking an IBM Security Access Manager for Enterprise Single Sign-On user account when the administrator deletes the user from Active Directory.

There are two different versions of the distributed Provisioning Agent archive. The difference between the versions is whether the Java Runtime Environment (JRE) is included in the distribution.

The archives and their sizes are:

- SAMESSOProvAgent.zip (34.0 MB)
- SAMESSOProvAgent_NoJRE.zip (5.29 MB)

The directory structure of the distributable archive is as follows:

- **bin**: The binary files used for starting, stopping, installing, or uninstalling the Provisioning Agent as a Windows service.
- **config**: The Provisioning Agent configuration files.
- **docs**: The Provisioning Agent documentation.
- **j2re1.5.0**: The JRE, if it is included in the archive.
- **lib**: The libraries that are needed by the Provisioning Agent.
- **logs**: The location of the log files.

Installing the Provisioning Agent

Use the IMS Configuration Utility to set up a new IMS Bridge and install the Provisioning Agent.

Before you begin

The IMS Server is installed and configured.

The Provisioning Agent requires JRE version 1.5.0 or later. If the correct version is not provided in the distributable archive, you can download the JRE from <http://www.java.com>.

After installing the JRE, the system property **JAVA_HOME** must be set to the base directory of the JRE installation (for example, C:\j2re1.5.0).

Procedure

1. With the IMS Configuration Utility, set up a new IMS Bridge on the IMS Server. The Provisioning Agent connects to Active Directory with the configured IMS provisioning bridge.
 - a. Select **IMS Bridges > Configure**.
 - b. Specify the IP address (IMS Bridge IP addresses) of the machine where the Provisioning Agent is installed.
 - c. Create a user name (Name) and password (IMS Bridge password) for the IMS Bridge. These user names are used later in the Provisioning Agent configuration.
2. Extract the distributable archive to a directory. For example, C:\IBM.
3. Maintain the directory structure in the archive.
4. The Provisioning Agent uses one-way SSL to communicate with the IMS Server. This configuration means that the SSL certificate of the IMS Server must be trusted by importing it into a truststore.

The truststore can either be an existing one used by other applications, or the truststore provided in config\truststore.jks.

To import the Base-64 certificate to a truststore, use the Java **keytool.exe** command-line tool.

Example

See the example usage:

```
j2re1.5.0\bin\keytool.exe -import -alias ImsSsl -file C:\IBM\sslCert.cer  
-keystore C:\IBM\config\truststore.jks
```

Configuring the Provisioning Agent

Configure the Provisioning Agent by editing the provisioningAgent.properties file located in the config directory. For example, if the archive is extracted to C:\IBM, the configuration file is at C:\IBM\config\provisioningAgent.properties.

What to do next

See “Starting the Provisioning Agent” on page 57.

Setting up the Provisioning Agent with Active Directory Application Mode (ADAM)

If a host with ADAM is used, configure Active Directory synchronization for the IBM Security Access Manager for Enterprise Single Sign-On Provisioning Agent to work.

- The **sAMAccountName** of the user is synchronized from Active Directory to ADAM.
- The X.500 DN of the user on ADAM (for example, CN=john,OU=us,DC=ibm,DC=com,DC=adam) might not be the same as the X.500 DN on Active Directory (for example, CN=john,OU=us,DC=ibm,DC=com).
- ADAM and MIIS or ADAM synchronization must be configured correctly to ensure that their DNs are the same.

Starting the Provisioning Agent

Use command lines to test the Provisioning Agent configuration, install the Provisioning Agent as a Windows service, and manually start the Provisioning Agent service.

Before you begin

Before starting the Provisioning Agent, test the configurations by launching the Provisioning Agent from the command prompt with the console option as follows:

```
C:\IBM\bin\enPrvAgt.bat console
```

About this task

The Provisioning Agent queries Active Directory or ADAM. Verify the connection based on the displayed log messages. If there are errors, the configuration values must be corrected. Press **Ctrl+C** to stop the Provisioning Agent and check the configuration settings before trying again.

Procedure

1. To install the Provisioning Agent as a Windows service, launch the Provisioning Agent from the command prompt with the install option as follows:

```
C:\IBM\bin\enPrvAgt.bat install
```

2. The Provisioning Agent automatically starts whenever the computer starts.

You can also manually start the Provisioning Agent service with the following command:

```
C:\IBM\bin\enPrvAgt.bat start
```

What to do next

See “Provisioning Agent configuration parameters.”

Provisioning Agent configuration parameters

Set up the Provisioning Agent by editing the `provisioningAgent.properties` file located in the `config` directory. For example, if the archive is extracted to `C:\IBM`, the configuration file is at `C:\IBM\config\provisioningAgent.properties`.

The Provisioning Agent properties file contains the following configuration parameters:

ldap.userid_attribute

The LDAP attributes are cached by the Provisioning Agent and are used to determine the user name on the IMS Server. This configuration can contain multiple values. It is optional and the default value is `sAMAccountName`.

```
ldap.userid_attribute = sAMAccountName
```

ldap.search_filter

The LDAP search filter that is used when searching for users. This configuration is optional and by default there is no search filter.

```
ldap.search_filter = (sAMAccountName=*)
```

revokeOrDeleteUser

Whether a user is revoked or deleted from the IMS Server when the user is

detected as removed from Active Directory. There are two possible values for this configuration key: revoke and delete. This configuration is optional and the default value is revoke.

revokeOrDeleteUser = revoke

maxRemovalAttempts

The number of attempts the Provisioning Agent makes to remove a user from IMS Server. After the maximum number of removal attempts, an error is logged in the Windows Event Log. No further action is taken on that user. This configuration value must be an integer. It is optional and the default value is 5.

maxRemovalAttempts = 5

userCacheFilePathAndName

The location of the file where the users from Active Directory are cached. A configuration value is optional and the default value is agentUserCacheV2.xml. This value is set in the default configuration that ships with Provisioning Agent.

userCacheFilePathAndName = ../config/agentUserCacheV2.xml

executionIntervalInMinutes

The interval that the Provisioning Agent polls Active Directory or ADAM for changes. When a change is detected, the agent de-provisions the user from the IMS Server. This configuration must be an integer. It is optional and the default value is 30.

executionIntervalInMinutes = 30

provisioningbridge.userName

The user name used to authenticate with the IMS Server. A configuration value is required.

provisioningBridge.userName=ImsBridge

provisioningbridge.password

The password used to authenticate with the IMS Server. A configuration value is required.

provisioningBridge.password=password

ldap.server_uri

The LDAP server URI. Location of the Active Directory server (for example, ldap://machinename). A configuration value is required.

ldap.server_uri = ldap://ActiveDirectoryServerNameOrIPAddress:389

Note: The agent can be configured for connecting through LDAPS, which is LDAP over SSL. Set the **ldap.server_uri** entry to ldaps://ActiveDirectoryName:636.

ldap.lookup_user_name

The Active Directory user name with permissions for lookup operations. If not set, the Active Directory must support anonymous connections.

ldap.lookup_user_name = LookupUserName

ldap.lookup_user_password

The corresponding password for the Active Directory user name with permissions for lookup operations.

ldap.lookup_user_password = LookupPassword

ldap.lookup_user_base_dn

The base distinguished name (DN) of the lookup Active Directory user. A configuration value is required.

ldap.lookup_user_base_dn = CN=Users,DC=company,DC=com

ldap.users_tree_dn

The distinguished names (DNs) of the users in Active Directory. A configuration value is required.

```
ldap.users_tree_dn = CN=Users,DC=company,DC=com ; CN=Users2,DC=company,DC=com
```

ldap.context_factory

The qualified class name of the factory class that creates an initial context. A configuration value is optional and the default value is `com.sun.jndi.ldap.LdapCtxFactory`.

```
ldap.context_factory = com.sun.jndi.ldap.LdapCtxFactory
```

ldap.security_protocol

The protocol used to connect to Active Directory. There are two possible values for this configuration: `ssl` and `none`. A configuration value is optional and the default is `none`.

```
ldap.security_protocol = none
```

ldap.authentication

The mechanism used to authenticate with Active Directory. There are two possible values for this configuration: `simple` and `none`. A user name and password are required for `simple` authentication, while an anonymous bind is done when `none` is configured. A configuration value is optional and the default value is `simple`.

```
ldap.authentication = simple
```

ldap.referral

Specifies how referrals returned by the LDAP server are processed. There are two possible values for this configuration: `follow` and `ignore`. If set to `follow`, any referrals are followed automatically. If set to `ignore`, any referrals are ignored. A configuration value is optional and the default value is `follow`.

```
ldap.referral = follow
```

ldap.search_scope

Specifies the scope for user search on the LDAP server. There are two possible values for this configuration: `one_level` and `sub_tree`. If set to `one_level`, a single level is searched for users. If set to `sub_tree`, the entire subtree is searched for users. A configuration value is optional and the default value is `one_level`.

```
ldap.search_scope = one_level
```

ldap.user_dn_attribute

The distinguished name (DN) attribute for the users. A configuration value is optional and the default is `cn`.

```
ldap.user_dn_attribute = cn
```

count_limit

The maximum number of user names retrieved from Active Directory when searching for users. The value is greater than the page size. A configuration value is optional and the default value is 200.

```
count_limit = 200
```

time_limit

The maximum time (in milliseconds) before a connection timeout occurs. A configuration value is optional and the default value is 3000.

```
time_limit = 3000
```

page_size

The user names retrieved from Active Directory and retrieved in groups (called pages) to obtain all the users without exceeding the maximum retrieval limit.

This value is the size of that page, which must be less than the maximum retrieval limit. A configuration value is optional and the default value is 100.

```
page_size = 100
```

Appendix A. WSDL for server authentication

The WSDL for the server authentication API can be obtained from an installed IMS Server.

Navigate to the following URL. Replace `imsserver` with the host name of your IMS Server. `https://imsserver/ims/services/encentuate.ims.service.ServerAuthentication?wsdl`.

The sample WSDL for server authentication is also included here for reference purposes.

```
<?xml version="1.0" encoding="UTF-8"?>

<wsdl:definitions targetNamespace="https://imsserver/ims/services/
encentuate.ims.service.ServerAuthentication"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="https://imsserver/ims/services/
encentuate.ims.service.ServerAuthentication"
xmlns:intf="https://imsserver/ims/services/
encentuate.ims.service.ServerAuthentication"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://result.ims.encentuate"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <wsdl:types>
    <schema targetNamespace="http://result.ims.encentuate"
      xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <complexType name="ResultMessage">
        <sequence>
          <element name="resultCode" type="xsd:int" />
          <element name="resultString" nillable="true" type="xsd:string" />
        </sequence>
      </complexType>
      <element name="ResultMessage" nillable="true" type="tns1:ResultMessage" />
    </schema>
  </wsdl:types>

  <wsdl:message name="loginByPasswordResponse">
    <wsdl:part name="loginByPasswordReturn" type="tns1:ResultMessage" />
  </wsdl:message>

  <wsdl:message name="terminateSessionResponse">
    <wsdl:part name="terminateSessionReturn" type="xsd:int" />
  </wsdl:message>

  <wsdl:message name="loginByPasswordRequest">
    <wsdl:part name="serverId" type="xsd:string" />
    <wsdl:part name="password" type="xsd:string" />
  </wsdl:message>

  <wsdl:message name="terminateSessionRequest">
    <wsdl:part name="sessionKey" type="xsd:string" />
  </wsdl:message>

  <wsdl:portType name="ServerAuthentication">
    <wsdl:operation name="terminateSession" parameterOrder="sessionKey">
```

```

<wsdl:input message="impl:terminateSessionRequest"
name="terminateSessionRequest"/>
<wsdl:output message="impl:terminateSessionResponse"
name="terminateSessionResponse"/>
</wsdl:operation>

<wsdl:operation name="loginByPassword" parameterOrder="serverId password">
<wsdl:input message="impl:loginByPasswordRequest"
name="loginByPasswordRequest"/>
<wsdl:output message="impl:loginByPasswordResponse"
name="loginByPasswordResponse"/>
</wsdl:operation>
</wsdl:portType>

<wsdl:binding name="encentuate.ims.service.ServerAuthenticationSoapBinding"
type="impl:ServerAuthentication">
<wsdl:soap:binding style="rpc" transport="http://schemas.xmlsoap.org/
soap/http"/>
<wsdl:operation name="terminateSession">
<wsdl:soap:operation soapAction="soapAction=""/>
<wsdl:input name="terminateSessionRequest">
<wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://imsserver/ims/services/encentuate.ims.service.
ServerAuthentication" use="encoded"/>
</wsdl:input>

<wsdl:output name="terminateSessionResponse">
<wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://imsserver/ims/services/encentuate.ims.service.
ServerAuthentication" use="encoded"/>
</wsdl:output>
</wsdl:operation>

<wsdl:operation name="loginByPassword">
<wsdl:soap:operation soapAction="soapAction=""/>
<wsdl:input name="loginByPasswordRequest">
<wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://imsserver/ims/services/encentuate.ims.service.
ServerAuthentication" use="encoded"/>
</wsdl:input>

<wsdl:output name="loginByPasswordResponse">
<wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://imsserver/ims/services/encentuate.ims.service.
ServerAuthentication" use="encoded"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>

<wsdl:service name="ServerAuthenticationService">
<wsdl:port binding="impl:encentuate.ims.service.
ServerAuthenticationSoapBinding" name="encentuate.ims.service.
ServerAuthentication">
<wsdl:soap:address location="https://imsserver/ims/services/
encentuate.ims.service.ServerAuthentication"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

Appendix B. WSDL for provisioning service

You can view the WSDL for provisioning service APIs.

The WSDL for the server authentication API can be obtained from an installed IMS Server at the following URL (replace `imsserver` with the host name of your IMS Server): `https://imsserver/ims/services/encentuate.ims.service.ProvisioningService?wsdl`.

The sample WSDL for provisioning services is also included here for reference purposes.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="https://localhost/ims/services/encentuate.ims.service.ProvisioningService"
  xmlns:intf="https://localhost/ims/services/encentuate.ims.service.ProvisioningService"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns1="http://result.ims.encentuate"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <schema targetNamespace="http://result.ims.encentuate"
      xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <complexType name="NameValue">
        <sequence>
          <element name="name" nillable="true" type="xsd:string"/>
          <element name="value" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <complexType name="ResultMessage">
        <sequence>
          <element name="resultCode" type="xsd:int"/>
          <element name="resultString" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <complexType name="ResultArrayMap">
        <sequence>
          <element name="resultCode" type="xsd:int"/>
          <element name="resultString" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <complexType name="ResultStringList">
        <sequence>
          <element name="errorCode" type="xsd:int"/>
          <element name="stringList" nillable="true" type="impl:ArrayOf_xsd_string"/>
        </sequence>
      </complexType>
      <complexType name="ResultStringList" nillable="true" type="tns1:ResultStringList"/>
    </schema>
  </wsdl:types>
  <wsdl:message name="setAccountCredentialWithAccountFieldsResponse">
    <wsdl:part name="setAccountCredentialWithAccountFieldsReturn" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="addAccountCredentialWithAccountFieldsResponse">
    <wsdl:part name="addAccountCredentialWithAccountFieldsReturn" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="revokeImsAccountWithEntIdResponse">
    <wsdl:part name="revokeImsAccountWithEntIdReturn" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="createWalletResponse">
    <wsdl:part name="createWalletReturn" type="tns1:ResultMessage"/>
  </wsdl:message>
  <wsdl:message name="getAllActiveUserIdsRequest">
    <wsdl:part name="sessionId" type="xsd:string"/>
    <wsdl:part name="filter" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="setAccountCredentialWithEntIdRequest">
    <wsdl:part name="sessionId" type="xsd:string"/>
    <wsdl:part name="enterpriseId" type="xsd:string"/>
    <wsdl:part name="adminEntId" type="xsd:string"/>
    <wsdl:part name="authId" type="xsd:string"/>
    <wsdl:part name="accountType" type="xsd:string"/>
    <wsdl:part name="username" type="xsd:string"/>
    <wsdl:part name="password" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="setAccountCredentialWithSecondKeyRequest">
    <wsdl:part name="sessionId" type="xsd:string"/>
    <wsdl:part name="enterpriseId" type="xsd:string"/>
    <wsdl:part name="authId" type="xsd:string"/>
  </wsdl:message>
</wsdl:definitions>
```

```

</wsdl:part>
<wsdl:part name="accountType" type="xsd:string">
</wsdl:part>
<wsdl:part name="username" type="xsd:string">
</wsdl:part>
<wsdl:part name="password" type="xsd:string">
</wsdl:part>
<wsdl:part name="secondKey" type="xsd:string">
</wsdl:part>

</wsdl:message>
<wsdl:message name="resetPasswordResponse">
  <wsdl:part name="resetPasswordReturn" type="xsd:int">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="revokeImsAccountRequest">
  <wsdl:part name="sessionId" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="enterpriseId" type="xsd:string">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="getProvisioningCertRequest">
  <wsdl:part name="sessionId" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="enterpriseId" type="xsd:string">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="deleteImsAccountRequest">
  <wsdl:part name="sessionId" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="enterpriseId" type="xsd:string">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="revokeImsAccountResponse">
  <wsdl:part name="revokeImsAccountReturn" type="xsd:int">
  </wsdl:part>
</wsdl:message>

<wsdl:message name="lockAccountResponse">
  <wsdl:part name="lockAccountReturn" type="xsd:int">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="getRegistrationStatusResponse">
  <wsdl:part name="getRegistrationStatusReturn" type="xsd:int">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="getAllActiveUserIDsResponse">
  <wsdl:part name="getAllActiveUserIDsReturn" type="tnsl:ResultStringList">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="revokeImsAccountWithEntIdRequest">
  <wsdl:part name="sessionId" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="enterpriseId" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="adminEntId" type="xsd:string">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="getRegistrationStatusRequest">
  <wsdl:part name="sessionId" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="enterpriseId" type="xsd:string">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="addAccountCredentialRequest">
  <wsdl:part name="sessionId" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="enterpriseId" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="authId" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="accountType" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="username" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="password" type="xsd:string">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="deleteImsAccountResponse">
  <wsdl:part name="deleteImsAccountReturn" type="xsd:int">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="preProvisionImsUserResponse">
  <wsdl:part name="preProvisionImsUserReturn" type="xsd:int">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="deleteImsAccountUsingAttrsRequest">
  <wsdl:part name="sessionId" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="attributes" type="impl:ArrayOf_tns1_NameValue">
  </wsdl:part>
</wsdl:message>

<wsdl:message name="getUserAccountsRequest">
  <wsdl:part name="sessionId" type="xsd:string">
  </wsdl:part>
  <wsdl:part name="enterpriseId" type="xsd:string">
  </wsdl:part>
</wsdl:message>

```

```

< wsdl:message name="removeAccountCredentialWithEntIdResponse">
  < wsdl:part name="removeAccountCredentialWithEntIdReturn" type="xsd:int">
    < /wsdl:part>
  < /wsdl:message>
< wsdl:message name="createWalletRequest">
  < wsdl:part name="sessionId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="walletId" type="xsd:string">
    < /wsdl:part>
< /wsdl:message>
< wsdl:message name="deleteImsAccountWithEntIdRequest">
  < wsdl:part name="sessionId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="adminEntId" type="xsd:string">
    < /wsdl:part>
< /wsdl:message>
< wsdl:message name="deleteImsAccountWithEntIdResponse">
  < wsdl:part name="deleteImsAccountWithEntIdReturn" type="xsd:int">
    < /wsdl:part>
< /wsdl:message>
< wsdl:message name="preProvisionImsUserWithEntIdRequest">
  < wsdl:part name="sessionId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="adminEntId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="initialPassword" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="attributes" type="impl:ArrayOf_tns1_NameValue">
    < /wsdl:part>
< /wsdl:message>
< wsdl:message name="addAccountCredentialWithAccountFieldsRequest">
  < wsdl:part name="sessionId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="authId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="accountType" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="accountFields" type="impl:ArrayOf_tns1_NameValue">
    < /wsdl:part>
< /wsdl:message>
< wsdl:message name="removeAccountCredentialWithAccountFieldsRequest">
  < wsdl:part name="sessionId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="authId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="accountType" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="accountFields" type="impl:ArrayOf_tns1_NameValue">
    < /wsdl:part>
< /wsdl:message>
< wsdl:message name="removeAccountCredentialRequest">
  < wsdl:part name="sessionId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="authId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="accountType" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="username" type="xsd:string">
    < /wsdl:part>
< /wsdl:message>
< wsdl:message name="lockAccountRequest">
  < wsdl:part name="sessionId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
    < /wsdl:part>
  < wsdl:part name="adminEntId" type="xsd:string">
    < /wsdl:part>
< /wsdl:message>
< wsdl:message name="unlockAccountResponse">
  < wsdl:part name="unlockAccountReturn" type="xsd:int">
    < /wsdl:part>
< /wsdl:message>
< wsdl:message name="deleteImsAccountUsingAttrsResponse">
  < wsdl:part name="deleteImsAccountUsingAttrsReturn" type="xsd:int">
    < /wsdl:part>
< /wsdl:message>
< wsdl:message name="getUserAccountsResponse">
  < wsdl:part name="getUserAccountsReturn" type="tns1:ResultArrayMap">
    < /wsdl:part>
< /wsdl:message>
< wsdl:message name="removeAccountCredentialWithSecondKeyResponse">
  < wsdl:part name="removeAccountCredentialWithSecondKeyReturn" type="xsd:int">
    < /wsdl:part>

```

```

< /wsdl:message>
< wsdl:message name="resetPasswordRequest">
  < wsdl:part name="sessionId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="newPassword" type="xsd:string">
  < /wsdl:part>
< /wsdl:message>
< wsdl:message name="addAccountCredentialResponse">
  < wsdl:part name="addAccountCredentialReturn" type="xsd:int">
  < /wsdl:part>
< /wsdl:message>
< wsdl:message name="removeAccountCredentialWithAccountFieldsResponse">
  < wsdl:part name="removeAccountCredentialWithAccountFieldsReturn" type="xsd:int">
  < /wsdl:part>
< /wsdl:message>
< wsdl:message name="removeAccountCredentialResponse">
  < wsdl:part name="removeAccountCredentialReturn" type="xsd:int">
  < /wsdl:part>
< /wsdl:message>
< wsdl:message name="setAccountCredentialWithEntIdResponse">
  < wsdl:part name="setAccountCredentialWithEntIdReturn" type="xsd:int">
  < /wsdl:part>
< /wsdl:message>
< wsdl:message name="addAccountCredentialWithEntIdResponse">
  < wsdl:part name="addAccountCredentialWithEntIdReturn" type="xsd:int">
  < /wsdl:part>
< /wsdl:message>
< wsdl:message name="preProvisionImsUserWithEntIdResponse">
  < wsdl:part name="preProvisionImsUserWithEntIdReturn" type="xsd:int">
  < /wsdl:part>
< /wsdl:message>
< wsdl:message name="addAccountCredentialWithEntIdRequest">
  < wsdl:part name="sessionId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="adminEntId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="authId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="accountType" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="username" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="password" type="xsd:string">
  < /wsdl:part>
< /wsdl:message>
< wsdl:message name="setAccountCredentialWithAccountFieldsRequest">
  < wsdl:part name="sessionId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="authId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="accountType" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="accountFields" type="impl:ArrayOf_tns1_NameValue">
  < /wsdl:part>
< /wsdl:message>
< wsdl:message name="removeAccountCredentialWithEntIdRequest">
  < wsdl:part name="sessionId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="adminEntId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="authId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="accountType" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="username" type="xsd:string">
  < /wsdl:part>
< /wsdl:message>
< wsdl:message name="setAccountCredentialRequest">
  < wsdl:part name="sessionId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="authId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="accountType" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="username" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="password" type="xsd:string">
  < /wsdl:part>
< /wsdl:message>
< wsdl:message name="addAccountCredentialWithSecondKeyRequest">
  < wsdl:part name="sessionId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="enterpriseId" type="xsd:string">
  < /wsdl:part>
  < wsdl:part name="authId" type="xsd:string">
  < /wsdl:part>

```

```

    < wsdl:part name="accountType" type="xsd:string">

    < /wsdl:part>
    < wsdl:part name="username" type="xsd:string">
    < /wsdl:part>
    < wsdl:part name="password" type="xsd:string">
    < /wsdl:part>
    < wsdl:part name="secondKey" type="xsd:string">
    < /wsdl:part>
  < /wsdl:message>
  < wsdl:message name="revokeImsAccountUsingAttrsRequest">

    < wsdl:part name="sessionId" type="xsd:string">
    < /wsdl:part>
    < wsdl:part name="attributes" type="impl:ArrayOf_tns1_NameValue">
    < /wsdl:part>
  < /wsdl:message>
  < wsdl:message name="removeAccountCredentialWithSecondKeyRequest">
    < wsdl:part name="sessionId" type="xsd:string">
    < /wsdl:part>
    < wsdl:part name="enterpriseId" type="xsd:string">

    < /wsdl:part>
    < wsdl:part name="authId" type="xsd:string">
    < /wsdl:part>
    < wsdl:part name="accountType" type="xsd:string">
    < /wsdl:part>
    < wsdl:part name="username" type="xsd:string">
    < /wsdl:part>
    < wsdl:part name="secondKey" type="xsd:string">
    < /wsdl:part>

  < /wsdl:message>
  < wsdl:message name="getProvisioningCertResponse">
    < wsdl:part name="getProvisioningCertReturn" type="tns1:ResultMessage">
    < /wsdl:part>
  < /wsdl:message>
  < wsdl:message name="preProvisionImsUserRequest">
    < wsdl:part name="sessionId" type="xsd:string">
    < /wsdl:part>
    < wsdl:part name="enterpriseId" type="xsd:string">

    < /wsdl:part>
    < wsdl:part name="initialPassword" type="xsd:string">
    < /wsdl:part>
    < wsdl:part name="attributes" type="impl:ArrayOf_tns1_NameValue">
    < /wsdl:part>
  < /wsdl:message>
  < wsdl:message name="unlockAccountRequest">
    < wsdl:part name="sessionId" type="xsd:string">

    < /wsdl:part>

    < wsdl:part name="enterpriseId" type="xsd:string">
    < /wsdl:part>
    < wsdl:part name="adminEntId" type="xsd:string">
    < /wsdl:part>
  < /wsdl:message>
  < wsdl:message name="setAccountCredentialWithSecondKeyResponse">
    < wsdl:part name="setAccountCredentialWithSecondKeyReturn" type="xsd:int">
    < /wsdl:part>
  < /wsdl:message>

  < wsdl:message name="addAccountCredentialWithSecondKeyResponse">
    < wsdl:part name="addAccountCredentialWithSecondKeyReturn" type="xsd:int">
    < /wsdl:part>
  < /wsdl:message>
  < wsdl:message name="setAccountCredentialResponse">
    < wsdl:part name="setAccountCredentialReturn" type="xsd:int">
    < /wsdl:part>
  < /wsdl:message>
  < wsdl:message name="revokeImsAccountUsingAttrsResponse">

    < wsdl:part name="revokeImsAccountUsingAttrsReturn" type="xsd:int">
    < /wsdl:part>
  < /wsdl:message>
  < wsdl:portType name="ProvisioningService">
    < wsdl:operation name="preProvisionImsUser" parameterOrder="sessionId enterpriseId initialPassword attributes">
      < wsdl:input message="impl:preProvisionImsUserRequest" name="preProvisionImsUserRequest">
      < /wsdl:input>
      < wsdl:output message="impl:preProvisionImsUserResponse" name="preProvisionImsUserResponse">
      < /wsdl:output>

    < /wsdl:operation>
    < wsdl:operation name="preProvisionImsUserWithEntId" parameterOrder="sessionId enterpriseId adminEntId initialPassword
attributes">
      < wsdl:input message="impl:preProvisionImsUserWithEntIdRequest" name="preProvisionImsUserWithEntIdRequest">
      < /wsdl:input>
      < wsdl:output message="impl:preProvisionImsUserWithEntIdResponse" name="preProvisionImsUserWithEntIdResponse">
      < /wsdl:output>
    < /wsdl:operation>
    < wsdl:operation name="createWallet" parameterOrder="sessionId enterpriseId walletId">
      < wsdl:input message="impl:createWalletRequest" name="createWalletRequest">

      < /wsdl:input>
      < wsdl:output message="impl:createWalletResponse" name="createWalletResponse">
      < /wsdl:output>
    < /wsdl:operation>
    < wsdl:operation name="getProvisioningCert" parameterOrder="sessionId enterpriseId">
      < wsdl:input message="impl:getProvisioningCertRequest" name="getProvisioningCertRequest">
      < /wsdl:input>
      < wsdl:output message="impl:getProvisioningCertResponse" name="getProvisioningCertResponse">
      < /wsdl:output>

    < /wsdl:operation>
    < wsdl:operation name="addAccountCredential" parameterOrder="sessionId enterpriseId authId accountType username
password">

```

```

        < wsdl:input message="impl:addAccountCredentialRequest" name="addAccountCredentialRequest">
    < /wsdl:input>
    < wsdl:output message="impl:addAccountCredentialResponse" name="addAccountCredentialResponse">
    < /wsdl:output>
    < /wsdl:operation>
    < wsdl:operation name="addAccountCredentialWithEntId" parameterOrder="sessionId enterpriseId adminEntId authId
accountType username password">
        < wsdl:input message="impl:addAccountCredentialWithEntIdRequest" name="addAccountCredentialWithEntIdRequest">

    < /wsdl:input>
    < wsdl:output message="impl:addAccountCredentialWithEntIdResponse" name="addAccountCredentialWithEntIdResponse">
    < /wsdl:output>
    < /wsdl:operation>
    < wsdl:operation name="addAccountCredentialWithAccountFields" parameterOrder="sessionId enterpriseId authId accountType
accountFields">
        < wsdl:input message="impl:addAccountCredentialWithAccountFieldsRequest"
name="addAccountCredentialWithAccountFieldsRequest">
    < /wsdl:input>
    < wsdl:output message="impl:addAccountCredentialWithAccountFieldsResponse"
name="addAccountCredentialWithAccountFieldsResponse">
    < /wsdl:output>

    < /wsdl:operation>
    < wsdl:operation name="addAccountCredentialWithSecondKey" parameterOrder="sessionId enterpriseId authId accountType
username password secondKey">
        < wsdl:input message="impl:addAccountCredentialWithSecondKeyRequest" name="addAccountCredentialWithSecondKeyRequest">
    < /wsdl:input>
    < wsdl:output message="impl:addAccountCredentialWithSecondKeyResponse"
name="addAccountCredentialWithSecondKeyResponse">
    < /wsdl:output>
    < /wsdl:operation>
    < wsdl:operation name="removeAccountCredential" parameterOrder="sessionId enterpriseId authId accountType username">
        < wsdl:input message="impl:removeAccountCredentialRequest" name="removeAccountCredentialRequest">

    < /wsdl:input>
    < wsdl:output message="impl:removeAccountCredentialResponse" name="removeAccountCredentialResponse">
    < /wsdl:output>
    < /wsdl:operation>
    < wsdl:operation name="removeAccountCredentialWithAccountFields" parameterOrder="sessionId enterpriseId authId
accountType accountFields">
        < wsdl:input message="impl:removeAccountCredentialWithAccountFieldsRequest"
name="removeAccountCredentialWithAccountFieldsRequest">
    < /wsdl:input>
    < wsdl:output message="impl:removeAccountCredentialWithAccountFieldsResponse"
name="removeAccountCredentialWithAccountFieldsResponse">
    < /wsdl:output>

    < /wsdl:operation>
    < wsdl:operation name="removeAccountCredentialWithSecondKey" parameterOrder="sessionId enterpriseId authId accountType
username secondKey">
        < wsdl:input message="impl:removeAccountCredentialWithSecondKeyRequest"
name="removeAccountCredentialWithSecondKeyRequest">
    < /wsdl:input>
    < wsdl:output message="impl:removeAccountCredentialWithSecondKeyResponse"
name="removeAccountCredentialWithSecondKeyResponse">
    < /wsdl:output>
    < /wsdl:operation>
    < wsdl:operation name="removeAccountCredentialWithEntId" parameterOrder="sessionId adminEntId enterpriseId authId
accountType username">
        < wsdl:input message="impl:removeAccountCredentialWithEntIdRequest" name="removeAccountCredentialWithEntIdRequest">

    < /wsdl:input>
    < wsdl:output message="impl:removeAccountCredentialWithEntIdResponse" name="removeAccountCredentialWithEntIdResponse">
    < /wsdl:output>
    < /wsdl:operation>
    < wsdl:operation name="setAccountCredential" parameterOrder="sessionId enterpriseId authId accountType username
password">
        < wsdl:input message="impl:setAccountCredentialRequest" name="setAccountCredentialRequest">
    < /wsdl:input>
    < wsdl:output message="impl:setAccountCredentialResponse" name="setAccountCredentialResponse">
    < /wsdl:output>

    < /wsdl:operation>
    < wsdl:operation name="setAccountCredentialWithAccountFields" parameterOrder="sessionId enterpriseId authId accountType
accountFields">
        < wsdl:input message="impl:setAccountCredentialWithAccountFieldsRequest"
name="setAccountCredentialWithAccountFieldsRequest">
    < /wsdl:input>
    < wsdl:output message="impl:setAccountCredentialWithAccountFieldsResponse"
name="setAccountCredentialWithAccountFieldsResponse">
    < /wsdl:output>
    < /wsdl:operation>
    < wsdl:operation name="setAccountCredentialWithSecondKey" parameterOrder="sessionId enterpriseId authId accountType
username password secondKey">
        < wsdl:input message="impl:setAccountCredentialWithSecondKeyRequest" name="setAccountCredentialWithSecondKeyRequest">

    < /wsdl:input>
    < wsdl:output message="impl:setAccountCredentialWithSecondKeyResponse"
name="setAccountCredentialWithSecondKeyResponse">
    < /wsdl:output>
    < /wsdl:operation>
    < wsdl:operation name="setAccountCredentialWithEntId" parameterOrder="sessionId enterpriseId adminEntId authId
accountType username password">
        < wsdl:input message="impl:setAccountCredentialWithEntIdRequest" name="setAccountCredentialWithEntIdRequest">
    < /wsdl:input>
    < wsdl:output message="impl:setAccountCredentialWithEntIdResponse" name="setAccountCredentialWithEntIdResponse">
    < /wsdl:output>

    < /wsdl:operation>
    < wsdl:operation name="revokeImsAccountUsingAttrs" parameterOrder="sessionId attributes">
        < wsdl:input message="impl:revokeImsAccountUsingAttrsRequest" name="revokeImsAccountUsingAttrsRequest">
    < /wsdl:input>
    < wsdl:output message="impl:revokeImsAccountUsingAttrsResponse" name="revokeImsAccountUsingAttrsResponse">
    < /wsdl:output>
    < /wsdl:operation>
    < wsdl:operation name="deleteImsAccountUsingAttrs" parameterOrder="sessionId attributes">

```

```

    < wsdl:input message="impl:deleteImsAccountUsingAttrsRequest" name="deleteImsAccountUsingAttrsRequest">
  </wsdl:input>
  < wsdl:output message="impl:deleteImsAccountUsingAttrsResponse" name="deleteImsAccountUsingAttrsResponse">
</wsdl:output>
</wsdl:operation>
< wsdl:operation name="revokeImsAccount" parameterOrder="sessionId enterpriseId">
  < wsdl:input message="impl:revokeImsAccountRequest" name="revokeImsAccountRequest">
</wsdl:input>
  < wsdl:output message="impl:revokeImsAccountResponse" name="revokeImsAccountResponse">
</wsdl:output>

</wsdl:operation>
< wsdl:operation name="revokeImsAccountWithEntId" parameterOrder="sessionId enterpriseId adminEntId">
  < wsdl:input message="impl:revokeImsAccountWithEntIdRequest" name="revokeImsAccountWithEntIdRequest">
</wsdl:input>
  < wsdl:output message="impl:revokeImsAccountWithEntIdResponse" name="revokeImsAccountWithEntIdResponse">
</wsdl:output>
</wsdl:operation>
< wsdl:operation name="deleteImsAccount" parameterOrder="sessionId enterpriseId">
  < wsdl:input message="impl:deleteImsAccountRequest" name="deleteImsAccountRequest">
</wsdl:input>
  < wsdl:output message="impl:deleteImsAccountResponse" name="deleteImsAccountResponse">
</wsdl:output>
</wsdl:operation>
< wsdl:operation name="deleteImsAccountWithEntId" parameterOrder="sessionId enterpriseId adminEntId">
  < wsdl:input message="impl:deleteImsAccountWithEntIdRequest" name="deleteImsAccountWithEntIdRequest">
</wsdl:input>
  < wsdl:output message="impl:deleteImsAccountWithEntIdResponse" name="deleteImsAccountWithEntIdResponse">
</wsdl:output>

</wsdl:operation>
< wsdl:operation name="getRegistrationStatus" parameterOrder="sessionId enterpriseId">
  < wsdl:input message="impl:getRegistrationStatusRequest" name="getRegistrationStatusRequest">
</wsdl:input>
  < wsdl:output message="impl:getRegistrationStatusResponse" name="getRegistrationStatusResponse">
</wsdl:output>
</wsdl:operation>
< wsdl:operation name="getUserAccounts" parameterOrder="sessionId enterpriseId">
  < wsdl:input message="impl:getUserAccountsRequest" name="getUserAccountsRequest">
</wsdl:input>
  < wsdl:output message="impl:getUserAccountsResponse" name="getUserAccountsResponse">
</wsdl:output>
</wsdl:operation>
< wsdl:operation name="getAllActiveUserIDs" parameterOrder="sessionId filter">
  < wsdl:input message="impl:getAllActiveUserIDsRequest" name="getAllActiveUserIDsRequest">
</wsdl:input>
  < wsdl:output message="impl:getAllActiveUserIDsResponse" name="getAllActiveUserIDsResponse">
</wsdl:output>

</wsdl:operation>
< wsdl:operation name="resetPassword" parameterOrder="sessionId enterpriseId newPassword">
  < wsdl:input message="impl:resetPasswordRequest" name="resetPasswordRequest">
</wsdl:input>
  < wsdl:output message="impl:resetPasswordResponse" name="resetPasswordResponse">
</wsdl:output>
</wsdl:operation>
< wsdl:operation name="unlockAccount" parameterOrder="sessionId enterpriseId adminEntId">
  < wsdl:input message="impl:unlockAccountRequest" name="unlockAccountRequest">
</wsdl:input>
  < wsdl:output message="impl:unlockAccountResponse" name="unlockAccountResponse">
</wsdl:output>
</wsdl:operation>
< wsdl:operation name="lockAccount" parameterOrder="sessionId enterpriseId adminEntId">
  < wsdl:input message="impl:lockAccountRequest" name="lockAccountRequest">
</wsdl:input>
  < wsdl:output message="impl:lockAccountResponse" name="lockAccountResponse">
</wsdl:output>

</wsdl:operation>
</wsdl:portType>
< wsdl:binding name="encentuate.ims.service.ProvisioningServiceSoapBinding" type="impl:ProvisioningService">
  < wsdl:soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  < wsdl:operation name="preProvisionImsUser">
    < wsdl:soap:operation soapAction=""/>
    < wsdl:input name="preProvisionImsUserRequest">
      < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
    </wsdl:input>
    < wsdl:output name="preProvisionImsUserResponse">
      < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
  < wsdl:operation name="preProvisionImsUserWithEntId">
    < wsdl:soap:operation soapAction=""/>
    < wsdl:input name="preProvisionImsUserWithEntIdRequest">
      < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
    </wsdl:input>
    < wsdl:output name="preProvisionImsUserWithEntIdResponse">
      < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
  < wsdl:operation name="createWallet">
    < wsdl:soap:operation soapAction=""/>
    < wsdl:input name="createWalletRequest">
      < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
    </wsdl:input>

```

```
< wsdl:output name="removeAccountCredentialWithSecondKeyResponse">
  < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost/ims/services/enctuate.ims.service.ProvisioningService" use="encoded"/>
< /wsdl:output>
< /wsdl:operation>
< wsdl:operation name="removeAccountCredentialWithEntId">
< wsdl:soap:operation soapAction="" />
< wsdl:input name="removeAccountCredentialWithEntIdRequest">
< wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

Appendix B. WSDL for provisioning service 71

```

        < wsdl:input name="deleteImsAccountRequest">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:input>

        < wsdl:output name="deleteImsAccountResponse">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:output>
      < /wsdl:operation>
      < wsdl:operation name="deleteImsAccountWithEntId">
        < wsdl:soap:operation soapAction="" />
        < wsdl:input name="deleteImsAccountWithEntIdRequest">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:input>

        < wsdl:output name="deleteImsAccountWithEntIdResponse">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:output>
      < /wsdl:operation>
      < wsdl:operation name="getRegistrationStatus">
        < wsdl:soap:operation soapAction="" />
        < wsdl:input name="getRegistrationStatusRequest">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:input>

        < wsdl:output name="getRegistrationStatusResponse">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:output>
      < /wsdl:operation>
      < wsdl:operation name="getUserAccounts">
        < wsdl:soap:operation soapAction="" />
        < wsdl:input name="getUserAccountsRequest">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:input>

        < wsdl:output name="getUserAccountsResponse">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:output>
      < /wsdl:operation>
      < wsdl:operation name="getAllActiveUserIDs">
        < wsdl:soap:operation soapAction="" />
        < wsdl:input name="getAllActiveUserIDsRequest">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:input>

        < wsdl:output name="getAllActiveUserIDsResponse">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:output>
      < /wsdl:operation>
      < wsdl:operation name="resetPassword">
        < wsdl:soap:operation soapAction="" />
        < wsdl:input name="resetPasswordRequest">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:input>

        < wsdl:output name="resetPasswordResponse">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:output>
      < /wsdl:operation>
      < wsdl:operation name="unlockAccount">
        < wsdl:soap:operation soapAction="" />
        < wsdl:input name="unlockAccountRequest">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:input>

        < wsdl:output name="unlockAccountResponse">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:output>
      < /wsdl:operation>
      < wsdl:operation name="lockAccount">
        < wsdl:soap:operation soapAction="" />
        < wsdl:input name="lockAccountRequest">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:input>

        < wsdl:output name="lockAccountResponse">
          < wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="https://localhost/ims/services/encentuate.ims.service.ProvisioningService" use="encoded"/>
        < /wsdl:output>
      < /wsdl:operation>
    < /wsdl:binding>
    < wsdl:service name="ProvisioningServiceService">
      < wsdl:port binding="impl:encentuate.ims.service.ProvisioningServiceSoapBinding"
name="encentuate.ims.service.ProvisioningService">
        < wsdl:soap:address location="https://localhost/ims/services/encentuate.ims.service.ProvisioningService"/>
      < /wsdl:port>
    < /wsdl:service>
  < /wsdl:definitions>

```

Appendix C. Using the Keytool for provisioning

A **Keytool** is a utility that creates keystores or imports certificates.

About this task

When using the Sun **Keytool** utility, see the Java **Keytool** documentation. Go to <http://www.oracle.com> and search the JDK documentation for “Keytool - key and certificate management tool”.

When using the IBM **Keytool** utility, see the Java **Keytool** documentation in: <http://www.ibm.com/developerworks/java/jdk/> and search for “Class KeyTool”.

The following command-line example shows you how to use the **Keytool**:

```
keytool -import -alias ims -file <certificate_file_path> -keystore  
<truststore_path> -storepass <truststore_password>
```

See the procedure for information about obtaining a file certificate. You must first obtain a file certificate before running the **Keytool**.

Procedure

1. From the Microsoft Internet Explorer, go to <https://<imsserver>/> where <imsserver> is the host name of the IMS Server.
2. Click the lock icon and select **View certificates**.
3. Click the **Details** tab.
4. Click **Copy to File**.
5. Click **Next**.
6. Select the **Base-64 encoded X.509** format.
7. Click **Save**.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law :

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to

IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Trademarks

IBM, the IBM logo, and ibm.com[®] are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information; at www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.



Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Glossary

AccessAdmin. A web-based management console that Administrators and Helpdesk officers use to administer the IMS Server and to manage users and policies.

AccessAgent plug-in. A piece of script, written in VBscript or Javascript, that is embedded within an AccessProfile to perform custom checking of conditions or to execute custom actions. It is used for extending the capability of an AccessProfile beyond the built-in triggers and actions.

AccessAgent. The client software that manages the identity of the user, authenticates the user, and automates single sign-on and sign-off.

AccessAssistant. The web-based interface that helps users to reset their passwords and retrieve their application credentials.

AccessProfile widget / widget. An independent AccessProfile that consists of pinnable states, which can be used to build another AccessProfile.

AccessProfiles. AccessAgent uses these XML specifications to identify application screens that it can perform single sign-on and automation.

AccessStudio. An application used by Administrators for creating and maintaining AccessProfiles.

Account data bag. A data structure that holds user credentials in memory while single sign-on is performed on an application.

Account data item template. A template that defines the properties of an account data item.

Account data item. The user credentials required for login.

Account data template. A template that defines the format of account data to be stored for credentials captured by using a specific AccessProfile.

Account data. The login information required to verify an authentication service. It can be the user name, password, and the authentication service which the login information is stored.

Action. In profiling, an act that can be performed in response to a trigger. For example, automatic filling of user name and password details as soon as a sign-on window displays.

Active Directory (AD). A hierarchical directory service that enables centralized, secure management of an entire network, which is a central component of the Microsoft Windows platform.

Active Directory credentials. The Active Directory user name and password.

Active Directory password synchronization. An IBM Security Access Manager for Enterprise Single Sign-On feature that synchronizes the ISAM ESSO password with the Active Directory password.

Active RFID (ARFID). ARFID is both a second authentication factor and a presence detector. It can detect the presence of a user and AccessAgent can be configured to perform specific actions. In previous releases, it is called Active Proximity Badge.

ActiveCode. Short-lived authentication codes that are generated and verified by IBM Security Access Manager for Enterprise Single Sign-On. There are two types of ActiveCodes: Mobile ActiveCodes and Predictive ActiveCodes.

Mobile ActiveCodes are generated by IBM Security Access Manager for Enterprise Single Sign-On and dispatched to the mobile phone or email account of the user. Predictive ActiveCodes, or One Time Passwords, are generated from OTP tokens when a user presses its button.

Combined with alternative channels or devices, ActiveCodes provide effective second-factor authentication.

Administrator. A person responsible for administrative tasks such as access authorization and content management. Administrators can also grant levels of authority to users.

Application policies. A collection of policies and attributes governing access to applications.

Application programming interface (API). An interface that allows an application program written in a high-level language to use specific data or functions of the operating system or another program.

Application. One or more computer programs or software components that provide a function in direct support of a specific business process or processes. In AccessStudio, it is the system that provides the user interface for reading or entering the authentication credentials.

Audit. A process that logs the user, Administrator, and Helpdesk activities.

Authentication factor. The different devices, biometrics, or secrets required as credentials for validating digital identities. Examples of authentication

factors are passwords, smart card, RFID, biometrics, and one-time password tokens.

Authentication service. In IBM Security Access Manager for Enterprise Single Sign-On, a service that verifies the validity of an account against their own user store or against a corporate directory. Identifies the authentication service associated with a screen. Account data saved under a particular authentication service is retrieved and auto-filled for the logon screen that is defined. Account data captured from the logon screen defined is saved under this authentication service.

Authorization code. An alphanumeric code generated for administrative functions, such as password resets or two-factor authentication bypass with AccessAgent, AccessAssistant, and Web Workplace.

Auto-capture. A process that allows a system to collect and reuse user credentials for different applications. These credentials are captured when the user enters information for the first time, and then stored and secured for future use.

Automatic sign-on. A feature where users can log on to the sign-on automation system and the system logs on the user to all other applications.

Base distinguished name. A name that indicates the starting point for searches in the directory server.

Bidirectional language. A language that uses a script, such as Arabic and Hebrew, whose general flow of text proceeds horizontally from right to left, but numbers, English, and other left-to-right language text are written from left to right.

Bind distinguished name. A name that specifies the credentials for the application server to use when connecting to a directory service. The distinguished name uniquely identifies an entry in a directory. See also *Distinguished name*.

Biometrics. The identification of a user based on a physical characteristic of the user, such as a fingerprint, iris, face, voice, or handwriting.

Card Serial Number (CSN). A unique data item that identifies a hybrid smart card. It has no relation to the certificates installed in the smart card

Cell. In WebSphere Application Server, a cell is a virtual unit that consists of a deployment manager and one or more nodes.

Certificate authority (CA). A trusted organization or company that issues the digital certificates. The certificate authority typically verifies the identity of the individuals who are granted the unique certificate.

IMS Server Certificate. Used in IBM Security Access Manager for Enterprise Single Sign-On. The IMS Server Certificate allows clients to identify and authenticate an IMS Server.

Client AccessAgent. AccessAgent installed and running on the client machine.

Client workstation, client machine, client computers. Computers where AccessAgent installed.

Clinical Context Object Workgroup (CCOW). A vendor independent standard, for the interchange of information between clinical applications in the healthcare industry.

Clustering. In WebSphere Application Server, clustering is the ability to group application servers.

Clusters. A group of application servers that collaborate for the purposes of workload balancing and failover.

Command line interface. A computer interface in which the input command is a string of text characters.

Credentials. Information acquired during authentication that describes a user, group associations, or other security-related identity attributes, and that is used to perform services such as authorization, auditing, or delegation. For example, a user ID and password are credentials that allow access to network and system resources.

Cryptographic application programming interface (CAPI). An application programming interface that provides services to enable developers to secure applications using cryptography. It is a set of dynamically-linked libraries that provides an abstraction layer which isolates programmers from the code used to encrypt the data.

Cryptographic Service Provider (CSP). A feature of the i5/OS[®] operating system that provides APIs. The CCA Cryptographic Service Provider enables a user to run functions on the 4758 Coprocessor.

Data source. The means by which an application accesses data from a database.

Database (DB) server. A software program that uses a database manager to provide database services to software programs or computers.

DB2[®]. A family of IBM licensed programs for relational database management.

Deployment manager profiles. A WebSphere Application Server runtime environment that manages operations for a logical group, or cell, of other servers.

Deployment manager. A server that manages and configures operations for a logical group or cell of other servers.

Deprovision. To remove a service or component. For example, to deprovision an account means to delete an account from a resource.

Desktop application. Application that runs in a desktop.

Desktop Manager. Manages concurrent user desktops on a single workstation

Direct auth-info. In profiling, direct auth-info is a direct reference to an existing authentication service.

Directory service. A directory of names, profile information, and computer addresses of every user and resource on the network. It manages user accounts and network permissions. When a user name is sent, it returns the attributes of that individual, which might include a telephone number, or an email address. Directory services use highly specialized databases that are typically hierarchical in design and provide fast lookups.

Directory. A file that contains the names and controlling information for objects or other directories.

Disaster recovery site. A secondary location for the production environment in case of a disaster.

Disaster recovery. The process of restoring a database, system, policies after a partial or complete site failure that was caused by a catastrophic event such as an earthquake or fire. Typically, disaster recovery requires a full backup at another location.

Distinguished name. The name that uniquely identifies an entry in a directory. A distinguished name is made up of attribute:value pairs, separated by commas. For example, CN=person name and C=country or region.

Distributed IMS Server. The IMS Servers are deployed in multiple geographical locations.

Domain name server (DNS). A server program that supplies name-to-address conversion by mapping domain names to IP addresses.

Dynamic link library (DLL). A file containing executable code and data bound to a program at load time or run time, rather than during linking. The code and data in a DLL can be shared by several applications simultaneously.

Enterprise directory. A directory of user accounts that define IBM Security Access Manager for Enterprise Single Sign-On users. It validates user credentials during sign-up and login, if the password is synchronized with the enterprise directory password. An example of an enterprise directory is Active Directory.

Enterprise Single Sign-On (ESSO). A mechanism that allows users to log on to all applications deployed in the enterprise by entering a user ID and other credentials, such as a password.

Enterprise user name. The user name of a user account in the enterprise directory.

ESSO audit logs. A log file that contains a record of system events and responses. ESSO audit logs are stored in the IMS Database.

ESSO Credential Provider. Previously known as the Encentuate Credential Provider (EnCredentialProvider), this is the IBM Security Access Manager for Enterprise Single Sign-On GINA for Windows Vista and Windows 7.

ESSO credentials. The ISAM ESSO user name and password.

ESSO GINA. Previously known as the Encentuate GINA (EnGINA). IBM Security Access Manager for Enterprise Single Sign-On GINA provides a user interface that is integrated with authentication factors and provide password resets and second factor bypass options.

ESSO Network Provider. Previously known as the Encentuate Network Provider (EnNetworkProvider). An AccessAgent module that captures the Active Directory server credentials and uses these credentials to automatically log on the users to their Wallet.

ESSO password. The password that secures access to the user Wallet.

Event code. A code that represents a specific event that is tracked and logged into the audit log tables.

Failover. An automatic operation that switches to a redundant or standby system in the event of a software, hardware, or network interruption.

Fast user switching. A feature that allows users to switch between user accounts on a single workstation without quitting and logging out of applications.

Federal Information Processing Standard (FIPS). A standard produced by the National Institute of Standards and Technology when national and international standards are nonexistent or inadequate to satisfy the U.S. government requirements.

Fix pack. A cumulative collection of fixes that is made available between scheduled refresh packs, manufacturing refreshes, or releases. It is intended to allow customers to move to a specific maintenance level.

Fully qualified domain name (FQDN). In Internet communications, the name of a host system that

includes all of the subnames of the domain name. An example of a fully qualified domain name is `rchland.vnet.ibm.com`.

Graphical Identification and Authentication (GINA).

A dynamic link library that provides a user interface that is tightly integrated with authentication factors and provides password resets and second factor bypass options.

Group Policy Object (GPO). A collection of group policy settings. Group policy objects are the documents created by the group policy snap-in. Group policy objects are stored at the domain level, and they affect users and computers contained in sites, domains, and organizational units.

High availability (HA). The ability of IT services to withstand all outages and continue providing processing capability according to some predefined service level. Covered outages include both planned events, such as maintenance and backups, and unplanned events, such as software failures, hardware failures, power failures, and disasters.

Host name. In Internet communication, the name given to a computer. The host name might be a fully qualified domain name such as `mycomputer.city.company.com`, or it might be a specific subname such as `mycomputer`.

Hot key. A key sequence used to shift operations between different applications or between different functions of an application.

Hybrid smart card. An ISO-7816 compliant smart card which contains a public key cryptography chip and an RFID chip. The cryptographic chip is accessible through contact interface. The RFID chip is accessible through contactless (RF) interface.

IBM HTTP server. A web server. IBM offers a web server, called the IBM HTTP Server, that accepts requests from clients and forward to the application server.

IMS Bridge. A module embedded in third-party applications and systems to call to IMS APIs for provisioning and other purposes.

IMS Configuration Utility. A utility of the IMS Server that allows Administrators to manage lower-level configuration settings for the IMS Server.

IMS Configuration wizard. Administrators use the wizard to configure the IMS Server during installation.

IMS Connector. A module that connects IMS to external systems to dispatch a mobile active code to a messaging gateway.

IMS data source. A WebSphere Application Server configuration object that defines the location and parameters for accessing the IMS database.

IMS Database. The relational database where the IMS Server stores all ESSO system, machine, and user data and audit logs.

IMS Root CA. The root certificate authority that signs certificates for securing traffic between AccessAgent and IMS Server.

IMS Server. An integrated management system for ISAM ESSO that provides a central point of secure access administration for an enterprise. It enables centralized management of user identities, AccessProfiles, authentication policies, provides loss management, certificate management, and audit management for the enterprise.

Indirect auth-info. In profiling, indirect auth-info is an indirect reference to an existing authentication service.

Interactive graphical mode. A series of panels that prompts for information to complete the installation.

IP address. A unique address for a device or logical unit on a network that uses the Internet Protocol standard.

Java Management Extensions (JMX). A means of doing management of and through Java technology. JMX is a universal, open extension of the Java programming language for management that can be deployed across all industries, wherever management is needed.

Java runtime environment (JRE). A subset of a Java developer kit that contains the core executable programs and files that constitute the standard Java platform. The JRE includes the Java virtual machine (JVM), core classes, and supporting files.

Java virtual machine (JVM). A software implementation of a processor that runs compiled Java code (applets and applications).

Keystore. In security, a file or a hardware cryptographic card where identities and private keys are stored, for authentication and encryption purposes. Some keystores also contain trusted, or public, keys.

Lightweight Directory Access Protocol (LDAP). An open protocol that uses TCP/IP to provide access to directories that support an X.500 model. An LDAP can be used to locate people, organizations, and other resources in an Internet or intranet directory.

Lightweight mode. A Server AccessAgent mode. Running in lightweight mode reduces the memory footprint of AccessAgent on a Citrix/Terminal Server and improves the single sign-on startup duration.

Load balancing. The monitoring of application servers and management of the workload on servers. If one server exceeds its workload, requests are forwarded to another server with more capacity.

Lookup user. A user who is authenticated in the Enterprise Directory and searches for other users. IBM Security Access Manager for Enterprise Single Sign-On uses the lookup user to retrieve user attributes from the Active Directory or LDAP enterprise repository.

Main AccessProfile. The AccessProfile that contains one or more AccessProfile widgets

Managed node. A node that is federated to a deployment manager and contains a node agent and can contain managed servers.

Microsoft Cryptographic application programming interface (CAPI). An interface specification from Microsoft for modules that provide cryptographic functionality and that allow access to smart cards.

Mobile ActiveCode (MAC). A one-time password that is used by users for two-factor authentication in Web Workplace, AccessAssistant, and other applications. This OTP is randomly generated and dispatched to user through SMS or email.

Mobile authentication. An authentication factor which allows mobile users to sign-on securely to corporate resources from anywhere on the network.

Network deployment. Also known as a clustered deployment. A type of deployment where the IMS Server is deployed on a WebSphere Application Server cluster.

Node agent. An administrative agent that manages all application servers on a node and represents the node in the management cell.

Nodes. A logical group of managed servers.

One-Time Password (OTP). A one-use password generated for an authentication event, sometimes communicated between the client and the server through a secure channel.

OTP token. A small, highly portable hardware device that the owner carries to authorize access to digital systems and physical assets.

Password aging. A security feature by which the superuser can specify how often users must change their passwords.

Password complexity policy. A policy that specifies the minimum and maximum length of the password, the minimum number of numeric and alphabetic characters, and whether to allow mixed uppercase and lowercase characters.

Personal applications. Windows and web-based applications where AccessAgent can store and enter credentials.

Some examples of personal applications are web-based mail sites such as Company Mail, Internet banking sites, online shopping sites, chat, or instant messaging programs.

Personal desktop. The desktop is not shared with any other users.

Personal Identification Number (PIN). In Cryptographic Support, a unique number assigned by an organization to an individual and used as proof of identity. PINs are commonly assigned by financial institutions to their customers.

Pinnable state. A state from the AccessProfile widget that is declared as 'Can be pinned in another AccessProfile'.

Pinned state. A pinnable state that is attached to a state in the main AccessProfile.

Policy template. A predefined policy form that helps users define a policy by providing the fixed policy elements that cannot be changed and the variable policy elements that can be changed.

Portal. A single, secure point of access to diverse information, applications, and people that can be customized and personalized.

Presence detector. A device that, when fixed to a computer, detects when a person moves away from it. This device eliminates manually locking the computer upon leaving it for a short time.

Primary authentication factor. The IBM Security Access Manager for Enterprise Single Sign-On password or directory server credentials.

Private desktop. Under this desktop scheme, users have their own Windows desktops in a workstation. When a previous user return to the workstation and unlocks it, AccessAgent switches to the desktop session of the previous user and resumes the last task.

Private key. In computer security, the secret half of a cryptographic key pair that is used with a public key algorithm. The private key is known only to its owner. Private keys are typically used to digitally sign data and to decrypt data that has been encrypted with the corresponding public key.

Provisioning API. An interface that allows IBM Security Access Manager for Enterprise Single Sign-On to integrate with user provisioning systems.

Provisioning bridge. An automatic IMS Server credential distribution process with third party provisioning systems that uses API libraries with a SOAP connection.

Provisioning system. A system that provides identity lifecycle management for application users in enterprises and manages their credentials.

Provision. To provide, deploy, and track a service, component, application, or resource.

Public Key Cryptography Standards. A set of industry-standard protocols used for secure information exchange on the Internet. Domino® Certificate Authority and Server Certificate Administration applications can accept certificates in PKCS format.

Published application. Application installed on Citrix XenApp server that can be accessed from Citrix ICA Clients.

Published desktop. A Citrix XenApp feature where users have remote access to a full Windows desktop from any device, anywhere, at any time.

Radio Frequency Identification (RFID). An automatic identification and data capture technology that identifies unique items and transmits data using radio waves.

Random password. An arbitrarily generated password used to increase authentication security between clients and servers.

Registry hive. In Windows systems, the structure of the data stored in the registry.

Registry. A repository that contains access and configuration information for users, systems, and software.

Remote Authentication Dial-In User Service (RADIUS). An authentication and accounting system that uses access servers to provide centralized management of access to large networks.

Remote Desktop Protocol (RDP). A protocol that facilitates remote display and input over network connections for Windows-based server applications. RDP supports different network topologies and multiple connections.

Replication. The process of maintaining a defined set of data in more than one location. Replication involves copying designated changes for one location (a source) to another (a target) and synchronizing the data in both locations.

Revoke. To remove a privilege or an authority from an authorization identifier.

Root certificate authority (CA). The certificate authority at the top of the hierarchy of authorities by which the identity of a certificate holder can be verified.

Scope. A reference to the applicability of a policy, at the system, user, or machine level.

Secret question. A question whose answer is known only to the user. A secret question is used as a security feature to verify the identity of a user.

Secure Remote Access. The solution that provides web browser-based single sign-on to all applications from outside the firewall.

Secure Sockets Layer (SSL). A security protocol that provides communication privacy. With SSL, client/server applications can communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery.

Secure Sockets Layer virtual private network (SSL VPN). A form of VPN that can be used with a standard web browser.

Security Token Service (STS). A web service used for issuing and exchanging of security tokens.

Security trust service chain. A group of module instances that are configured for use together. Each module instance in the chain is called in turn to perform a specific function as part of the overall processing of a request.

Self-service features. Features in IBM Security Access Manager for Enterprise Single Sign-On which users can use to perform basic tasks such as resetting passwords and secrets with minimal assistance from Help desk or your Administrator.

Serial ID Service Provider Interface (SPI). A programmatic interface intended for integrating AccessAgent with third-party Serial ID devices used for two-factor authentication.

Serial number. A unique number embedded in the IBM Security Access Manager for Enterprise Single Sign-On Keys, which is unique to each Key and cannot be changed.

Server AccessAgent. AccessAgent deployed on a Microsoft Windows Terminal Server or a Citrix server.

Server locator. A locator that groups a related set of web applications that require authentication by the same authentication service. In AccessStudio, server locators identify the authentication service with which an application screen is associated.

Service Provider Interface (SPI). An interface through which vendors can integrate any device with serial numbers with IBM Security Access Manager for Enterprise Single Sign-On and use it as a second factor in AccessAgent.

Session management. Management of user session on private desktops and shared desktops.

Shared desktop. A desktop configuration where multiple users share a generic Windows desktop.

Shared workstation. A workstation shared among users.

Sign up. To request a resource.

sign-on automation. A technology that works with application user interfaces to automate the sign-on process for users.

sign-on information. Information required to provide access to users to any secure application. This information can include user names, passwords, domain information, and certificates.

Signature. In profiling, unique identification information for any application, window, or field.

Silent mode. A method for installing or uninstalling a product component from the command line with no GUI display. When using silent mode, you specify the data required by the installation or uninstallation program directly on the command line or in a file (called an option file or response file).

Simple Mail Transfer Protocol (SMTP). An Internet application protocol for transferring mail among users of the Internet.

Simple Object Access Protocol (SOAP). A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. SOAP can be used to query and return information and invoke services across the Internet.

Single sign-on. An authentication process in which a user can access more than one system or application by entering a single user ID and password.

Smart card middleware. Software that acts as an interface between smart card applications and the smart card hardware. Typically the software consists of libraries that implement PKCS#11 and CAPI interfaces to smart cards.

Smart card. An intelligent token that is embedded with an integrated circuit chip that provides memory capacity and computational capabilities.

Stand-alone deployment. A deployment where the IMS Server is deployed on an independent WebSphere Application Server profile.

Stand-alone server. A fully operational server that is managed independently of all other servers, and it uses its own administrative console.

Strong authentication. A solution that uses multi-factor authentication devices to prevent unauthorized access to confidential corporate information and IT networks, both inside and outside the corporate perimeter.

Strong digital identity. An online persona that is difficult to impersonate, possibly secured by private keys on a smart card.

System modal message. A system dialog box that is typically used to display important messages. When a system modal message is displayed, nothing else can be selected on the screen until the message is closed.

Terminal emulator. A program that allows a device such as a microcomputer or personal computer to enter and receive data from a computer system as if it were a particular type of attached terminal

Thin client. A client machine that has little or no installed software. It has access to applications and desktop sessions that is running on network servers that are connected to it. A thin client machine is an alternative to a full-function client such as a workstation.

Tivoli Common Reporting tool. A reporting component that you can use to create, customize, and manage reports.

Tivoli Identity Manager adapter. An intermediary software component that allows IBM Security Access Manager for Enterprise Single Sign-On to communicate with Tivoli Identity Manager.

Transparent screen lock. A feature that, when enabled, permits users to lock their desktop screens but still see the contents of their desktop.

Trigger. In profiling, an event that causes transitions between states in a states engine, such as, the loading of a web page or the appearance of window on the desktop.

Trust service chain. A chain of modules operating in different modes. For example: validate, map and issue.

Truststore. In security, a storage object, either a file or a hardware cryptographic card, where public keys are stored in the form of trusted certificates, for authentication purposes in web transactions. In some applications, these trusted certificates are moved into the application keystore to be stored with the private keys.

TTY (terminal type). A generic device driver for a text display. A tty typically performs input and output on a character-by-character basis.

Two-factor authentication. The use of two factors to authenticate a user. For example, the use of password and an RFID card to log on to AccessAgent.

Uniform resource identifier. A compact string of characters for identifying an abstract or physical resource.

User credential. Information acquired during authentication that describes a user, group associations, or other security-related identity attributes, and that is used to perform services such as authorization, auditing, or delegation. For example, a user ID and password are credentials that allow access to network and system resources.

User deprovisioning. Removing the user account from IBM Security Access Manager for Enterprise Single Sign-On.

User provisioning. The process of signing up a user to use IBM Security Access Manager for Enterprise Single Sign-On.

Virtual appliance. A virtual machine image with a specific application purpose that is deployed to virtualization platforms.

Virtual channel connector. A connector that is used in a terminal services environment. The virtual channel connector establishes a virtual communication channel to manage the remote sessions between the Client AccessAgent component and the Server AccessAgent.

Virtual Member Manager (VMM). A WebSphere Application Server component that provides applications with a secure facility to access basic organizational entity data such as people, logon accounts, and security roles.

Virtual Private Network (VPN). An extension of a company intranet over the existing framework of either a public or private network. A VPN ensures that the data that is sent between the two endpoints of its connection remains secure.

Visual Basic (VB). An event-driven programming language and integrated development environment (IDE) from Microsoft.

Wallet caching. When performing single sign-on for an application, AccessAgent retrieves the logon credentials from the user credential Wallet. The user credential Wallet is downloaded on the user machine and stored securely on the IMS Server. So users can access their Wallet even when they log on to IBM Security Access Manager for Enterprise Single Sign-On from a different machine later.

Wallet manager. The IBM Security Access Manager for Enterprise Single Sign-On GUI component that users can use to manage application credentials in the personal identity Wallet.

Wallet Password. A password that secures access to the Wallet.

Wallet. A secured data store of access credentials of a user and related information, which includes user IDs, passwords, certificates, encryption keys.

Web server. A software program that is capable of servicing Hypertext Transfer Protocol (HTTP) requests.

Web service. A self-contained, self-describing modular application that can be published, discovered, and invoked over a network using standard network protocols. Typically, XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available.

Web Workplace. A web-based interface that users can log on to enterprise web applications by clicking links without entering the passwords for individual applications. This interface can be integrated with the existing portal or SSL VPN of the customer.

WebSphere Administrative console. A graphical administrative Java application client that makes method calls to resource beans in the administrative server to access or modify a resource within the domain.

WebSphere Application Server profile. The WebSphere Application Server administrator user name and profile. Defines the runtime environment.

WebSphere Application Server. Software that runs on a web server and that can deploy, integrate, execute, and manage e-business applications.

Windows logon screen, Windows logon UI mode. The screen where users enter their user name and password to log on to the Windows desktop.

Windows native fast user switching. A Windows XP feature which allows users to quickly switch between user accounts.

Windows Terminal Services. A Microsoft Windows component that users use to access applications and data on a remote computer over a network.

WS-Trust. A web services security specification that defines a framework for trust models to establish trust between web services.

Index

A

- accessibility viii
- Active Directory Application Mode 55
- Active Directory Lightweight Directory Service
 - See* Active Directory Application Mode
- ADAM 55
- API types 6

B

- books
 - See* publications

C

- command-line tools 10
- conventions
 - typeface ix

D

- directory names, notation x

E

- education
 - See* Tivoli technical training
- environment variables, notation x

I

- IMS Server
 - IMS Bridge setup 7, 8, 10, 12
 - provisioning setup 3
 - SOAP API setup 31
- ims.xml file 3

J

- Java Provisioning API
 - certificate store for IMS Bridge 7
 - IMS Bridge parameter descriptions 8
 - invoking the IMS Bridge API 10
 - sample configuration file 7

K

- keytool utility 73

M

- manuals
 - See* publications

N

- notation
 - environment variables x
 - path names x
 - typeface x

O

- online publications
 - accessing viii
- ordering publications viii

P

- path names, notation x
- provisioning
 - See also* Provisioning API
 - about 1
 - features 2
 - file certificates 73
 - integration with other systems 3
 - Java API 7, 8, 9, 10, 12
 - keytool 73
 - Provisioning Agent
 - about 53
 - features 54
 - installing 55
 - requirements 54
 - setup 56, 57
 - starting 57
 - workflows 54
 - SOAP API 28, 30, 31
- Provisioning Agent
 - See* provisioning
- Provisioning API 5, 6
 - about 5
 - adding application accounts 51
 - data types for SOAP API 32
 - de-provisioning users 51
 - deleting application accounts 51
 - Java API 7, 12
 - maintaining accounts 50
 - provisioning new users 50
 - resetting application passwords 51
 - setting up accounts 50
 - SOAP API 28, 30, 31
 - provisioning service operations 36
 - server authentication operations 35
 - system requirements 6
 - WSDL for provisioning service 63
 - WSDL for server authentication 61
- publications vi
 - accessing online viii
 - ordering viii

S

- SOAP API
 - server authentication operations 35
- SOAP API Provisioning usage types
 - account list in Wallets 30
 - active user enterprise IDs 30
 - deleting users 30
 - provisioning typical users 29
 - removing application account credentials 30
 - resetting application passwords 29
 - resetting passwords 30
 - revoking users 30
 - user status 30
- SOAP clients
 - developing using Apache Axis 31
 - developing using Visual Studio .NET 31
- system requirements
 - Provisioning API 6

T

- Tivoli Information Center viii
- Tivoli technical training ix
- Tivoli user groups viii
- training, Tivoli technical ix
- typeface conventions ix

U

- user groups, Tivoli viii

V

- variables, notation for x



Printed in USA

SC23-9957-03

