CA Identity Manager

Java Connector Server Implementation Guide

r12.5 SP16

CA Technologies Product References

This document references the following CA products:

- CA Identity Manager

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Java Connector Server

This section contains the following topics:

Knowledge Requirements

This guide is intended for administrators who have the following technical background:

- An understanding of J2EE architecture and standards
- Experience with application servers and related tasks
- Familiarity with Provisioning Manager and CA Identity Manager

About the Java Connector Server

The Java Connector Server (JCS) is a server component which handles hosting, routing to, and management of Java connectors. The JCS provides a Java alternative to the C++ Connector Server (CCS). However, it has a Java API instead of a C++ API, which allows connectors to be implemented in Java. In addition, the JCS is data-driven rather than code-driven, which allows it to do much of the connector's work for it.

The *Provisioning Server* handles provisioning of users, and then delegates to connectors (using the Java CS or CCS) to manage endpoint accounts, groups, and so on.

Note: For the most current technical information, see the JavaDoc included with the JCS SDK install.

Endpoint Systems

An *endpoint* is a specific installation of a platform or application, such as Active Directory or Microsoft Exchange, which communicates with CA Identity Manager to synchronize information. A connector server uses a connector to manage an endpoint.

A *connector* is the software that enables communication between CA Identity Manager and an endpoint system. You can generate a dynamic connector using Connector Xpress, and you can develop a custom static connector in Java.

For each endpoint that you want to manage, you must have a connector. Connectors are responsible for representing each of the object classes in your endpoint in a consistent manner. Connectors translate add, modify, delete, rename, and search LDAP operations on those objects into corresponding actions against the endpoint system.

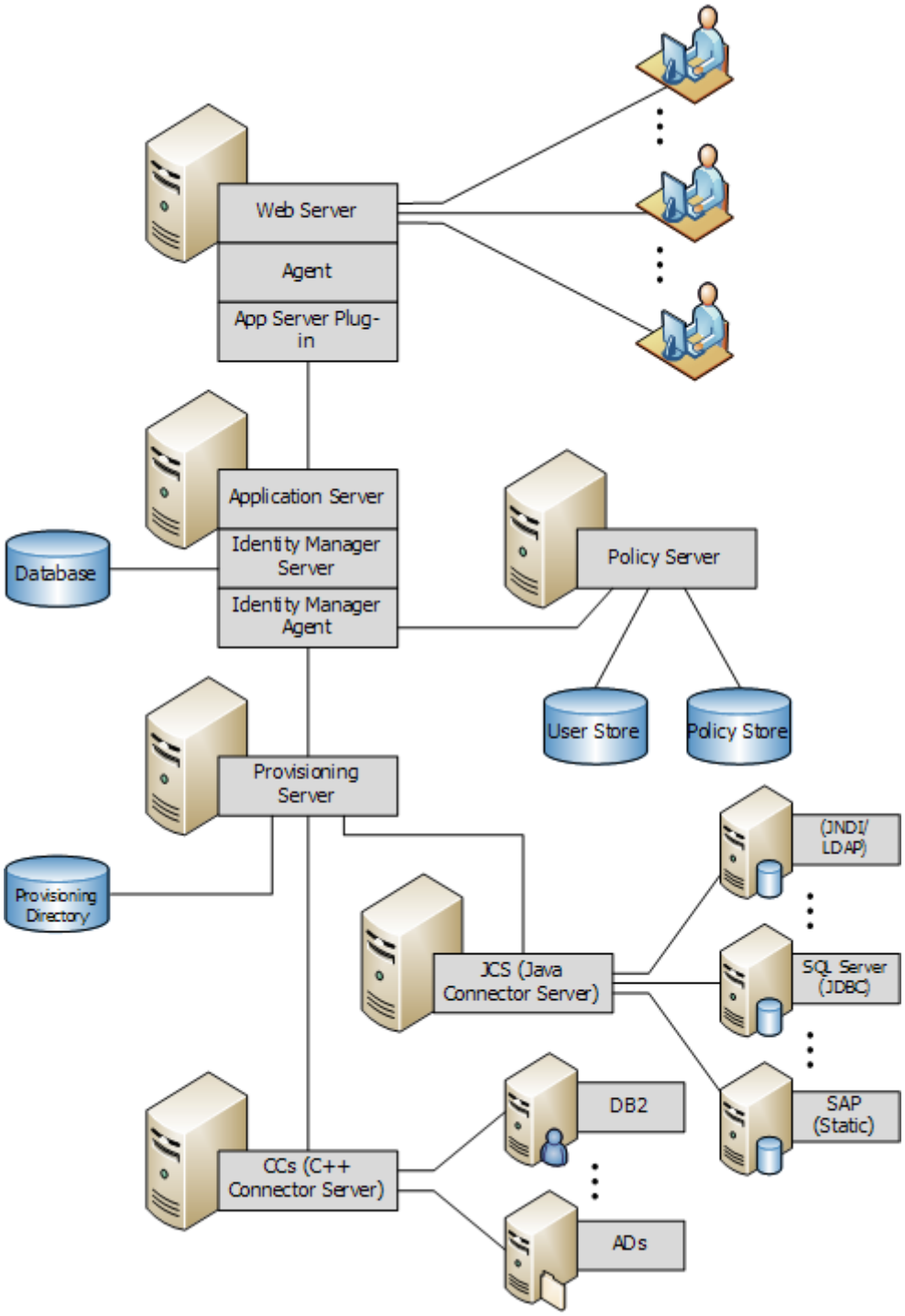
Users use Connector Xpress to generate and maintain XML metadata for JDBC and JNDI dynamic connectors. Developers can also maintain data for other connectors manually, or adjust metadata for released connectors (for instance adding site-specific mappings for custom attributes).

Connector Deployment

A connector's software is deployed to a Connector Server using an implementation bundle .jar file for a JCS or as a dynamic-link library or shared library for a C++ Connector Server. The Connector Server then creates connector instances when requests are received to add new endpoints.

JCS Architecture

This following illustration shows the JCS in a typical installation.



JCS Framework

The JCS makes extensive use of third party open source Java software libraries. It is built on top of Apache Directory Service (ApacheDS) and relies on the Spring Framework for XML configuration support.

You can find documentation at the following web sites:

- http://directory.apache.org
- http://www.springframework.org

The JCS framework supports:

- Run-time configurable data model and operation bindings using XML metadata
- Mapping of object and attribute names to / from LDAP to native equivalents
- Built-in connection resiliency
- Consistent data representation, validation, and conversion
- Extension and reuse of existing connector components
- (Optional) Reverse association handling, where associative relationships need only be expressed in one direction

What JCS Does

The JCS provides services to the connector implementations deployed to it while forcing standardized behavior in the same way that a J2EE application server provides for its hosted applications.

JCS provides a framework and a standard set of service to ease connector development. JCS:

- Uses XML files to simplify development and allow uniform configuration of all attributes.
- Enforces standardized representation of data types.
- Enables reuse among connector implementations, thus reducing the time, effort, and risk required to develop new connectors.

Connectors

A *connector* is the software that enables communication between a Provisioning Server and an endpoint system. The connector virtual directory maps LDAP operations such as add, modify, delete, search, and rename to equivalent APIs calls and protocols specific to endpoint systems.

Note: Traditionally, Provisioning Manager connector implementations (previously known as Options) are written in C++ and are deployed to the C++ Connector Server. These C++ connectors perform the same job as Java connectors, and the C++ Connector Server performs a similar role to the JCS although it provides less assistance to the connector developer. For more information, see the *Programming Guide for Provisioning.*

Connector Interfaces

Connectors are responsible for translating incoming LDAP operations to the equivalent operations on endpoint systems. Each connector that the JCS hosts must implement at least one (and possibly all, for example, like the JDBC connector) of the following processing styles defined by the JCS framework:

- **Attribute Style Processor –** Maps LDAP attributes to endpoint attributes, usually through JCS framework support driven by datamodel metadata.
- **Method Style Processor –** Maps LDAP operations to PRE/OP/POST methods invoked on the endpoint system (for example, stored procedure support in JDBC connector). The opbindings metadata drives this.
- **Scripting Style Processor –** Maps LDAP operations and attribute into scripted output which is then submitted to the endpoint system for processing. The opbindings metadata drives this.

Note: For more information about Style Processors, see the *Programming Guide for Java Connector Server.*

Connector Implementation

You implement connectors using a combination of the following:

- Core connector code targeting an endpoint type technology.
- Validator and converter plug-in code which can be wired into your implementation using XML configuration.
- User maintained XML metadata, managed with Connector Xpress, drives configured plug-ins.
- Reusing existing connector implementations through inheritance, and only coding the minimal specializations required.
- Specializing existing connector behavior by using method-style or script-style opbindings, which are run before, instead of, or after targeted LDAP operations on specified target object classes.

Chapter 2: Installing JCS

This section contains the following topics:

System Requirements

JCS requires CA Identity Manager r12.5 SP2 or later.

Note: You do not need to install the JCS or the JCS SDK on the same computer as the Provisioning Server or CA Identity Manager Server.

Java connectors require that the following software products are installed for the Provisioning Server:

- Oracle Connector (required for Oracle Java Connector only)
- OS/400 Connector (required for OS/400 Java Connector only)
- MS SQL Connector (required for MS SQL Java Connector only)

  Note: This software is required for their Provisioning Server (parser tables) and Provisioning Manager (GUI) components.

JCS SDK requires the following software products:

- Sun J2SE Development Kit 6.0 Update 10 or later
- Apache Ant 1.7.0

Important! We recommend that you disable all antivirus software before installing JCS, JCS SDK, and Connector Xpress. If anti-virus software is enabled while installation processes are taking place problems can occur. Remember to reenable your antivirus protection after you complete installation.

Required Privileges

Administrative or root privileges are required to run the JCS installer.

Any user can run the JCS SDK installer, or the Connector Xpress installer.

UNIX Considerations

We recommend that you consider carefully the ulimit -n setting for the user for which you install the JCS. The default setting is too low to allow the JCS to function properly under load.

When this problem occurs the Java virtual machine shuts down and the following message appears in the jcs_daily log:

```
exiting because of 120 exceptions in a row: Too many open files
```

A minimum ulimit -n setting of around 80 is required for the JCS to start.

We recommend that you set the ulimit -n to at least 50 + 2 x ("configuration.maxThreads" in conf/override/server_jcs.xproperties[defaults=200]). For example, 450.

Time Zone Considerations

To achieve consistent results when setting and querying times (some stored in the Provisioning Server and some on endpoints referred to by the JCS), we recommended that you install all components on computers configured to use the same time zone.

The JCS is neutral regarding all time values passed in and out of it, and does not perform time zone transformations on them. The DYN Provisioning Manager Plug-in, which deals with dynamic endpoint types, accepts times in the local time zone and converts them to UTC. The plug-in passes them on to the JCS, and does the opposite transformation from UTC to the local time zone for query results.

Installation Components

The installation components include:

- JCS

  The JCS can be expanded through the static samples package (ZIP or TAR). The package contains binary versions of some additional connectors, including all the SDK samples.

- JCS SDK
- Connector Xpress

To see a list of the connectors included with the JCS, download the Platform Support Matrix, then find the SUPPORTED CONNECTOR ENDPOINT TYPES table near the end of the document.

Provisioning Server Registration

We recommend that you always register the JCS with the Provisioning Server.

Registering tells the Provisioning Server to use the JCS being installed to manage all the static connectors that have been deployed to it. If you want a different JCS to manage a specific static or dynamic connector, you can use Connector Xpress to specify the JCS you want to manage the connector.

It is also possible to use Connector Xpress create new namespaces in a Provisioning Server, where the connector has already been deployed to a JCS. Either use bundled template files, or where they are not available, create a project by importing the connector's metadata. When the metadata is available, deploy a new namespace.

Note: For more information see the *Connector Xpress Guide*.

Install the JCS

To manage the hosting, routing to, and management of Java connectors, install the JCS.

Follow these steps:

1. Locate the CA Identity Manager installation download or other media.
2. Extract the product files (ZIP or TAR).
3. Navigate to the following subfolder and double-click the setup file.

   `Provisioning\ConnectorServer`
4. When prompted, complete the following fields on the Provisioning Server Details screen to register the installation with a Provisioning Server.

   Domain

   Defines the Provisioning Manager domain.

   Server Host

   Defines the Provisioning Server.

   Server Port

   Defines the port on which the Provisioning Server runs.

   Username

   Specifies the Provisioning Manager administrator.

   Password

   Defines the Provisioning Manager administrator password.
5. When prompted, complete the following fields on the Connector Server (Java) Configuration screen.

   LDAP Port

   Defines the port JCS listens to.

   Production JCS port: 20410

   Development JCS port: 20412

   LDAPS Port

   Defines the secure port JCS listens to.

   Production JCS secure port: 20411

   Development JCS secure port: 20413

   Component Password

   Defines the password used to authenticate to this JCS.

   Limits: The minimum password length for the JCS password is six characters.

6. Click Next.
7. When prompted, complete the following fields on the HTTP Proxy screen, to set up proxy details that can be used by connectors:

   Enable global HTTP proxy settings?

   Enables the following settings. If you do not select this, connectors will not be able to use an HTTP proxy. You cannot set different HTTP proxy details for each connector; you must set them here.

   Note: If your organization has a direct connection to the internet, we recommend that you disable the proxy settings by selecting this check box.

   Host

   Specifies the name of the HTTP proxy server that you want to use to connect to endpoints.

   Port

   Specifies the port on which JCS can access the HTTP proxy.

   Domain

   Specifies the domain of the HTTP proxy.

   Username

   Specifies the user name you want to use to log in to the proxy server.

   Note: We recommend that you specify a user name and password if your organization's proxy server requires authentication.

   Password

   Specifies the domain password for the HTTP proxy.
8. When prompted, select whether you want to activate FIPS 140-2 Compliance Mode.
9. Click Next.

   The wizard installs the JCS.

Using an HTTP Proxy

During the JCS installation, you can specify details of an HTTP proxy. These details are then used by the following connectors:

- Google Apps
- Salesforce

When you create a Google Apps or Salesforce endpoint, you have the option of enabling the use of this HTTP proxy, or not using a proxy. You cannot specify a different proxy.

If you want to change your HTTP proxy details, you can run the JCS installer again, and enter the new proxy details.

Install the JCS SDK

To learn how to write JCS connectors by looking at worked examples which are companions to the JCS Programming Guide, install the JCS SDK.

Follow these steps:

1. Locate the CA Identity Manager installation download or other media.
2. Extract the product files (ZIP or TAR).
3. Navigate to the following subfolder and double-click the setup file:

   `Provisioning/ConnectorServerSDK`
4. Follow the onscreen instructions to complete the installation.

Note: You do not need to install the JCS SDK on the same computer as the JCS.

Note: For more information about the contents of the JCS SDK image, see the following:

`c:\program files\CA\Identity Manager\Connector Server SDK\Readme.txt`

How You Install the JCS or Connector Xpress Silently

The JCS and Connector Xpress both support a silent mode of installation. To run a silent install create a response file. The process for the JCS and Connector Xpress is the same:

1. Do *either* of the following:
   - Enter the following command, and then follow the instructions in the file to update the parameters in the file.

     `setup –options-template template_file`
   - Enter the following command and run through the installation:

     `setup -options-record response file.`

     The response file records the input parameters.
2. Start the silent installation using one of the previously prepared response files.

   Note: Use fully qualified path names when generating and running with response files. For example, *responsefile.txt* is not valid but *C:\responsefile.txt* is valid.

Install the Connector Samples

Important! We recommend that you use the sample connectors only in a test environment. We do not support the sample connectors.

To install connector samples, or upgrade an existing install to include the sample connectors, extract both the installer for your operating environment and the samples archive to the same folder before performing an install.

The samples are installed and registered.

Note: You cannot uninstall the connector samples.

Connector Uninstall

We do not support uninstallation of any connectors (including the samples) for this release.

Manual Activation of JDBC Vendor Support

Manual activation of JDBC connection for several vendors is required, as it is not possible to bundle various third-party drivers and licenses, or both, with the JCS, or because additional specific manual configuration is required.

More information:

How to Activate JDBC DB2 for Z/Os Vendor Support (see page 22)
How to Activate Microsoft SQL Native Authentication on Windows (see page 23)
How to Activate JDBC Sybase Vendor Support (see page 24)

How to Activate JDBC DB2 for Z/Os Vendor Support

To activate JDBC connection to DB2 for Z/Os, you need do download a driver and license for DB2 Connect.

Follow these steps:

1. Download the file db2_v9_db2driver_for_jdbc_sqlj.zip from the drivers section of:

   http://www-306.ibm.com/software/data/db2/java/

   Note: You need an IBM Registration to download the files.
2. Copy the files db2jcc.jar and db2jcc_license_cisuz.jar into the following locations:

   jcs_dir/extlib/

   conxp_dir/lib/
3. Stop and restart the JCS service.
4. Stop and restart all Connector Xpress sessions.

How to Activate Microsoft SQL Native Authentication on Windows

Microsoft SQL Native Authentication on Windows can only be activated when both Connector Xpress and the JCS are running on Windows operating systems. The required library sqljdbc_auth.dll is bundled with the Java CS.

The user must run Connector Xpress on the same domain as the Microsoft SQL server, and the Microsoft SQL server must be configured to allow that user to access the appropriate database instances.

Follow these steps:

1. Update the Java CS service to run as the required Windows user and restart the service.
2. By default the Java CS service is set to run as the local SYSTEM user, however if you are using trusted authentication you must run the service as a domain user. Do the following:
   a. Click Start, Control Panel, Administrative Tools, Services.

      The Services window appears.

   b. Right-click CA Identity Manager-Java Connector Server, then click Properties.

      The Properties dialog appears.

   c. Select the This account check box, and then complete the details of the domain user under which you want to run the service.
3. Stop and restart the JCS service.
4. When you set up Microsoft SQL datasource in Connector Xpress, select the Native check box on the Edit Sources dialog.

   Connector Xpress adds the following to the JDBC URL used for the connection:

   integratedSecurity=true

   Note: For more information about configuring data sources, see the *Connector Xpress Guide*.

How to Activate JDBC Sybase Vendor Support

A JDBC connection to Sybase requires a license for Sybase. To activate JDBC Sybase vendor support, do the following:

Follow these steps:

1. Download the 6.0.5 driver .zip file from http://www.sybase.com, or from your Sybase product media.
2. Extract the following file:

   jConnect-6_0\classes\jconn3.jar

   from the following archive:

   jConnect-6_05.zip
3. Copy the file into the following two locations:

   jcs_dir/extlib/

   conxp_dir/lib/
4. Shutdown and restart the JCS service.
5. Shutdown and restart all Connector Xpress sessions.

Install Connector Xpress

You do not need to install Connector Xpress on the same computer as Java CS or any other server component, including the Provisioning Server.

Important! We recommend that you disable all antivirus software before installing Java CS, Java CS SDK, and Connector Xpress.

Follow these steps:

1. Locate the CA Identity Manager installation download or other media.
2. Extract the product files (ZIP or TAR).
3. Navigate to the following subfolder and double-click the setup file.
   `Provisioning\ConnectorXpress`
4. Follow the onscreen instructions to complete the installation.

File Locations

The default Windows and UNIX directories are listed in the following table. Your actual installation directories depend on your operating system and selections during the installation process.

Path Notation	Default Directory	
	Windows	**UNIX**
im-home	C:\Program Files\CA\Identity Manager	/opt/CA/IdentityManager
imps-home	C:\Program Files\CA\Identity Manager\Provisioning Server	/opt/CA/IdentityManager/Provisioning Server
jcs-home	C:\Program Files\CA\Identity Manager\Connector Server	/opt/CA/IdentityManager/ConnectorServer
jcs-sdk-home	*jcs-home*/build/dist	*jcs-home*/build/dist
conxp-home	C:\Program Files\CA\Identity Manager\Connector Xpress	/opt/CA/IdentityManager/ConnectorXpress
sample-home	C:\Program Files\CA\Identity Manager\Connector Server SDK\connectors\sdk	/opt/CA/IdentityManager/ConnectorServerSDK/connectors/sdk

Start and Stop the JCS

You can start the JCS using one of the following methods.

- Windows Service (see page 25)
- UNIX Daemon (see page 26)
- Windows command line (see page 26)

Windows Service

You can start, stop, and configure the JCS from the Windows Services application.

The JCS installation process creates a Windows service to run the JCS. This service is named *CA Identity Manager -Connector Server*.

Start the JCS Service from the Command Prompt Window

To start the JCS service from the Command Prompt window, use the following command:

```
net start im_jcs
```

Stop the JCS Service from the Command Prompt Window

To stop the JCS service from the Command Prompt window, enter the following command:

```
net stop im_jcs
```

Start the JCS Using the UNIX Daemon

The JCS installation process creates a startup script named *-jcs* and links it to the rc.d system on the local system. The script responds to standard start, stop, restart and status requests, and automatically runs the JCS in run levels 2-5, or shuts it down on 0,1, and 6 corresponding to system halt, single user mode, and reboot.

/etc/init.d/im_jcs start

Starts the JCS daemon

/etc/init.D/im_jcs restart

Restarts the JCS daemon

/etc/init.d/im_jcs stop

Stops the JCS daemon

/etc/init.d/im_jcs status

Displays the status of the JCS daemon

Run the JCS from the Command Line

You can run JCS interactively using the service launcher to diagnose problems.

```
cd jcs-home\bin
JCS.exe //TS//im_jcs
```

You can also use Apache procrun arguments to start, stop, disable, remove, or configure the service.

For a comprehensive list of command-line arguments, see the Procrun section in the Jakarta Commons section:

http://jakarta.apache.org

Chapter 3: Deploying Connectors

This section contains the following topics:

How you Deploy a Connector

The following describes two ways to deploy a connector:

- Use the Connector Xpress and connector metadata to map the generic DYN schema to an endpoint system. The metadata for the connector information is sent to a running JCS through the Provisioning Server.

  Note: A JCS restart is not required for the connector to become active.

- Deploy a custom Java connector to the JCS by copying the connectors jcs-connector-*.jar file and any additional .jar files it depends on to the *jcs-home*/lib/ directory, and then restart the JCS. Use the Provisioning Manager to acquire endpoints managed by this connector.

  Note: See the *Programming Guide for Java Connector Server* for more information.

When the JCS is used with the Provisioning Server, the latter acts as the persistent store for connector metadata, therefore the JCS is stateless.

Connector Deployment

You can use Connector Xpress to configure metadata settings and to deploy the connector. Connector Xpress lets you save metadata settings to a project file and then deploy a connector from that file. For example, you can use a project file when you move connectors from a test environment to a production environment to ensure that you have the same settings.

Connector Xpress Wizard

You can create a dynamic connector to provision accounts and groups using either the Connector Xpress wizard to map database tables (JDBC) or to map object classes within a directory's schema (JNDI).

Note: For more information, see the *Connector Xpress Guide.*

Consider the following guidelines when mapping database tables:

- (JDBC) Because space padding can require additional exploration, try using VARCHAR rather than CHAR types where possible for columns mapped to naming attributes.
- (JDBC and JNDI) If using a default value for Boolean column, ensure the default value is a valid Boolean.

Chapter 4: Configuring JCS

This section contains the following topics:

Server Configuration File

The server_jcs.xml file contains configuration attributes for both the JCS and the Apache DS directory which acts as its host. The server_jcs.xml file also includes settings that affect connector behavior.

The server_jcs.xml file uses Spring Framework XML support, which allows Java classes adhering to JavaBean standards (for example, property getters and setters) to be created and initialized using XML tags.

The default file path is:

jcs_home\conf\server_jcs.xml

Note: The installer overrides the server_jcs.xml file during upgrades. If you make any manual customizations, we recommend that you use the matching property settings in the *jcs_home*\conf\override\server_jcs.properties file, as the installer does not overwrite this file.

You can find examples of the settings which you can manually set in the file *jcs_home*\conf\override\SAMPLE.server_jcs.properties. You can also override most other settings by using property names matching the nested structure of the entries in server_jcs.xml, as shown in the SAMPLE.server_jcs.properties file.

server_jcs.xml file—Configure Attributes

The server_jcs.xml file contains the following configuration settings:

java.naming.security.authentication

Specifies the authentication methods. Only simple is currently supported.

java.naming.security.principal

Specifies the authentication principal. This is hardwired to uid=admin,ou=system by ApacheDS, but an optional java.naming.security.principal.alias= can be specified to ease integration. When this alias is received for authentication, it is treated exactly as uid=admin,ou=system.

java.naming.security.credentials

Specifies the authentication credentials for the configured principal. The authentication credentials can be stored in the file as plain text or as a SHA one-way hash.

We recommend that you store them as a SHA one-way hash, rather than as plain text.

We recommend that you do not change this password except through the installer, as the value specified in this file must match the value inside ApacheDS persistent store.

maxThreads

Specifies the maximum number of requests that can be processed concurrently for all activated connectors hosted by a JCS. It defaults to 200 to match the Provisioning Servers configuration. When increasing it be sure to consider also increasing other configuration settings like heap-space for the Java Virtual Machine or “ulimit –n” setting for open files on UNIX.

Note: The manual settings in the server_jcs.properties file can potentially override this setting.

Note: For more information, see UNIX Considerations (see page 16).

ldapPort

Specifies the port which the JCS listens on for insecure connections. Set the port to one of the recommended ports unless multiple C++ Connector Servers or JCSs run on the same computer. Where a secure port is configured, use the secure port instead.

The insecure port can be useful for debugging purposes. By default, the JCS installer mandates the use of the ldapsPort exclusively. Set the port to one of the following port numbers:

- Production JCS: 20410
- Development JCS: 20412

ldapsPort

Specifies the port which the JCS listens on for secure connections. The ldapsPort, with associated properties enableLdaps, ldapsCertificateFileldapsCertificateFile, and ldapsCertificatePassword, must be a different port from the one chosen for ldapPort. Traffic on this port is secured using the configured certificate and the Transport Layer Security (TLS) protocol. The ldapsPort can also be useful for debugging by setting the logging level in the JCS log4j.properties file to trace LDAP requests as they are delivered to the JCS.

Set the port to one of the following port numbers:

- Production JCS secure port: 20411
- Development JCS secure port: 20413

The ldapsCertificateFile is configured to reference a Java keystore containing the standard IM Provisioning Server certificate. The JCS installer sets the default ldapsCertificatePassword.

bootstrapSchemas

Specifies which LDAP schemas the JCS knows. This property incorporates schemas which have been converted to Java objects by the ApacheDS build process. Additional OpenLDAP formatted .schema files (see http://www.openldap.org/doc/admin23/schema.html) can be loaded by placing them in the Java CS conf/ directory (like eta_dyn_openldap.schema) or ideally contributed from the conf/ directory within a specific connector's JCS-connector-*.jar file (refer to SDK connector's conf/etaeta_sdk_openldap.schema _nds_openldap.schema registered through its conf/connector.xml descriptor in the jcs-connector-sdk.jar sample connector).

partitionLoaderService

Configures the service that detects LDAP traffic and determines when it is JCS related. This service handles lazy activation of connectors as they are mentioned in LDAP ADD requests.

interceptorConfigurations

Specifies any other standard ApacheDS interceptor services. Interceptor services not required by the JCS have been deactivated.

connectorPersister

Specifies the Persister for connector and connector levels of the DIT. The Persister is not required when the JCS is accessed through the Provisioning Manager, but can be of interest in other deployment situations.

cryptoService

Users can configure the crypto service users who want to activate encryption convertors on specific fields according to their metadata properties, most importantly the isEncrypted boolean metadata setting.

Validators and Converters

The server_jcs.xml file contains these configuration attributes for validators and converters:

- The global set of validators that is available for reuse in all contained connectors (see the validatorManager JavaBean for input reference.)
- The global set of converters that is available for reuse in all contained connectors (see the converterManager JavaBean for input reference)

Note: Any JCS wide changes made to these attributes in the server_jcs.xml file are lost when upgrading, so achieve the same effect through the same attributes in individual connectors' connector.xml files instead.

Note: For more information about validators and converters, see the *Programming Guide for Java Connector Server.*

Connector Configuration File

The server_jcs.xml file contains configuration items that cover the whole JCS, whereas each connector has a conf/connector.xml file which covers a single connector. Some sections like validator and converter configuration exist in both files, in which the connector.xml can add to, or override, values in server_jcs.xml.

Each connector's jar file has a connector.xml file in it. To allow easier customization of its values after deployment, a matching file under *jcs_home*/conf/override/*connector*/connector.xml can be used to override settings.Template files with the name SAMPLE.connector.xml are provided in each directory. To customize, rename the templates to connector.xml, edited as required, and then restart the JCS.

For an example of a connector.xml template file see, *jcs_home*/conf/override/jdbc/SAMPLE.connector.xml.

The following sections describe the important sections in the conf/connector.xml file.

server_jcs.xml — Initial TLS Settings

The server_jcs.xml file has the following initial TLS settings:

Note: Manual settings in the server_jcs.properties file can potentially override all these settings.

ldapsCertificateFile

Specifies the JCS LDAPS certificate store. Specifies a path to the file which contains all the certificates used to verify the identity of the JCS server during inbound LDAPS (TLS) connections. At least one certificate with an accompanying private key issued to represent JCS is placed in this store.

ldapsCertificatePassword

Specifies the password protecting the JCS certificate store specified in ldapsCertificateFile.

Note: The password can either be cleartext or obfuscated. For example:

```
{ALGORITHM}ciphertext where ALGORITHM would be typically set to
'CACRYPT' . For example, {AES}LQpBXeIjOMGSsGLU
```

connectorClientCertStore

Specifies the JCS wide client certificate store. Specifies a path to the file which contains trusted certificates used to verify the identity of the endpoint server during SSL handshakes. Used for outbound TLS connections made by the connectors themselves, to the endpoint systems they manage. Import any issuer certificates for the endpoints to which TLS connections into this store.

connectorClientCertStoreType

Specifies the certificate store type (JKS or PKCS12).

connectorClientCertStorePassword

Specifies the password protecting the connector client store. The same rules apply as for the ldapsCertificatePassword.

connectorSSLVerifyPeer

If false, specifies that during SSL handshakes, the peer certificate sent by the endpoint to which a connection is made, is not verified for trust. That is, the connectorClientCertStore value is ignored and not required for outbound SSL connections in this configuration. If true, the endpoint host certificate presented to JCS undergoes trust checks against connectorClientCertStore contents.

Default: False

connectorSSLTrace

Set to true if verbose SSL handshake information is output to log.

Connector Pool Configuration

Most connectors use a connection pool configured in connector.xml, for example, through:

- poolConfig for JNDI and most connectors()

  Note: For more information, see the Class GenericObjectPool on http://jakarta.apache.org

- dataSourceConfigProps for JDBC

  Note: For more information, see http://jakarta.apache.org for a complete list and documentation of available configuration parameters.

Change Pool Related Settings

To maximize scalability for a connector by configuring it to match expected usage patterns, you can change pool related settings.

To change pool related settings

1. Copy *jcs-home*/conf/override/jdbc/SAMPLE.connector.xml to connector.xml.
2. Edit the connector.xml file.
3. Restart the JCS.

Retry Configuration

You can configure the Exception Map setting to contain groups of exception messages that require special handling (and optionally associated retry delay and retry count settings).

In particular, the JDBC connector defines entries for exceptions signifying these conditions which drive retrying when connections to the endpoint experience problems:

- **Stale**—The connection to the endpoint has become stale and is reestablished immediately.
- **Retriable**—The connection to the endpoint has encountered what can be a transient soft failure, in which case a retry loop is started with the configured count and delay. If the count is exhausted before connectivity is restored, then the current request is considered to have suffered a hard failure which is reported to the JCS client.
- **Busy**—The endpoint has reported it is too busy to complete a request in which case a retry loop is started with a separate retry delay and count settings. For example, the MSSQL and Ingres databases both report deadlock exceptions when they are unable to complete processing of a transaction within a certain time interval. The delay and recount settings are typically much longer than the Retriable case.

In addition to these triggering exceptions, each ExceptionRetryGroup has associated resilientDelay and resilientMaxRetries settings which specify how many retry attempts are required when a matching exception is encountered, and the delay between each attempt.

Edit JVM Memory Options on Windows

To edit the JVM memory options JvmMs, JvmMx, JvmSs and Classpath use the service update command or edit the following registry key on Windows:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\Identity
Manager\Procrun 2.0\im_jcs\Parameters\Java
```

Note: You can use Apache procrun arguments to update the service parameters. For more information, see Procrun service application at http://jakarta.apache.org

Java Virtual Machine Out-of-memory Errors

Symptom:

I receive JVM out-of-memory errors during stress or high load periods that affect the functionality of the JCS.

Solution:

We recommend that you set JVM debugging options so that you are alerted when out-of-memory conditions occur.

Note: For more information about setting JVM debugging options, see Debugging Options in Java HotSpot VM Options at http://www.oracle.com.

Edit JVM Memory Options on UNIX

To edit JVM options on UNIX, create a file named jvm_options.conf in the data folder with the following Java arguments:

```
-Xms128M -Xmx192M -d64
```

-Xms

Specifies the minimum heap memory allowed for the JCS

Example: -Xms128M, specifies the minimum heap memory allowed for the JCS is 128 MB.

-Xmx

Specifies the maximum heap memory allowed for the JCS.

Example: -Xmx192M, specifies the maximum heap memory allowed for the JCS is 192 MB.

-d64

Specifies that the JVM is run in a 64 bit environment.

Note: For more information, see the documentation for the Java command tool at http://java.sun.com

JXplorer to JCS Connection Parameters

If you have a standard installation, you can use the following parameters to connect to JCS from an LDAP browser such as JXplorer.

Note: For more information on JXplorer, see http://www.jxplorer.org.

These settings are configured in server_jcs.xml. Changing the User DN is problematic because of assumptions within ApacheDS. Consequently, the property *java.naming.security.principal.alias* has been added to simulate use of a different user DN (this is an alias to "uid=admin,ou=system").

Host

Specifies the host server name of the JCS

Protocol

LDAP v3

Port

2041, when using level: SSL + User + Password (TLS)

20410, when using the less safe level: User + Password

User DN

uid=admin,ou=system

Password

As configured during installation.

Change the JCS Administration Stored Password

To ensure better security across a deployment you can change the JCS administration users stored password.

Note: The JCS remembers all passwords for the administrator user because the last restart and all passwords are accepted as valid for bind requests. This allows phased upgrade of passwords for a JCS where multiple Provisioning Servers connect to the JCS, but do not update their stored passwords for the JCS at the same time.

Follow these steps:

1. Start JCS.
2. Bind to JCS using the following details:

   Host

   Specifies the host computer name of the JCS

   Protocol

   LDAP v3

   Port

   2041, when using level: SSL + User + Password (TLS)

   20410, when using the less safe level: User + Password

   User DN

   uid=admin,ou=system

   Password

   As configured during installation.
3. Change the password for “uid=admin,ou=system” to a new value.

   This change takes effect immediately.

Set the TLS Store Certificate Password

When LDAP clients request TLS secured connections to the JCS, you can configure the password used on the Java keystore. We recommend that you configure the password to overwrite the temporary cached password for this keystore when freshly installed.

Note: If you want to install your own certificate instead of the default Provisioning Manager certificate configured by the installer, you can also manage the keystore using the keytool utility included in the Java Runtime Environment

Follow these steps:

1. Shut down the JCS.
2. On Windows, open a Command Prompt window, then enter the following command:

   `cd $jcs_home/_uninst/_jvm/bin`

   Windows changes the directory to the JVM's bin folder.
3. Enter the following command:

   `cd jcs_home/bin`
4. Do *one* of the following
   a. Run the following command:

      `ldaps_password new-password../conf/override/server jcs.properties`

      The encrypted ldapsCertificatePassword value in server_jcs.properties is updated.

   b. Run the following command:

      `ldaps_password new-password ../conf/override/server_jcs.properties connectorManager.connectorClientCertStorePassword`

      The encrypted connectorClientCertStorePassword value in server_jcs.properties is updated.
5. Restart the JCS.

   Note: The password for the keystore is the password that you set during the JCS installation.

Adjust JCS Service Start Parameters

To adjust any JCS service start (including related JVM parameters), go to the following location in the Windows registry:

```
HKEY_LOCAL_MACHINE\ComputerAssociates\Identity Manager\Procrun
2.0\JCS\Parameters
```

Connection Pooling

Connection pooling is configured through the connector.xml file for an individual connector (for example, *jcs-home*/conf/override/jdbc/connector.xml, copied from SAMPLE.connector.xml in the same directory) file, rather than in the server_jcs.xml global configuration file.

Determine Java CS Version Number

To determine the version number of your JCS installation, run the following at the command line:

```
cd jcs_home/lib
java -jar jcs.jar
```

The JCS version number is printed. For example:

```
CA Identity Manager - Java Connector Server
Version 12.5..nnnnn
```

Where Connector Xpress Stores User Preferences

Connector Xpress stores user preferences through the Java Preferences API.

On Windows, this stores data in the following registry key:

```
HKEY_CURRENT_USER\Software\JavaSoft\Prefs
```

On UNIX, this stores data in the user's home directory under the .java/.userPrefs folder.

Note: The uninstaller does not remove this data, so subsequent installs preserve user preferences.

Chapter 5: JCS Logging

This section contains the following topics:

Logging Configuration

Log files are critical aids in troubleshooting problems with the JCS and hosted connectors. We recommend that you start with jcs_daily* files and work downwards to connector-specific log files as required.

How Logging is Managed

Both the JCS and ApacheDS code bases use the NLOG4J library to manage logging, which is configured through the following file:

jcs-home\conf\log4j.properties

You should rarely (if ever) need to change the log levels of any ApacheDS classes. These classes perform fundamental functions such as parsing and formatting LDAP messages.

Note: This file contains a number of potentially interesting settings commented out, with a description of the circumstances under which they can be useful.

You can also use a file named log4j_verbose.properties in the same directory. When you raise an issue, copy log4j_verbose.properties to log4j.properties, restart the JCS, cause the issue to occur, shutdown the JCS, then zip up the entire the jcs-home/logs/ directory.

Connector Specific Logging

You can configure the logging level for connector-specific logs using the following attributes:

- eTLog=1 (active)
- eTLogDestination='F' (file)
- eTLogFileSeverity

Turn on Connector Instance Logging

You can specify that each destination receives only messages of a certain severity level or levels, such as error, warning, or fatal messages.

Follow these steps:

1. In Provisioning Manager, click the Endpoint Types button.
2. Select the Endpoint Type from the Object Type list.
3. Click Search.

   The Endpoints under the Endpoint Types are displayed.
4. Right click the Directory from the DirectoryNameCommon column, then click Properties.

   The Global Properties sheet appears.
5. Click the Logging tab.

   The Logging tab appears.
6. Select the Enabled check box.

   Logging is enabled for the Provisioning Manager.

   Note: If you do not select this check box, the Provisioning Manager does not keep any log of its actions, except for any information that is available in the message window.
7. Enable logging for Provisioning Manager, and select to log all message types.

Logging Severities

The severities are mapped as follows:

IMPS Severity	IMPS Severity Code	Log4J Level
Information	'I'	DEBUG
Non-Admin Success	'S'	INFO
Warning	'W'	WARN
Error	'E'	ERROR
Fatal	'F'	FATAL

Note: Clients other than the Provisioning Manager can configure an attribute used to set the logging level in server_jcs.xml through the logLevelAttr attribute of the configured MetaConnectorFactory, in which case the Log4J Levels listed in the table above are the possible values.

Setting the JCS Logging Level

To set the JCS systemwide logging level, set the line of the form log4j.logger.com.ca=DEBUG, then restart the JCS, or use the log4j_verbose.properties file instead. This setting controls the logging level for the JCS framework.

Note: The file log4j.properties has properties that determine whether log files are appended daily, and the format of the lines output.

Note: For information about the settings in the file, see the comments in the file.

The endpoint-type/jcs_conn_connector-name.log file generated for each active connector instance contains most of the logging data related to its corresponding connector. However, also look for relevant logging in the jcs_daily.log* systemwide log file. Some examples of messages that are logged to the systemwide file include:

- Connectors that use third-party libraries
- Connectors developed without sufficient attention to logging
- Problems that occur while creating or activating a connector

Files in the Log Directory

The following table lists the log files which are written to the log directory *jcs-home*\logs

Log File Name	Description
jcs_daily.log	Today's ApacheDS and Java CSJCS logging
jcs_daily.log.*YYYYMMDD*	ApacheDS and JCS logging for date
endpoint-type/jcs_conn_*connector-name*.log	Specific logging output for named connector
endpoint-type/jcs_conn_*connector-name*.log.*YYYYMMDD*	Specific logging output for named connector for date

Connector Xpress Logging

Connector Xpress log messages are written to this file:

im_home\Connector Xpress\logs\conxp-log.txt

Note: The generated metadata is usually the primary resource about the problem being diagnosed. We recommend that you analyze the generated metadata using the Connector Xpress log files.

Turn on Standalone JCS Logging

You can turn on Standalone JCS endpoint specific logging and set the log4j severity level.

Follow these steps:

1. Open the *jcs-home* \conf\log4j.properties file.
2. Change the log4j severity level in the following line:

   log4j.category.jcs_conn=OFF, jcs_conn_Appender

   For example, log4j.category.jcs_conn=DEBUG, jcs_conn_Appender changes the severity level to debug.

   Note: For more information about log4j severity levels, see, Logging Severities (see page 44).
3. Restart the Standalone JCS.

Provisioning Manager Logging

The following log files have relevant modification times that show what is happening on the Provisioning Manager or Provisioning Server side:

- *imps-home*\Logs\etatrans*.log
- *imps-home*\Logs\satrans*.log

Note: To view, compress or archive any log files for the current day, make a copy of the log file, as the Provisioning Manager continues to write to the log files. Save the log file copies using the original file name, but in a different directory.

These logs are also relevant to JCS problems and to problems deploying new connectors from Connector Xpress.

Provisioning services-related installer log messages are written to:

```
%temp%\imps_*_install.log
```

where %temp% is the temp directory of the current user.

Provisioning Manager logs appear in files matching "*imps-home*\Logs\etayyyymmdd.log

Turn on Provisioning Server Logging

You can specify that the Provisioning Server logs only messages of a certain severity level or levels, such as error, warning, or fatal messages.

Follow these steps:

1. In the Provisioning Manager, click System, Global Properties.

   The Global Properties dialog appears.

2. Click the Logging tab.

   The Logging tab appears.

3. Select the Enabled check box.

   Logging is enabled for the Provisioning Server.

   Note: If you do not select this check box, the Provisioning Manager does not keep any log of its actions, except for any information that is available in the message window.

4. Enable logging for the Provisioning Manager, and select to log all message types.

Turn on Provisioning Manager Logging

You can specify that the Provisioning Manager logs only messages of a certain severity level or levels, such as error, warning, or fatal messages.

Follow these steps:

1. In Provisioning Manager, click File, Preferences.

   The Preferences dialog appears.

2. Click the Logging tab.

   The Logging tab appears.

3. Enable logging for Provisioning Manager and select to log all message types.

   Provisioning Manager logging is enabled.

Glossary

Apache Ant

Apache Ant is a software tool for automating software build processes. It is similar to make but is written in Java, and is best suited to building Java projects.

Apache Directory Server (ApacheDS)

The *Apache Directory Server (ApacheDS)* is an embeddable LDAP server written in Java.

C++ Connector Server

The *C++ Connector Server* is a connector server that manages C++ connectors. It can be installed on the Provisioning Server or on a remote system. The C++ Connector Server provides an object-oriented application framework that simplifies development of connectors, which are responsible for communication between the C++ Connector Server and the endpoint.

connector

A *connector* is the software that enables communication between CA Identity Manager and an endpoint system. You can generate a dynamic connector using Connector Xpress, and you can develop a custom static connector in Java.

Connector Xpress

Connector Xpress is a utility for managing dynamic connectors, mapping dynamic connectors to endpoints, and establishing routing rules for endpoints.

converter

A *converter* is a Java class used by the Java CS infrastructure for converting data to and from endpoint-specific formats.

deployment

Deployment describes two different actions, saving metadata from Connector Xpress to an endpoint in the Provisioning Server or making a new connector implementation available to the Java CS by placing its jcs-connector-*.jar file into the <jcs_home>/lib directory.

directory information tree (DIT)

A *directory information tree* (DIT) is a tree of objects presented to clients by an LDAP server in which the clients can directly name and manipulate those objects by issuing LDAP commands.

Distinguished Name (DN)

A *Distinguished Name (DN)* is a unique name for an entry in a directory service. In LDAP, this consists of its *Relative Distinguished Name (RDN)* constructed from some attributes in the entry, followed by the parent entry's DN. The DN is analogous to a full filename, and the RDN is analogous to a relative filename.

dynamic connector

A *dynamic connector* has metadata that is generated and maintained by end users using Connector Xpress. The metadata drives the behavior of the dynamic connector at runtime.

endpoint

An *endpoint* is a specific installation of a platform or application, such as Active Directory or Microsoft Exchange, which communicates with CA Identity Manager to synchronize information. A connector server uses a connector to manage an endpoint.

endpoint Type

An *endpoint type* is a logical grouping of endpoints. There is one connector for each endpoint type, but there can be multiple endpoint systems for each connector.

eTrust Admin Server

See *Provisioning Server*.

Java Connector Server (JCS)

The Java Connector Server (JCS) is a server component which handles hosting, routing to, and management of Java connectors. The JCS provides a Java alternative to the C++ Connector Server (CCS).

JDBC (Java DataBase Connectivity)

JDBC (Java Database Connectivity) is a Java API for executing SQL statements against relational databases.

JDK (Java Development Kit)

JDK (Java Development Kit) is a software development kit for producing Java applications.

JIAM (Java Identity and Access Management)

JIAM (Java Identity and Access Management) is a Java front end to the Provisioning Server.

JNDI (Java Naming and Directory Interface)

JNDI (Java Naming and Directory Interface) is a Java API that enables access to naming and directory services such as LDAP servers.

JVM (Java Virtual Machine)

JVM (Java Virtual Machine) is an execution environment that converts Java bytecode to machine language and executes it.

LDAP (Lightweight Directory Access Protocol)

LDAP (Lightweight Directory Access Protocol) is a software protocol for locating organizations, individuals, and other resources such as files and devices in a network, whether on the public Internet or on a corporate intranet. An LDAP directory is a simple tree hierarchy which distributed among many servers.

Log4j

Log4j is a popular Java logging library used primarily as a debugging tool, and in Java CS custom connector development.

metadata

Metadata is XML data that describes the structure of a connector to the Java CS.

namespace

A *namespace* is a specific type of endpoint system.
See *endpoint type*.

option

See *Connector.*

Provisioning Server

The *Provisioning Server* handles provisioning of users, and then delegates to connectors (using the Java CS or CCS) to manage endpoint accounts, groups, and so on.

SuperAgent

See *C++ Connector Server.*

validator

A *validator* is a Java class that checks the validity of individual values, attributes, or an entire object class. The Java CS uses validators to ensure that data values meet specific requirements before being passed to or from the endpoint system for processing.

Index