CA Identity Manager

Java Connector Server Implementation Guide

r12

Second Edition

CA Product References

This document references the following CA products:

- CA Identity Manager

Contact CA

Contact Technical Support

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

- 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

Provide Feedback

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

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is also available on the CA support website, found at http://ca.com/support.

Contents

Chapter 1: Java Connector Server

The *Java Connector Server (Java CS)* is a server component which handles hosting, routing to, and management of Java connectors. The Java CS provides a Java alternative to the C++ Connector Server. It is architecturally and functionally similar to the C++ Connector Server, except that it has a Java API instead of a C++ API, which allows your connectors to be implemented in Java. In addition, the Java CS is data-driven rather than code-driven, which allows more functionality to be addressed by the container (or Java CS) instead of by connectors themselves.

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

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 Identity Manager

Endpoint Systems

An *endpoint* is a specific target system or application, such as Active Directory or Microsoft Exchange, that is managed by the Provisioning Server.

A *connector* is the software that enables communication between a Provisioning Server and an endpoint system. 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, and translating add, modify, delete, rename, and search LDAP operations on those objects into corresponding actions against the endpoint system.

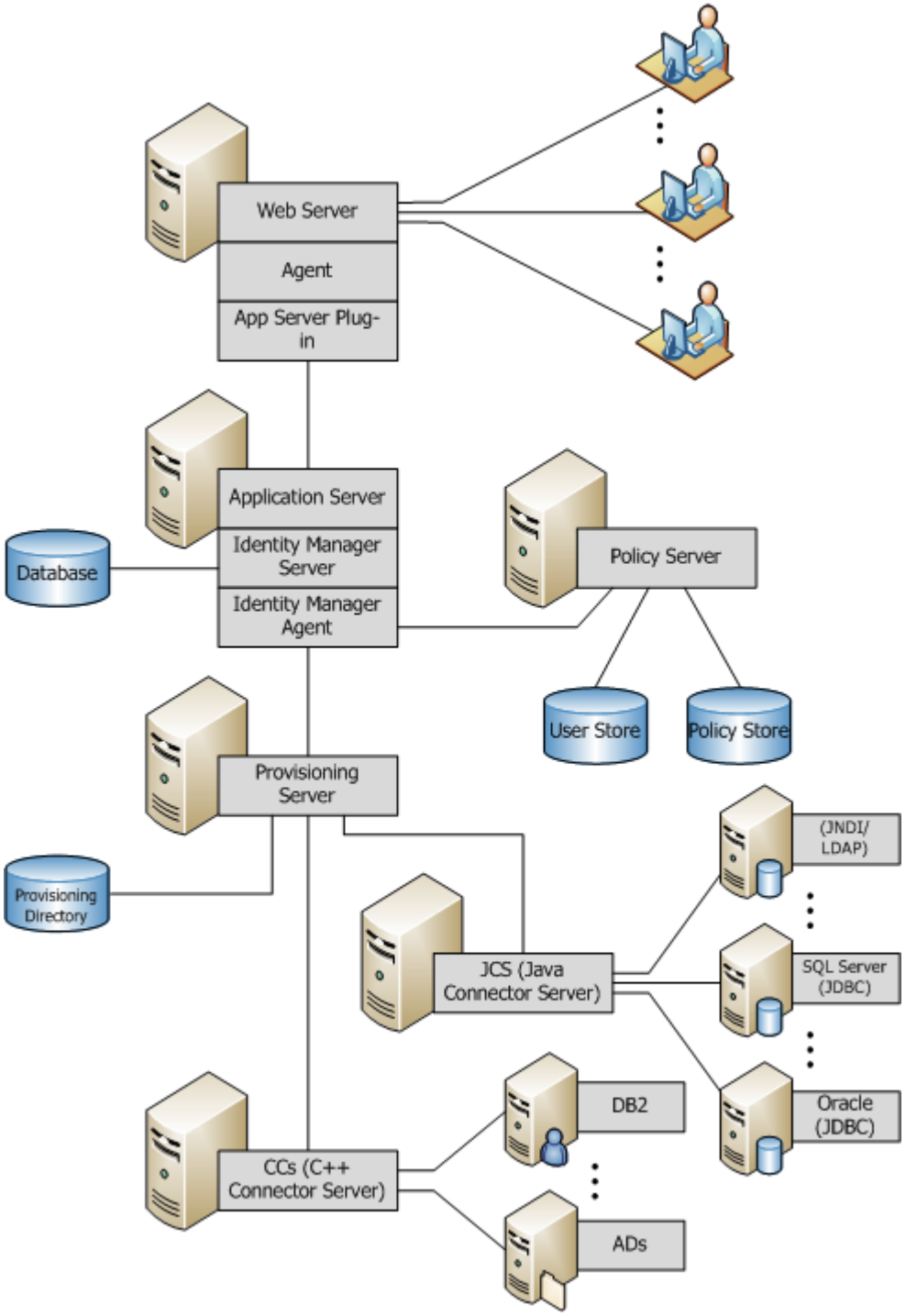
A dynamic connector has XML metadata that is generated and maintained by end-users using Connector Xpress, whereas a static connector has metadata that is generated and maintained by developers rather than end-users.

Connector Deployment

A connector's software is deployed to a Connector Server using an implementation bundles .jar files for a Java CS or as a dynamic link library or shared library for a C++ Connector Server. Connector instances are then created by the Connector Server using this software when requests are received to add a new Provisioning Directory.

Java CS Architecture

This following illustration shows the Java CS in a typical installation.



Java CS Framework

The Java CS 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 websites:

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

The Java CS 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 Java CS Does

Like the C++ Connector Server, the Java CS provides services to the connector implementation deployed to it while imposing minimal restrictions, in the same way that a J2EE application server provides for its hosted applications.

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

- 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 Java CS 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 is hosted by Java CS must implement at least one (and possibly all, like the JDBC connector) of the following processing styles defined by the Java CS framework:

- **Attribute Style Processor –** Maps LDAP attributes to endpoint attributes, usually through Java CS framework support driven by 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).
- **Scripting Style Processor –** Maps LDAP operations and attribute into scripted output which is then submitted to the endpoint system for processing.

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

Connector Implementation

Connectors are implemented using a combination of the following:

- CA-maintained implementation. The actual code is determined by the connector technology (for example, JDBC, JNDI).
- CA-maintained XML configuration (like registering plug-in validators and converters).
- End-user maintained XML metadata, managed with Connector Xpress, drives configured plug-ins.

Chapter 2: Installing Java CS

This section contains the following topics:

System Requirements

Java CS requires CA Identity Manager r12 or later.

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

Java Connectors require the following software products:

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

  Note: These are required for their Provisioning Manager components.

Java CS SDK requires the following software products:

- Sun J2SE Development Kit 5.0 Update 5
- Apache Ant 1.6.5

Important! Be sure to disable all antivirus software before installing Java CS, Java CS SDK, and Connector Xpress. Anti-virus software may cause problems if enabled while installation processes are taking place. Remember to re-enable your antivirus protection after you complete installation.

Required Privileges

Administrative or root privileges are required to run the Java Connector Server installer.

Any user may run the Java Connector Server SDK installer.

Any user may run the Connector Xpress installer.

Solaris Considerations

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

The Java Virtual Machine will shutdown 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 Java CS 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.

Timezone Considerations

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

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

JCS Upgrade Considerations

- We only support communication between an r12 Provisioning Server and an r12 Java Connector Server.
- If you have an 8.1 SDK installed and you have made any source file changes or modifications that you want to keep, we recommend that you backup your SDK directory, as the upgrade process will override any changes you have made.
- The 8.1 SDK code compiles against the r12 SDK, but some methods and classes have been deprecated. In comparison to the to the 8.1 API, there have been some minor changes in areas not directly used by the SDK example (primarily classes moving into new subpackages, impacting only Java import statements)

Note: For more information about classes that have been put in the new sub-packages in the SDK, see the *Programming Guide for Java Connector Server* and the *JCS Javadoc*.

Override server_jcs.xml Settings

The server_jcs.xml file is overwritten by the installer during upgrades. We recommend that if you have made any manual customizations you should use the matching property settings in the jcs_home\conf\override\sample.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.

To override server_jcs.xml file settings

1. Rename jcs-home/conf/override/SAMPLE.server_jcs.properties to server_jcs.properties.
2. In the new server_jcs.properties file, remove the leading # character from the property you want to override.

   For example, the property configuration.ldapsCertificatePassword overrides <property name=ldapsCertificatePassword><value>…<value></property> nested within <bean id=configuration in server_jcs.xml.
3. Update value of property (text after the '=' character) to the new value.
4. Restart the JCS.

Copy Customized server_jcs.xml Settings

Any customizations to the server_jcs.xml file you have made are overwritten during the upgrade. If you have changed any properties in the jcs-home/conf/server_jcs.xml file for an 8.1SP2 installation (see the properties in SAMPLE.server_jcs.properties for a list of possibilities), you need to copy the customized settings to the new server_jcs.properties file.

To copy customized server_jcs.properties settings

1. Make a copy of the existing 8.1SP2 jcs-home/conf/server_jcs.xml file.
2. Install the r12 Java CS.

   The installation overwrites the old customized server_jcs.xml file.
3. Copy any custom modifications from the copy of the original 8.1SP2 server_jcs.xml file to jcs-home/conf/override/server_jcs.properties.

Delete Connector.xml Files

If you are upgrading from 8.1SP2, delete any *jcs-home*/conf/override/*/connector.xml files you have not explicitly modified manually. These files are now included only as unused templates named SAMPLE.connector.xml and need to be renamed to connector.xml (and left in their original directory) to actively allow overriding of the properties in them.

Switching Between Java and C++ Versions of a Connector is not Supported

Java CS does not support switching between the bundled Java and C++ versions of a connector.

If you temporarily need to have both a bundled Java and C++ versions of a connector active at the same time (for example, during migration), then you must have two separate installations of both the Provisioning Server and the Provisioning Manager: one for the bundled Java CS connectors and another for the C++ version.

Java CS Can Not be Installed on Network Drives

You cannot install the Java CS on a network drive because Windows does not allow the Java CS Windows Service to start. You must install Java CS onto a local drive (localhost) in order to run properly.

Installation Components

The Java CS installation components include:

- Java Connector Server (can be expanded through the sample connector archive).
- Java Connector Server SDK
- Connector Xpress

Installed Java CS sample connectors include the following:

- SDK (requires SDK user interface plug-in to Provisioning Manager to be configured).
- JDBC for databases
- SDK DYN (uses DYN schema and DYN Provisioning Manager so no additional configuration is required)
- JNDI for LDAP directories
- SDK SCRIPT (also uses DYN schema and DYN Provisioning Manager so no additional configuration is required, with connector implemented in JavaScript).

Provisioning Server Registration

We recommend that you always register the Java CS with the Provisioning Server the first time you install a Java CS. (see page 19)

You do not need to register with a Provisioning Server for any subsequent additional Java CS installations.

Registering tells the Provisioning Server to use the Java CS being installed to manage all of the static connectors that have been deployed to it. If a specific static or dynamic connector needs to be managed by a different Java CS, this can be configured using Connector Xpress at any time.

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

More information:

Install the Java CS (see page 19)

Install the Java CS

To install the Java CS

1. Locate the Identity Manager Provisioning Components download or other media.
2. Start the Product Explorer.
3. Select Install Products, Server.
4. Select Connector Server (Java).
5. 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.

6. When prompted complete the following fields on the Connector Server Configuration screen.

   LDAP Port

   Defines the port Java CS listens to.

   Production Java CS port: 20410

   Development Java CS port: 20412

   LDAPS Port

   Defines the secure port Java CS listens to.

   Production Java CS secure port: 20411

   Development Java CS secure port: 20413

Component Password

Defines the password used to authenticate to this Java CS.

Click Next.

The wizard installs the Java CS.

Install the Java Connector Server SDK

To install the Java CS SDK

1. Locate the Identity Manager Provisioning Components download or other media.
2. Start the Product Explorer.
3. Select Install Products, Developer Resources.
4. Select Connector Server SDK (Java).
5. Follow the onscreen instructions to complete the installation.

Note: The Java CS SDK does not need to be installed on the same machine as the Java CS.

For more information on the contents of the Java CS SDK image, see the following:

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

How You Install the Java CS or Connector Xpress Silently

The Java CS and Connector Xpress both support a silent mode of installation. To run a silent install you need to create a response file. The process for the Java CS 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 manually 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: You must 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

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

The samples are installed and registered.

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

Static Connector Uninstall

We do not support uninstallation of the sample connectors or reverting to C++ versions of any bundled static connectors for this release.

Manual Activation of JDBC Vendor Support

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

- DB2
- MS SQL (Native Authentication on Windows)
- Sybase

More information:

How to Activate JDBC DB2 for Z/Os Vendor Support (see page 23)
How to Activate MS SQL Native Authentication on Windows (see page 24)
How to Activate JDBC Sybase Vendor Support (see page 25)

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.

To activate JDBC DB2 for Z/Os vendor support, do the following:

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. Shutdown and restart the Java CS service.
4. Shutdown and restart all Connector Xpress sessions.

How to Activate MS SQL Native Authentication on Windows

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

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

To activate MS SQL Native Authentication on Windows, do the following:

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're 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. Shutdown and restart the Java CS service.
4. In Connector Xpress, when you set up MS SQL datasource, select the Native check box on the Edit Sources dialog box.

   This adds the following to the JDBC URL used for the connection:

   integratedSecurity=true

   Note: For more information on 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:

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 Java CS service.
5. Shutdown and restart all Connector Xpress sessions.

Install Connector Xpress

Important! Be sure to disable all antivirus software before installing Java CS, Java CS SDK, and Connector Xpress.

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

To install Connector Xpress

1. Locate the Identity Manager Provisioning Components download or other media.
2. Start the Product Explorer.
3. Select Install Products, Clients.
4. Select Connector Xpress.
5. Follow the onscreen instructions to complete the installation.

Default Installation Directories

The Java CS and Java CS SDK installation programs create directories for the Java CS executables and associated files. The default Windows and Solaris directories are listed in the following table. Your actual installation directories depend on your operating system and selections during the installation process.

The following notation is used throughout this guide to refer to various CA product installation directories:

Path Notation	Default Directory (Windows)	Default Directory (Solaris)
jcs-sdk-home	C:\Program Files\CA\Identity Manager\Connector Server	opt/CA/Identity Manager/Connector Server
jcs-home **Note:** Under production JCS, this has the same structure as JCS home	jcs-sdk-home\build\dist	jcs-sdk-home\build\dist
sample-home	C:\Program Files\CA\Identity Manager\Connector Server SDK\connectors\sdk	opt/CA/Identity Manager/Connector Server SDK/connectors/sdk
im-home	C:\Program Files\CA\Identity Manager	opt/CA/Identity Manager
imps-home	C:\Program Files\CA\Identity Manager\Provisioning Server	opt/CA/Identity Manager/Provisioning Server
conxp-home	C:\Program Files\CA\Identity Manager\Connector Xpress	opt/CA/Identity Manager/Connector Xpress

Start the Java CS

You can start the Java CS using one of the following methods.

- Windows Service (see page 27)
- Unix Daemon (see page 28)
- Windows Command Line (see page 28)

Windows Service

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

The Java CS installation process creates a Windows service to run the Java CS. This service is called CA Identity Manager - Connector Server.

Start the Java CS Service from the Windows Command Line

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

```
net start jcs
```

Stop the Java CS Service from the Command Prompt Window

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

```
net stop jcs
```

Start the Java CS Using the Unix Daemon

The Java CS installation process creates a startup script called JCS and links it to the rc.d system on the local system. This script responds to standard start, stop, restart and status requests, and automatically runs the Java CS 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/jcs START

Starts the Java CS daemon

/etc/init.D/jcs restart

Restarts the Java CS daemon

/etc/init.d/jcs stop

Stops the Java CS daemon

/etc/init.d/jcs status

Displays the status of the Java CS daemon

Run the Java CS from the Command Line

You can run Java CS interactively via 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

There are two ways to deploy a connector:

- Use the Connector Xpress and connector metadata to map the generic DYN schema to an endpoint system. This involves sending metadata for the connector information to a running Java CS via the Provisioning Server.

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

- Deploy a custom Java connector to the Java CS by copying the .jar files to the *jcs-home*/lib/ directory. 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 Java CS is used in conjunction with the Provisioning Server, the latter acts as the persistent store for connector metadata.

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 local file and then deploy a connector from that file. This can be useful in cases where the same settings apply on multiple Provisioning Servers, for example, when you move connectors from a test environment to a production environment after testing is complete.

Connector Undeployment

You can use Connector Xpress to undeploy connectors. This deletes the connector from the Provisioning Server's persistent storage as well as from the Java Connector Server. You can also delete connectors (or instances of their parent endpoint types) using the Provisioning Manager user interface. In both cases, you do not need to restart the Java CS, and Connector Xpress does not delete data on the endpoint.

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 Java CS

This section contains the following topics:

Server Configuration File

The server_jcs.xml file contains configuration attributes for both the Java Connector Server and the Apache DS directory which acts as its host. It 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 server_jcs.xml file is overwritten by the installer during upgrades, and consequently we recommend that if you make any manual customizations you should 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. These 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 via 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 Java CS. 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 Solaris.

Note: This setting can be potentially overridden by manual settings in the server_jcs.properties file.

Note: For more information, see Solaris Considerations.

ldapPort

Specifies the port which the Java CS listens on for insecure connections. This should be set to one of the recommended ports unless multiple C++ Connector Servers or Java Connector Servers are to be run on the same machine. Where a secure port is configured it should be used instead, this insecure port may be useful for debugging purposes. By default, the Java CS installer mandates the use of the ldapsPort exclusively.This should be set to one of the following port numbers:

- Production Java CS: 20410
- Development Java CS: 20412

ldapsPort

Specifies the port which the Java CS listens on for secure connections. The ldapsPort, with associated properties enableLdaps, ldapsCertificateFile 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 Java CS log4j.properties file to trace LDAP requests as they are delivered to the Java CS.

This should be set to one of the following port numbers:

- Production Java CS secure port: 20411
- Development Java CS secure port: 20413

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

bootstrapSchemas

Specifies which LDAP schemas are known to the Java CS. 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 via 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 Java CS related. This service handles lazy activation of connectors as they are mentioned in an LDAP ADD requests.

interceptorConfigurations

Specifies any other standard ApacheDS interceptor services. Those not required by the Java CS have been deactivated.

connectorPersister

Specifies the Persister for connector and connector levels of the DIT. This is not required when the Java CS is accessed via the Provisioning Manager, but may be of interest in other deployment situations.

cryptoService

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

server_jcs.xml — Initial TLS Settings

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

Note: All these setting can be potentially overridden by manual settings in the server_jcs.properties file.

ldapsCertificateFile

Specifies the Java CS wide trusted certificate store. Specifies a path to the file which contains all of the trusted CA certificates to be used to verify identity of the endpoint Java CS machines when establishing SSL connections to it or from it to endpoint machines.

ldapsCertificatePassword

Specifies the password protecting the Java CS wide trusted certificate store.

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

{ALGORITHM}ciphertext where ALGORITHM would be set to 'CACRYPT' normally, for example, {AES}LQpBXeIjOMGSsGLU+IZi9Q==

connectorClientCertStore

Specifies the Java CS wide client certificate store. Specifies a path to the file which contains end entity certificates used to verify the identity of the endpoint server machines during SSL handshakes.

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

Specifies that during SSL handshakes the peer certificate sent by the endpoint to which a connection is made, is not verified for trust (ldapsCertificateFile connectorCertStore value is thus ignored and not required for outbound SSL connections in this configuration).

Default: False

connectorSSLTrace

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

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 Java CS wide changes made to these attributes in the server_jcs.xml file will be lost when upgrading, so achieve the same effect via 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 Java CS, 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 and/or override values in server_jcs.xml.

Each connector's jar file has a connector.xml file within it, but to allow easier customization of the values within it after deployment, a matching file under *jcs_home*/conf/override/*connector*/connector.xml can be used to override settings.

For an example of a connector.xml file see:

jcs_home/conf/override/jdbc/connector.xml

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

Connector Pool Configuration

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

- Via poolConfig for JNDI and most connectors()

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

- Via 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 change pool related settings

1. Back-up the following file:

   jcs-home/conf/override/jdbc/connector.xml

   Note: We recommend you use the name BAK.Connector.xml.

2. Edit the connector.xml file.
3. Restart the Java CS.

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 should be re-established immediately.
- **Retriable—**The connection to the endpoint has encountered what may 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 Java CS client.
- **Busy**—The endpoint has reported it is too busy to complete a request (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), in which case a retry loop is started with a separate retry delay and count settings (typically much longer then those for the Retriable case).

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\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

Edit JVM Memory Options on Unix

To edit JVM options on Unix, place a file called jvm_options.conf in the data folder containing a one-line list of the java arguments.

This will override the following existing defaults:

FILE: ../data/jvm_options.conf (default settings)

-Xms256M -Xmx256M -server

FILE: ../dataf/jvm_options.conf (for example, a basic 64-bit solaris config with low memory)

-Xms128M -Xmx192M -server -d64

Note: If you want to keep these defaults, you must specify them in the conf file.

Note: For more information, see your Java documentation for a full list of available Java tuning commands.

JXplorer to Java CS Connection Parameters

If you have a standard installation, you can use the following parameters to connect to Java CS from JXplorer:

Host

Specifies the host machine name of the Java CS

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.

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").

Change the Java CS Administration Stored Password

To change the Java CS administration users stored password

1. Start Java CS.
2. Bind to Java CS using the following details:

   Host

   Specifies the host machine name of the Java CS

   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.

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

Set the TLS Certificate Password

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

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

To set the TLS Certificate password

1. Shutdown the Java Connector Server.
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 Java CS.

   Note: The initial password for the keystore after installation is *secret.*

Adjust Java CS Service Start Parameters

To adjust any Java CS 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 usually configured via the connector.xml file for an individual connector (for example *jcs-home*/conf/override/jdbc/connector.xml) file rather than in the server_jcs.xml global configuration file.

Determine Java CS Version Number

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

```
cd jcs_home/lib

java -jar JCS.jar
```

The Java CS version number is printed. For example:

```
CA Identity Manager - Java Connector Server

Version 1.0.nnnnn
```

Where Connector Xpress Stores User Preferences

Connector Xpress stores user preferences via 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: Java CS Logging

This section contains the following topics:

Logging Configuration

Log files are critical aids in troubleshooting problems with the Java CS and hosted connectors. System administrators should start with jcs_daily* files and work downwards to connector-specific log files as required. Connector developers should start with /endpoint type *nam/jcs_conn_connector-name** files and work upwards to jcs_daily* files.

How Logging is Managed

Both the Java CS and ApacheDS code bases use the NLOG4J library to manage logging, which is configured via 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, along with a description of the circumstances under which they may be useful.

Use the log4j_verbose.properties File for Troubleshooting

The jcs-home/conf/log4j_verbose.properties file includes useful logging settings that you can use for troubleshooting.

To use the jcs-home/conf/log4j_verbose.properties file

1. Shutdown the Java CS.
2. Enter the following command:

   cd <javacs-home>/conf
3. Delete the log4j.properties file.

   Note: A copy of this file is available as log4j_quiet.properties.
4. Copy log4j_verbose.properties to log4j.properties.
5. Delete all files and directories under the following directory:

   <javacs-home>/log/
6. Restart the Java CS.
7. After you have finished troubleshooting, do the following:
   a. Shutdown the Java CS.
   b. Copy log4j_quiet.properties to log4j.properties.
8. Restart the Java CS.

Connector Specific Logging

The logging level for connector specific logs can be configured 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.

To turn on connector instance logging

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 will 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 via the logLevelAttr attribute of the configured MetaConnectorFactory, in which case the Log4J Levels listed in the table above are the possible values.

Setting the Java CS Logging Level

To set the the Java CS system-wide logging level, set the line of the form log4j.logger.com.ca=DEBUG, then restart the Java CS.

This setting controls the logging level for the Java CS 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 on the settings in the file, see the comments in the file.

The jcs_conn_*connector-name*.log file generated for each active connector instance contains most of the logging data related to its corresponding connector. However, in the following cases it may be necessary to look in the system-wide file:

- Connectors that make use of third-party libraries
- Connectors that have been 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 CS logging
jcs_daily.log.*YYYYMMDD*	ApacheDS and Java CS 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 following file:

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

Note: The generated metadata is usually the primary resource about the problem being diagnosed, and should be analyzed in conjunction with Connector Xpress log files.

Provisioning Manager Logging

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

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

Note: To view, compress or archive any log files for the current day, you will need to copy the log file because the Provisioning Manager continues to write to them. Save these log file copies using the original file name, but in a different directory.

These logs are relevant both to Java CS 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 receives only messages of a certain severity level or levels, such as error, warning, or fatal messages.

To turn on Provisioning Server logging

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 will 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 receives only messages of a certain severity level or levels, such as error, warning, or fatal messages.

To turn on Provisioning Manager logging

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.

Index