

CA Identity Manager

Java Connector Server Implementation Guide

r12.5 SP5



Provisioning Components r1

This documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2010 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

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 you need for your Home Office, Small Business, and Enterprise CA Technologies 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 Technologies product documentation, you can send a message to techpubs@ca.com.

If you would like 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 9

Knowledge Requirements	9
About the Java Connector Server	9
Endpoint Systems	10
Connector Deployment	10
Java CS Architecture	11
Java CS Framework	12
What Java CS Does	12
Connectors	12
Connector Interfaces	13
Connector Implementation	13

Chapter 2: Installing Java CS 15

System Requirements	15
Required Privileges	16
Solaris Considerations	16
Time Zone Considerations	16
Installation Components	17
Provisioning Server Registration	18
Install the Java CS	18
Install the Java CS SDK	19
How You Install the Java CS or Connector Xpress Silently	20
Install the Connector Samples	20
Connector Uninstall	21
Manual Activation of JDBC Vendor Support	21
How to Activate JDBC DB2 for Z/Os Vendor Support	21
How to Activate Microsoft SQL Native Authentication on Windows	22
How to Activate JDBC Sybase Vendor Support	22
Install Connector Xpress	23
Default Installation Directories	24
Start the Java CS	24
Windows Service	25
Start the Java CS Service from the Windows Command Line	25
Stop the Java CS Service from the Command Prompt Window	25
Start the Java CS Using the UNIX Daemon	25
Run the Java CS from the Command Line	26

Chapter 3: Deploying Connectors	27
How you Deploy a Connector	27
Connector Deployment	27
Connector Undeployment	28
Connector Xpress Wizard	28
Chapter 4: Configuring Java CS	29
Server Configuration File	29
server_jcs.xml file—Configure Attributes	30
Validators and Converters	32
Connector Configuration File	32
server_jcs.xml — Initial TLS Settings	33
Connector Pool Configuration	34
Edit JVM Memory Options on Windows	35
Edit JVM Memory Options on UNIX	35
JXplorer to Java CS Connection Parameters	36
Change the Java CS Administration Stored Password	37
Set the TLS Store Certificate Password	38
Adjust Java CS Service Start Parameters	39
Connection Pooling	39
Determine Java CS Version Number	39
Where Connector Xpress Stores User Preferences	39
Chapter 5: Java CS Logging	41
Logging Configuration	41
How Logging is Managed	41
Connector Specific Logging	41
Turn on Connector Instance Logging	42
Logging Severities	42
Setting the Java CS Logging Level	43
Files in the Log Directory	43
Connector Xpress Logging	44
Provisioning Manager Logging	44
Turn on Provisioning Server Logging	45
Turn on Provisioning Manager Logging	45

Glossary

47

Index

51

Chapter 1: Java Connector Server

This section contains the following topics:

[Knowledge Requirements](#) (see page 9)

[About the Java Connector Server](#) (see page 9)

[Connectors](#) (see page 12)

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 (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 (CCS). However, it has a Java API instead of a C++ API, which allows connectors to be implemented in Java. In addition, the Java CS 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 Java CS 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. An endpoint is managed by a connector server using a specific connector.

A *connector* is the software that enables communication between a Provisioning Server and an endpoint system. A dynamic connector can be generated by Connector Xpress, and a custom static connector can be developed in Java or C++.

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.

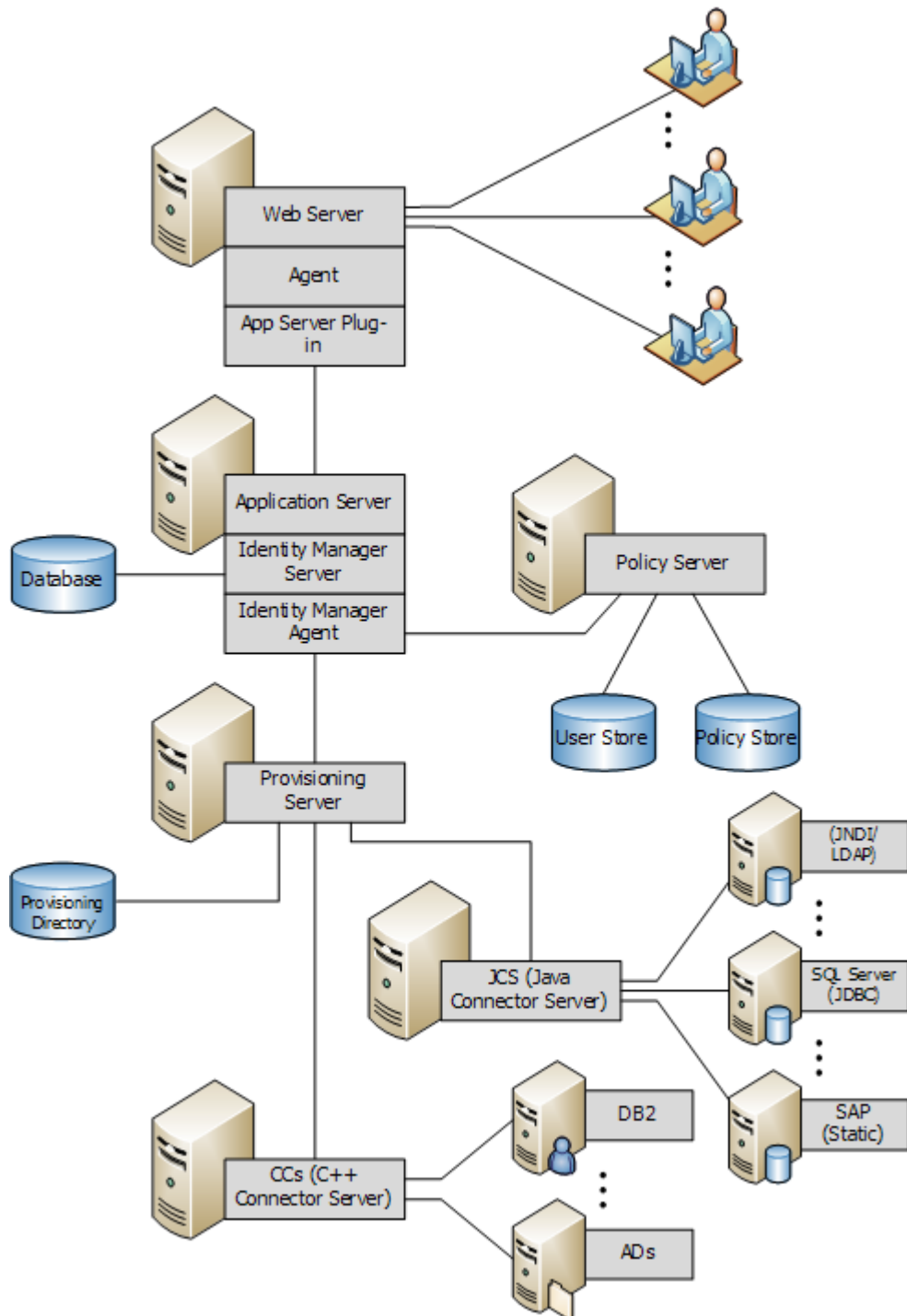
Users use Connector Xpress to generate and maintain XML metadata for JDBC and JNDI dynamic connectors. Developers can also maintain data for other connectors manually, or adjust metadata for released connectors (for instance adding site-specific mappings for custom attributes).

Connector Deployment

A connector's software is deployed to a Connector Server using an implementation bundle .jar file for a Java CS 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 Provisioning Directories.

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 web sites:

- <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

The Java CS 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.

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 the Java CS hosts must implement at least one (and possibly all, for example, 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 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 Java CS

This section contains the following topics:

[System Requirements](#) (see page 15)

[Installation Components](#) (see page 17)

[Provisioning Server Registration](#) (see page 18)

[Install the Java CS](#) (see page 18)

[Install the Java CS SDK](#) (see page 19)

[How You Install the Java CS or Connector Xpress Silently](#) (see page 20)

[Install the Connector Samples](#) (see page 20)

[Connector Uninstall](#) (see page 21)

[Manual Activation of JDBC Vendor Support](#) (see page 21)

[Install Connector Xpress](#) (see page 23)

[Default Installation Directories](#) (see page 24)

[Start the Java CS](#) (see page 24)

System Requirements

Java CS requires CA Identity Manager r12.5 SP2.

Note: You do not need to install the Java CS SDK (or Java CS 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.

Java CS 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 Java CS, Java CS SDK, and Connector Xpress. If anti-virus software is enabled while installation processes are taking place problems can occur. Remember to reenale your antivirus protection after you complete installation.

Required Privileges

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

Any user can run the Java CS SDK installer, or 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. The default setting is too low to allow the Java CS 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 Java CS to start.

We recommend that you set the `ulimit -n` to at least $50 + 2 \times$ ("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 Java CS), we recommend that you install all components on computers configured to use the same time zone.

The Java CS 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 Java CS, and does the opposite transformation from UTC to the local time zone for query results.

Installation Components

The Java CS related installation components include:

- Java CS

The Java CS 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.

- Java CS SDK
- Connector Xpress

Connectors which are part of the standard Java CS installer include:

- JDBC for databases
- JNDI for LDAP directories
- AS400
- Kerberos
- Lotus Notes Domino
- Oracle (Native Users)
- PeopleSoft
- SAP

Additional connectors which are part of the Java CS static samples package include the following:

- SDKDYN—Demonstrates flat connector implementation managing properties files on the local filesystem.
- SDKFVS—The SDKFVS connector has the same code as the SDKDYN connector but demonstrates implementation of a hierarchical connector, rather than a flat connector. In particular, the filesystem directories, and the properties files they contain, can be managed.
- SDKCOMPOUND – Demonstrates the use of JSON-encoded compound values, which means complex data structures can be flattened out into single attribute values.
- SDKSCRIPT— Similar functionality to the SDKDYN connector, but implemented in JavaScript.
- SDKUPOSCRIPT – Scripted connector which performs a similar function to the C++ UPO connector, that is, sending email.

Note: Unless otherwise stated, SDK connectors use the DYN schema and DYN plug-in within Provisioning Manager so no additional configuration is required to the Provisioning Server to use SDK connectors.

Provisioning Server Registration

We recommend that you always register the Java CS with the Provisioning Server.

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

More information:

[Install the Java CS](#) (see page 18)

Install the Java CS

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

To install the Java CS

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.

ProvisioningConnectorServer

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 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.

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

Click Next.

6. When prompted, select whether you want to activate FIPS 140-2 Compliance Mode.

Click Next.

The wizard installs the Java CS.

Install the Java CS SDK

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

To install the Java CS SDK

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

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

Note: For more information about 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 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 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

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.

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

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 Java CS, or because additional specific manual configuration is required.

More information:

[How to Activate JDBC DB2 for Z/Os Vendor Support](#) (see page 21)

[How to Activate Microsoft SQL Native Authentication on Windows](#) (see page 22)

[How to Activate JDBC Sybase Vendor Support](#) (see page 22)

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 Microsoft SQL Native Authentication on Windows

Microsoft 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.

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.

To activate Microsoft 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 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. Shutdown and restart the Java CS 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:

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

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.

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

To install Connector Xpress

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.
 ProvisioningConnectorXpress
4. 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)
<i>jcs-sdk-home</i>	C:\Program Files\CA\Identity Manager\Connector Server	/opt/CA/IdentityManager/Connector Server
<i>jcs-home</i>	<i>jcs-sdk-home</i> \build\dist	<i>jcs-sdk-home</i> \build\dist
Note: Under production JCS, this has the same structure as JCS home		
<i>sample-home</i>	C:\Program Files\CA\Identity Manager\Connector Server SDK\connectors\sdk	/opt/CA/IdentityManager/ConnectorServer SDK/connectors/sdk
<i>im-home</i>	C:\Program Files\CA\Identity Manager	/opt/CA/IdentityManager
<i>imps-home</i>	C:\Program Files\CA\Identity Manager\Provisioning Server	/opt/CA/IdentityManager/ProvisioningServer
<i>conxp-home</i>	C:\Program Files\CA\Identity Manager\Connector Xpress	/opt/CA/IdentityManager/ConnectorXpress

Start the Java CS

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

- [Windows Service](#) (see page 25)
- [Unix Daemon](#) (see page 25)
- [Windows command line](#) (see page 26)

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 named 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 im_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 im_jcs
```

Start the Java CS Using the UNIX Daemon

The Java CS 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 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/im_jcs start`

Starts the Java CS daemon

`/etc/init.D/im_jcs restart`

Restarts the Java CS daemon

`/etc/init.d/im_jcs stop`

Stops the Java CS daemon

`/etc/init.d/im_jcs status`

Displays the status of the Java CS daemon

Run the Java CS from the Command Line

You can run Java CS 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](#) (see page 27)
- [Connector Deployment](#) (see page 27)
- [Connector Undeployment](#) (see page 28)
- [Connector Xpress Wizard](#) (see page 28)

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 Java CS through 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 connectors `jcs-connector-*.jar` file and any additional `.jar` files it depends on to the `jcs-home/lib/` directory, and then restart the Java CS. 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 with the Provisioning Server, the latter acts as the persistent store for connector metadata, therefore the Java CS 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. Saving metadata to a project file is 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. Undeploying a connector deletes the connector from the Provisioning Servers persistent storage and from the Java CS. 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](#) (see page 29)

[Validators and Converters](#) (see page 32)

[Connector Configuration File](#) (see page 32)

[Edit JVM Memory Options on Windows](#) (see page 35)

[Edit JVM Memory Options on UNIX](#) (see page 35)

[JXplorer to Java CS Connection Parameters](#) (see page 36)

[Change the Java CS Administration Stored Password](#) (see page 37)

[Set the TLS Store Certificate Password](#) (see page 38)

[Adjust Java CS Service Start Parameters](#) (see page 39)

[Connection Pooling](#) (see page 39)

[Determine Java CS Version Number](#) (see page 39)

[Where Connector Xpress Stores User Preferences](#) (see page 39)

Server Configuration File

The `server_jcs.xml` file contains configuration attributes for both the Java CS 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 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: The manual settings in the server_jcs.properties file can potentially override this setting.

Note: For more information, see Solaris Considerations.

ldapPort

Specifies the port which the Java CS listens on for insecure connections. Set the port to one of the recommended ports unless multiple C++ Connector Servers or Java CSs 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 Java CS installer mandates the use of the ldapPort exclusively. Set the port 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`, `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.

Set the port 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 Java CS installer sets the default `ldapsCertificatePassword`.

bootstrapSchemas

Specifies which LDAP schemas the Java CS 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 Java CS 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 Java CS have been deactivated.

connectorPersister

Specifies the Persister for connector and connector levels of the DIT. The Persister is not required when the Java CS 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 converters 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 Java CS 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 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, 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.

IdapsCertificateFile

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

IdapsCertificatePassword

Specifies the password protecting the Java CS certificate store specified in IdapsCertificateFile.

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

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

connectorClientCertStore

Specifies the Java CS 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

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 IdapsCertificatePassword.

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 Java CS 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 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 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 Java CS 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>

Edit JVM Memory Options on UNIX

To edit JVM options on UNIX, (for example, a basic 64-bit solaris configuration with low memory) 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 Java CS

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

-Xmx

Specifies the maximum heap memory allowed for the Java CS.

Example: `-Xmx192M`, specifies the maximum heap memory allowed for the Java CS 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 Java CS Connection Parameters

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

Note: For more information on Jxplorer, see <http://www.jxplorer.org>.

Host

Specifies the host server 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").

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

Change the Java CS Administration Stored Password

To ensure better security across a deployment you can change the Java CS administration users 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 computer 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 because the last restart and all passwords are accepted as valid for bind requests. This allows phased upgrade of passwords for a Java CS where multiple Provisioning Servers connect to the Java CS, but do not update their stored passwords for the Java CS at the same time.

Set the TLS Store Certificate Password

When LDAP clients request TLS secured connections to the Java CS, 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

To set the TLS Certificate store password

1. Shut down the Java CS.
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 password for the keystore is the password that you set during the Java CS installation.

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 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 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 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: Java CS Logging

This section contains the following topics:

[Logging Configuration](#) (see page 41)

[How Logging is Managed](#) (see page 41)

[Connector Specific Logging](#) (see page 41)

[Provisioning Manager Logging](#) (see page 44)

Logging Configuration

Log files are critical aids in troubleshooting problems with the Java CS 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 Java CS 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 Java CS, cause the issue to occur, shutdown the Java CS, 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.

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 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 Java CS Logging Level

To set the Java CS systemwide logging level, set the line of the form `log4j.logger.com.ca=DEBUG`, then restart the Java CS, or use the `log4j_verbose.properties` file instead. 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 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
<code>jcs_daily.log</code>	Today's ApacheDS and Java CS logging
<code>jcs_daily.log.YYYYMMDD</code>	ApacheDS and Java CS logging for date
<code>endpoint-type/jcs_conn_connector-name.log</code>	Specific logging output for named connector
<code>endpoint-type/jcs_conn_connector-name.log.YYYYMMDD</code>	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. We recommend that you analyze the generated metadata using the Connector Xpress log files.

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 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 logs 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 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.

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.

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 a Provisioning Server and an endpoint system. A dynamic connector can be generated by Connector Xpress, and a custom static connector can be developed in Java or C++.

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. An endpoint is managed by a connector server using a specific connector.

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 (Java CS)

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 (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

A

Apache Directory Server • 12, 29

C

Connector Xpress • 27, 39, 44

connector.xml file • 32

connectors

 defined • 12

 deploying • 27

 functionality • 13

converters • 32

F

files

 connector.xml • 32

 logging • 32, 41, 44

 server_jcs.xml • 29, 32, 36

I

installation • 16, 17, 23

J

Java Connector Server

 architecture • 11

 directories • 23

 functionality • 12

 installing • 16, 17, 23

 requirements • 15

 running • 24, 25

Java Connector Server SDK

 installing • 16, 23

JavaBeans • 29, 32

L

LDAP

 data conversion • 12

 data validation • 12

 mapping • 12, 13

 operations • 29

log files • 32, 41, 44

M

metadata • 12, 13

P

pooling • 13, 39

Provisioning Server • 11, 27, 44

S

server_jcs.xml file • 29, 32, 36

Spring Framework • 12, 29

SuperAgent • 12

V

validators • 32