Source Guide

Version 0.1
July 2006

Contents

Overview

The OpenDS Directory Server is a pure Java implementation of a network-accessible database that is able to store information in a hierarchical form. Clients may communicate with it using a standard protocol (e.g., LDAP) to retrieve and update information in a variety of ways. We aim to provide unmatched performance and scalability, robust error handling and debugging capabilities, easy extensibility, and innovative feature sets.

As its name implies, the OpenDS Directory Server is available under the Common Development and Distribution License (CDDL) open source license. This is an OSI-approved open source license that provides a great deal of flexibility to users and developers while also offering legal benefits like rights to use associated patents and protection against infringement claims. We have decided to adopt an open development model, and welcome external participation. This document intends to provide the information that is needed to access and understand the OpenDS code base and to be able to build the server.

Copyright and trademark information

Feedback

Please direct any comments or suggestions about this document to:
dev@OpenDS.dev.java.net

Acknowledgements

The general format of this document was based on the documentation template used by OpenOffice.org.

Modifications and Updates

Date	***Description of Change***
July 2006	Initial draft

Prerequisites

Java Development Environment

The OpenDS Directory Server is written entirely in Java and therefore a Java development environment must be available. The Java Runtime Environment (JRE) is all that is required to use a pre-built version of the OpenDS Directory Server, but the full Java Development Kit (JDK) is needed to be able to compile the code.

Java SE 5.0 or later is required to build and run the Directory Server. At the time this document was written, it has also been found to run very well in recent Java SE 6.0 (also known as Project Mustang) early access builds. In many cases, Mustang builds provide notably better performance and improved debugging capabilities, which make it particularly attractive for use by developers.

The Java SE 5.0 development kit is freely available for download at http://java.com/en/download/ for Solaris (SPARC, x86, and x64 systems), Linux (x86 and x64 systems), and Windows (x86 systems). Development kits for other platforms (including HP-UX, AIX, Mac OS X, and FreeBSD) are provided by third parties. The Java SE 6.0 early access builds are freely available for download from https://mustang.dev.java.net/.

If you intend to examine or modify the OpenDS code, it may also be useful (although it is not required) to obtain an IDE to make it easier to develop and/or navigate the source code. Many Java IDEs are available, including both open source and proprietary offerings. The NetBeans IDE is a popular open source development environment that can be downloaded with the Java Development Kit for added convenience. The Eclipse Project is also a popular open source choice among Java developers.

Subversion Client

The OpenDS source code is stored in a Subversion repository. Subversion is a version control system, much like CVS, which can be used to track changes to source code and other files over time. It provides information about what changes have been made and by whom, and it helps significantly ease the process of managing multiple individuals making changes to the code at the same time.

Subversion clients come in many different forms:

- It has traditionally been available as a command-line utility, and this is the form that many experienced users prefer. This is the primary interface provided by the Subversion project, and the source code for this client is available for download at http://subversion.tigris.org/project_packages.html. That site also contains information for obtaining precompiled versions of this software for a number of platforms. It should be

noted that while this site does indicate that Solaris versions of the Subversion client are available via the SunFreeware site, they are also available through Blastwave, which provides a convenient `pkg-get` interface for managing dependencies between the packages.

- On Windows systems, the TortoiseSVN tool integrates with the Windows Explorer utility and in many ways can make interacting with a Subversion repository much like dealing with files on the local filesystem.

- A number of Java IDEs now include Subversion support, either bundled with the IDE or available as a plugin. Both the NetBeans and Eclipse projects offer plugins that can be used to interact with Subversion repositories from within the IDE.

Detailed instruction on using Subversion is beyond the scope of this document. There are a number of books and tutorials that cover this topic. The O'Reilly Version Control with Subversion book is a popular choice and is freely available online in HTML and PDF forms.

Checking Out the Source

The OpenDS Directory Server project is hosted at java.net, which is a popular site for hosting Java-based open source projects. It provides a lot of the framework required to manage these projects, including a version control system (like Subversion or CVS), mailing lists, discussion forums, an issue tracking system, and a website. The OpenDS Directory Server project can be found at https://OpenDS.dev.java.net/.

The Subversion repository for OpenDS is also hosted on java.net, and the URL to access it is https://OpenDS.dev.java.net/svn/opends. However this URL refers to the entire source tree which may include significantly more information than will be required for the vast majority of interaction with the repository. In most cases, https://OpenDS.dev.java.net/svn/opends/trunk/opends will be a better choice because it will only contain the most recent code for the Directory Server, and will not include other tags or branches that may have been defined, nor will it include the website content.

There are varying levels of access control that may be enforced for the repository. The OpenDS code is available for download anonymously (username "guest" with an empty password), but that will only grant read access. If you intend to actually commit changes that you make back into the Subversion repository, then you will need to create an account for yourself on java.net, which can be done at https://www.dev.java.net/servlets/Join, and you will also need to have a "developer" role in the OpenDS project (the process for becoming an approved developer is outside the scope of this document).

Using the Command-Line Subversion Client

Using the command-line Subversion client, the contents of the entire OpenDS source repository may be checked out using the following command (it should be all on a single line):

```
$ svn checkout --username guest
      https://OpenDS.dev.java.net/svn/opends/trunk/opends
```

You may be prompted as to whether you wish to trust the SSL certificate used by the java.net site, which you can choose to accept either on a one-time basis or permanently. You may also be prompted for a password for the account (if you're using the "guest" account, you may simply press ENTER to indicate that there is no password).

If Internet access is restricted and all Web traffic must go through a proxy server, then you will need to configure the client with information about that proxy. On UNIX-based systems (including Linux), the configuration files for Subversion are below `.subversion` in the user's home directory (i.e., `~/.subversion`). On Windows systems, they are below the `%APPDATA%\Subversion` directory. The `servers` file contains information about how to interact with Subversion servers, including proxy settings that may be required. It is possible to define

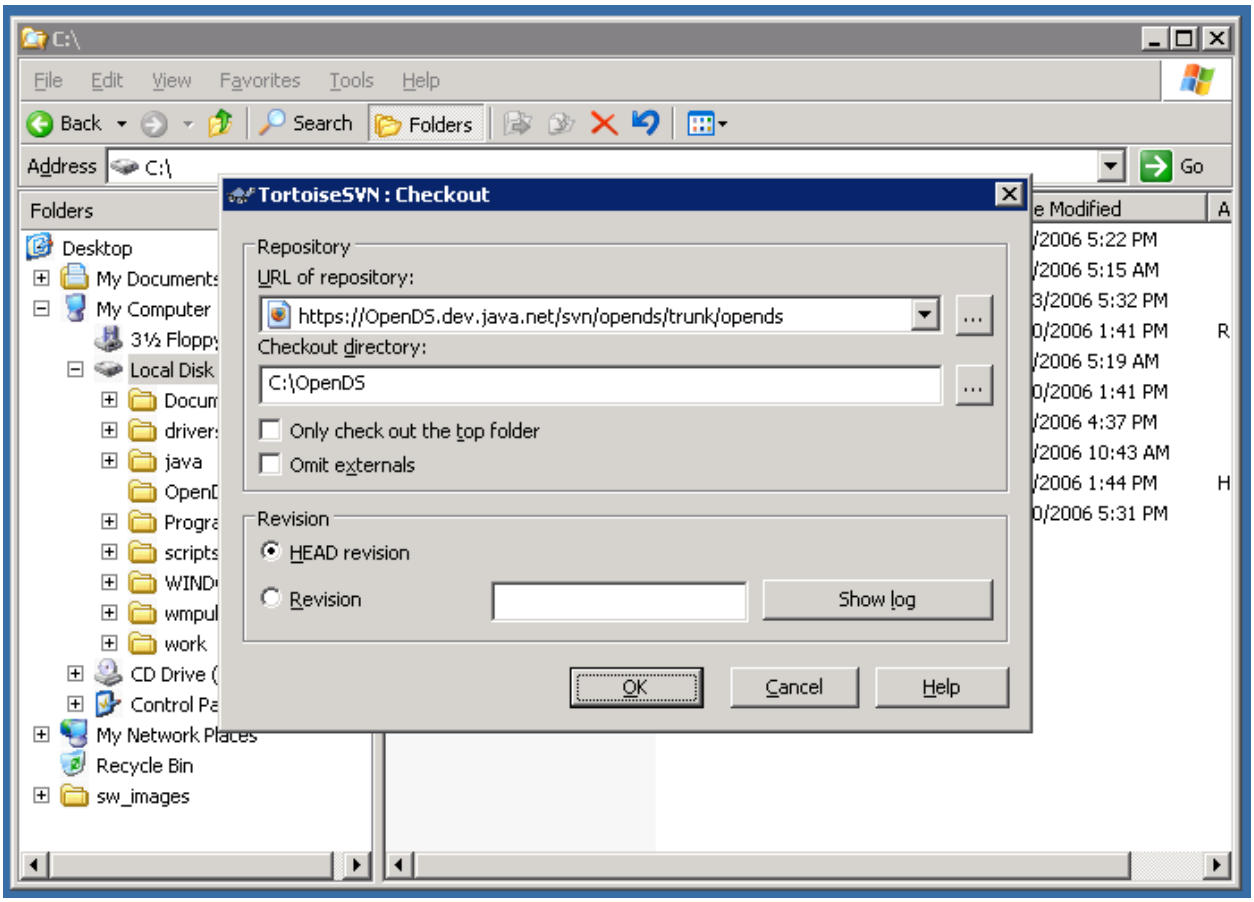
different proxy settings for different servers (e.g., in case there are some local servers that don't require a proxy to access, while there are others that do need a proxy). If all Subversion server access needs to go through a proxy, then it should be sufficient to make the changes in the "`[global]`" section at the end of the file, for example by adding lines like the following:

```
http-proxy-host = proxy.example.com
http-proxy-port = 8080
```

Additional information about using Subversion through a proxy server are available in comments in the `servers` file, or in the Subversion FAQ at http://subversion.tigris.org/faq.html#proxy.

Using the TortoiseSVN Client

If the TortoiseSVN client is to be used, then all interaction with the Subversion repository can be performed through the Windows explorer tool. To do this, create a new folder that you want to use to hold the code (e.g., `C:\OpenDS`). Then right-click on that folder name and choose the "SVN Checkout..." option from the context menu (or highlight that folder name and choose the "SVN Checkout..." option from the File menu). A dialog box will be displayed that allows you to provide information about the Subversion repository:



After clicking the "OK" button, you may be prompted as to whether you wish to accept the certificate presented by the server, and you will probably be asked to provide a username and password. Use a username of "guest" with an empty password to check out the code anonymously, or provide your real java.net username and password if necessary.

If you are required to use a proxy server, then that configuration may also be performed through the Windows Explorer interface. To do that, select the File menu and choose the "Settings" option from the TortoiseSVN sub-menu. The "Network" panel of the dialog box that appears can be used to define the proxy configuration.

Building OpenDS

Once the OpenDS source code has been checked out of the repository, it may be compiled and packaged into a format that can be used. The simplest way to accomplish this is using the command line as follows:

1. Go into the `OpenDS` directory that was created when the code was checked out of the repository. If the checkout was performed from the root of the repository, then this may need to be something like "`OpenDS/trunk/opends`". The correct directory should contain the `build.xml`, `build.sh` and `build.bat` scripts.

2. On UNIX-based systems, issue the command "`./build.sh`" to compile and package the server. On Windows systems, issue the command "`build.bat`". This will perform the minimal build required to be able to run the server. The packaged server may be found in the `build/package` subdirectory that is created during the build process.

As mentioned in the Prerequisites section, a Java 5.0 or higher SDK is required to build the server. The build script will automatically try to find a suitable Java installation, but if it is unable to do so then it may be necessary to explicitly specify the location of the Java SDK by setting the `JAVA_HOME` environment variable that provides the path to the install root of the SDK. On UNIX systems (at least those using the `sh`, `bash`, or `ksh` shells -- other shells may require a different syntax), this may be done as follows:

```
JAVA_HOME=/usr/java
export JAVA_HOME
```

On Windows systems, the path may be specified using a command like:

```
set JAVA_HOME=C:\java
```

With the `JAVA_HOME` environment variable defined, the build process should complete successfully.

There are a number of targets that may be provided when building the OpenDS Directory Server to control what processing is performed during the build. These targets include:

- `checkstyle` -- Performs a number of style checks against the OpenDS source code to ensure that it meets the code guidelines that have been defined for the project.

- `clean` -- Removes any dynamically-generated files that were created as part of an earlier build, including the `build` subdirectory and the `src/server/org/opends/server/util/DynamicConstants.java` source file.

- `compile` -- Compiles the main Directory Server source code into the `build/classes` directory. The compiled classes will not be packaged in any usable form, but this can be a relatively fast way to determine whether the code will compile.

- `package` -- Performs all of the work done by the `compile` target, but also creates a `build/package` directory that contains the OpenDS Directory Server in a form that may be used immediately for testing purposes, as well as bundled in a ZIP file that can be easily copied to another system. This is the default that will be used if no target is explicitly specified.

- `dsml` -- Performs all of the work done by the `package` target, but also builds a Web application that may be used as a DSML to LDAP gateway. The WAR file containing the DSML to LDAP gateway application will be placed in the `war` subdirectory under the packaged Directory Server.

- `test` -- Performs all of the work done by the `compile` target, but also compiles and runs all of the unit tests associated with the server.

- `javadoc` -- Performs all of the work done by the `dsml` target, but also generates Javadoc documentation from all of the server source code. The generated documentation will be placed in the `build/javadoc` subdirectory.

- `precommit` -- Performs all of the work done by the `checkstyle`, `javadoc`, and `test` targets. The code must build cleanly (i.e., with no warnings or errors) using this target before it is eligible to be committed to the repository.

- `daily` -- Performs all of the work that will be included as part of an OpenDS daily build. At present, this includes all of the work done by the `precommit` target, and also generates a code coverage report that details how much of the server code is exercised by the unit tests. In the future, this may include other processing that takes too long to be conveniently included in the `precommit` target.

- `weekly` -- Performs all of the work that will be included as part of an OpenDS weekly build. At present, this includes all of the work done in the `daily` target, but in the future may include additional processing that takes too long to be included in the daily build process.

- `all` -- Performs a build that includes virtually all build targets that have been defined.

- `rebuild` -- Performs a specialized version of the `package` build that does not invoke the `clean` target in the process. In particular, it rebuilds all of the source classes and creates the corresponding JAR files in the existing `build/package` directory without destroying any of the other content in that directory. This can be particularly useful for cases in which the OpenDS instance created under the build/package directory is configured and/or populated with data but a problem is found that requires a code change.

To use one of these build targets, simply include the target name on the command line. For example:

```
./build.sh precommit
```

will build the OpenDS Directory Server on a UNIX-based system using the `precommit` target.

OpenDS Repository Layout

As noted previously in this document, the Subversion repository containing the OpenDS codebase has a base URL of https://OpenDS.dev.java.net/svn/opends. If this URL is used to check out the code, there will be three top-level directories:

- `trunk` -- This directory structure contains the latest revision of the code. Most development work will be done in the portion of the repository below the `trunk` directory.

- `tags` -- This directory structure contains point-in-time snapshots of particular versions of the code. Tags provide a convenient way to check out a significant version by user-friendly name rather than requiring its version number. Tags should be treated in a read-only manner.

- `branches` -- This directory structure contains development branches of the code. These will be used as we near a release to allow continued development in the trunk while preparing that release in a more controlled manner.

At the time that the OpenDS code was initially made public, there were no tags or branches and therefore only the trunk contains any information. Over time, branches and tags will be created as needed. However, only those actively involved in developing the server code, and in particular those preparing for an upcoming release, will likely have any need to interact with tags and/or branches. For most individuals, only checking out the trunk code (e.g., using a base URL at or below https://OpenDS.dev.java.net/svn/opends/trunk) will be sufficient.

The following subdirectories can be found immediately below the `trunk` directory:

- `opends` -- This is effectively the root for the OpenDS codebase, as all the source code and build scripts exist below this directory.

- `www` -- This directory serves as the root for content that will appear on the https://OpenDS.dev.java.net/ website.

Unless you will be editing static content that appears on the project website, there is no real need to check out the contents of the `www` directory, so using a base URL of https://OpenDS.dev.java.net/svn/opends/trunk/opends will avoid the need to download a significant amount of information that is not generally needed to build or contribute to the OpenDS codebase.

The most frequently-accessed part of the Subversion repository will almost certainly be the content below the `trunk/opends` directory. The content contained in this directory include:

- `build.xml` -- This is the build definition that will be used by the Ant utility to specify the work that should be performed by the various types of builds that may need to be performed. See the Apache project page (http://ant.apache.org/) for information about the Ant build tool.

- `build.sh` -- This is a shell script that can be used to easily invoke the build process on UNIX-based systems. Because it ensures that appropriate environment variables and the correct Ant version are used, this script should be used to initiate the build rather than by directly invoking the Ant utility.

- `build.bat` -- This is a batch file that can be used to easily invoke the build process on Windows systems. As with UNIX systems, this batch file should be used to perform the build on Windows systems rather than directly invoking Ant.

- `PRODUCT` -- This file contains a set of information that will be dynamically compiled into the OpenDS product at build time. Information in this file includes the product name and version information. It also includes a placeholder that can be used for holding the bug IDs of any bugfixes included in the product (intended for use in supplying patches or hotfixes against an official build).

- `src` -- This subdirectory is the parent for all of the actual Java source code. The code for different portions of the server should reside in subdirectories below this parent.

- `src/server` -- This directory contains all of the source code for the core OpenDS Directory Server itself.

- `src/dsml` -- This directory contains the source code for the DSML to LDAP gateway component of the OpenDS Directory Server.

- `lib` -- This directory contains a set of libraries that should be included with the OpenDS Directory Server and are required to run it. Libraries in this directory include the Berkeley DB Java Edition and JavaMail.

- `ext` -- This directory contains a set of libraries that are required to build the OpenDS Directory Server, but are not actually needed to run the server. Elements included in this directory include the Apache Ant build tool, the Checkstyle source analysis tool, the EMMA code coverage tool, and the TestNG unit test framework.

- `resource` -- This directory contains a number of additional files that are needed during the build process and/or should be included in the packaged OpenDS Directory Server product. Components included here include wrapper scripts for invoking various tools and utilities, configuration and schema definition files, and legal notices.

- `tests` -- This directory is the parent directory for the various test frameworks used in the OpenDS project.

- `tests/unit-tests` -- This directory structure contains all of the code for the unit tests that will be run using the TestNG framework.

- `tests/integration-tests` -- This directory structure contains all of the components involved in the integration tests (also called functional tests) for the OpenDS Directory Server.

- `doc` -- This directory will contain all of the documentation to be delivered with the OpenDS Directory Server.

OpenDS Source Package Structure

As noted in the previous section, all of the Java source code for the core OpenDS Directory Server exists beneath the `trunk/opends/src/server` directory. Below this point, the directory structure mimics the Java package hierarchy. All code contained in the OpenDS project exists below the `org.opends` package, and all of the code for the core OpenDS Directory Server exists below the `org.opends.server` package. This code includes the following sub-packages:

- `api` -- This package contains all of the APIs that have been defined for the OpenDS Directory Server. The abstract classes and interfaces in this directory define the various types of components that may be implemented in the server. Some of these elements are primarily intended for use by the OpenDS developers, but others are for components that are well-suited for third-party developers in order to tailor the behavior of the server to meet their needs.

- `api.plugin` -- This package defines the OpenDS plugin API, which provides the ability to define custom code that is invoked at various points in operation and/or connection processing.

- `authorization` -- This package is used to hold classes for the access control implementation. At the time that the OpenDS code was initially released to the open source community, the access control subsystem had not yet been developed.

- `backends` -- This package holds the code used to implement the various backends for the OpenDS Directory server. The backends provide the mechanism for interacting with the actual data stored in the server.

- `backends.jeb` -- This package holds the code used to implement the OpenDS Directory Server backend that uses the Berkeley DB Java Edition as the data store. This is the primary backend for storing local data.

- `backends.task` -- This package holds the code used to implement the OpenDS Directory Server backend that is responsible for task processing, including the ability to schedule tasks for the future, and to create recurring tasks.

- `changelog` -- This package holds the code used to record changes that occur in the server so that those changes may be synchronized with other servers in the environment.

- `config` -- This package holds the code used to interact with the OpenDS configuration. It includes the default file-based configuration handler, as well as data structures for dealing with different types of configuration attributes.

- `controls` -- This package holds the code used to define the controls supported by the OpenDS Directory Server. Controls are primarily used by clients communicating via

LDAP, and may provide additional information for use in the request or response processing.

- `core` -- This package holds the code used to define the core server. This includes the startup and initialization code for the server, as well as various data structures and coordination logic used during server processing.

- `extensions` -- This package holds the code used to implement various extensions to the OpenDS Directory Server (e.g., extended operations, SASL mechanisms, password storage schemes, password validators, etc.).

- `loggers` -- This package holds the code used to implement the logging subsystem for the OpenDS Directory Server.

- `messages` -- This package holds the code used to define all of the individual message strings used by the OpenDS Directory Server. Each message that may be included in a response to the client or written to the error log must have a unique integer-based message ID, and the code defines those message IDs and associates them with the default English-language message strings that correspond to those IDs. The message strings for non-English messages will be defined in properties files rather than in the Directory Server code.

- `monitors` -- This package holds the code used to implement the Directory Server monitor providers which are used to provide information about the status of the server.

- `plugins` -- This package holds the code used to implement various Directory Server plugins that can provide custom code to be interjected into operation processing.

- `protocols.asn1` -- This package holds the code used to perform the ASN.1 BER encoding and decoding that is used by LDAP.

- `protocols.internal` -- This package holds the code used to perform internal operations within the Directory Server (e.g., operations that need to be performed as part of plugin processing).

- `protocols.jmx` -- This package holds the code used to interact with the OpenDS Directory Server using the JMX protocol.

- `protocols.ldap` -- This package holds the code used to interact with the OpenDS Directory Server using LDAP.

- `schema` -- This package holds the code used to provide the attribute syntax and matching rule implementations for the OpenDS Directory Server.

- `synchronization` -- This package holds the code used to provide the data synchronization capabilities for the OpenDS Directory Server.

- `tasks` -- This package holds the code used to implement various Directory Server tasks (e.g., to backup or restore data, to shut down or restart the server, etc.).

- `tools` -- This package holds the code used to implement a number of client-side tools (e.g., ldapsearch, ldapmodify) and administrative utilities (e.g., backup, restore, import, export) for the OpenDS Directory Server.

- `types` -- This package holds the code used to implement a number of data types used throughout the server.

- `util` -- This package holds various utility code used throughout the server.

OpenDS Code Style Guidelines

Many developers have their own unique coding styles that have been developed over time and allow them to write code quickly. Being required to write code in a "foreign" style can be frustrating in some cases, and has the potential to adversely impact productivity. On the other hand, it is important to ensure that the code is readable and maintainable by everyone involved. To address this, we have developed a set of coding style guidelines that we believe will help preserve readability and code quality while still allowing developers to maintain a sense of individual style in some areas.

It should be noted that many of the coding guidelines here are enforced by the Checkstyle utility, which is available at http://checkstyle.sourceforge.net/. For these checks, attempting to build using the `precommit` target will cause that build to fail if any of the associated requirements are violated. The checks performed by the Checkstyle utility over the entire codebase include:

- All source files must begin with the following header (note that the omission of the closing "`*/`" is intentional, as it is permissible to have additional content in the header after the "`CDDL HEADER END`" line):

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at
 * trunk/opends/resource/legal-notices/OpenDS.LICENSE
 * or https://OpenDS.dev.java.net/OpenDS.LICENSE.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at
 * trunk/opends/resource/legal-notices/OpenDS.LICENSE.  If applicable,
 * add the following below this CDDL HEADER, with the fields enclosed
 * by brackets "[]" replaced with your own identifying * information:
 *      Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
```

- All classes (regardless of their visibility) must have class-level Javadoc documentation.

- All non-private methods must have Javadoc documentation. Although it is not required by Checkstyle, it is strongly recommended that all but the most simple private methods also have Javadoc documentation.

- All public and protected fields must have Javadoc documentation.
- All Javadoc elements must have a period at the end of their first sentence, and there must not be any mismatched HTML elements (e.g., a "`<B>`" element without the corresponding "`</B>`").
- No source line may exceed 80 characters in length.
- No tabs may be used for indentation -- only spaces will be allowed. Further, although Checkstyle does not enforce it, we have agreed that two-space indents should be used.
- No source line may end with whitespace.
- The == operator must not be used to determine whether two strings are the same. The `equals()` method must be used instead.
- Any class that implements the `equals()` method must also implement the `hashCode()` method.
- Code that may fall through `case` elements in `switch` statements will not be allowed.
- All constant `long` values must use a capital "`L`" rather than a lowercase "`l`" because the lowercase character is too easily confused with the numeral "`1`".
- Redundant imports (i.e., importing the same class twice in the same file) will not be allowed.
- Unused imports (i.e., importing a class that is not actually used in that source file) will not be allowed.
- Imports of anything in the `sun.*` package (or any of its sub-packages) will not be allowed.
- Empty statements (i.e., those that just contain a semicolon) will not be allowed.

Further, more stringent requirements have been defined for a subset of the code that has been identified as being more likely for inclusion in documentation, or that is more likely to be referenced by individuals developing plugins or other extensions to the code. This includes code in the `org.opends.server.api`, `org.opends.server.protocols.internal`, and `org.opends.server.types` packages (and their sub-packages). The additional constraints that will be enforced for this code include:

- All methods (including private methods) must have Javadoc documentation.
- No source line may exceed 70 characters in length (which makes it much easier to copy and paste this code into documentation without reformatting).

Any style-related elements not mentioned here (e.g., position of curly braces, spacing around commas and parentheses, etc.) will be left to the preference of the developer writing the code, particularly when developing new code. When modifying existing code, it is recommended that the style of the original developer be used so that code will maintain some level of consistency to anyone that might be reading it.

Additional Subversion Client Configuration

For developers that wish to be active in the OpenDS development process, it may be necessary to make additional changes to the configuration of the Subversion client. These changes will not be required for individuals that are merely interested in checking out the code for read-only purposes.

Ensuring Correct End-of-Line Behavior

The most significant of these changes is around the management of the end-of-line characters. Because Windows systems use a different line termination sequence ("\r\n") than virtually every other platform ("\n"), these differences can be frequently visible in editors on different platforms. In particular, code that is developed on Windows systems might appear in a UNIX editor with "`^M`" characters at the end of each line, or code developed on UNIX systems might appear "run together" when viewed in some Windows applications. To deal with this, Subversion makes it possible to define a property (named "`svn:eol-style`") that can be applied to files to specify the end-of-line sequence that should be used when checking them out of the repository. Assigning the "`native`" value for this property indicates that the end-of-line sequence used should be the default for the client platform. That is, checking out a file with this property set on a UNIX system will get a file with lines ending with "\n", while the same file checked out on Windows will have lines that end with "\r\n".

Although it is possible to manually update each text file so that it contains this property, this can be a hassle to manage and it is easy to forget. Fortunately, it is possible to configure the Subversion client to automatically set this property when the files are added to the repository. This can be done by editing the `~/.subversion/config` file on UNIX systems or the `%APPDATA%\Subversion\config` file on Windows. The following changes should be made in that file:

1. In the "`[miscellany]`" section, change the value of the "`enable-auto-props`" property to "`yes`". This will make it possible for the client to automatically assign certain properties to the file when they are added to the repository.

2. In the "`[auto-props]`" section, add the following lines to ensure that the end-of-line sequence will always be correct for the underlying platform:
```
*.java = svn:eol-style=native
*.sh = svn:eol-style=native
*.bat = svn:eol-style=native
*.ldif = svn:eol-style=native
*.txt = svn:eol-style=native
*.xml = svn:eol-style=native
*.html = svn:eol-style=native
```

Common Development and Distribution License, Version 1.0

Unless otherwise noted, all components of the OpenDS Directory Server, including all source, configuration, and documentation, are released under the Common Development and Distribution License (CDDL), Version 1.0. The full text of that license is as follows:

1. Definitions.

1.1. "Contributor" means each individual or entity that creates or contributes to the creation of Modifications.

1.2. "Contributor Version" means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.

1.3. "Covered Software" means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof.

1.4. "Executable" means the Covered Software in any form other than Source Code.

1.5. "Initial Developer" means the individual or entity that first makes Original Software available under this License.

1.6. "Larger Work" means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.

1.7. "License" means this document.

1.8. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications" means the Source Code and Executable form of any of the following:

A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;

B. Any new file that contains any part of the Original Software or previous Modifications; or

C. Any new file that is contributed or otherwise made available under the terms of this License.

1.10. "Original Software" means the Source Code and Executable form of computer software code that is originally released under this License.

1.11. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.12. "Source Code" means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.

1.13. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants.

2.1. The Initial Developer Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).

(c) The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.

2.2. Contributor Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of:

(1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Availability of Source Code.

Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

3.2. Modifications.

The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

3.3. Required Notices.

You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

3.4. Application of Additional Terms.

You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients' rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.5. Distribution of Executable Versions.

You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipient's rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.6. Larger Works.

You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

4. Versions of the License.

4.1. New Versions.

Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

4.2. Effect of New Versions.

You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

4.3. Modified Versions.

When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

5. DISCLAIMER OF WARRANTY.

COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY

COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

6. TERMINATION.

6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as "Participant") alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

6.3. In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.

7. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOST PROFITS, LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

8. U.S. GOVERNMENT END USERS.

The Covered Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" (as that term is defined at 48 C.F.R. 252.227-7014(a)(1)) and "commercial computer software documentation" as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or

provision that addresses Government rights in computer software under this License.

9. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdiction's conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

10. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)

For Covered Software in this distribution, this License shall be governed by the laws of the State of California (excluding conflict-of-law provisions).

Any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California and the state courts of the State of California, with venue lying in Santa Clara County, California.