OpenDS

Synchronization Architecture

Version 0.1
July 2006

Contents

Overview

This document attempt to describe the general architecture of the synchronization feature of OpenDS.
Synchronization design is still in progress and this architecture is therefore still evolving and should not be considered as set in stone.

Copyright and trademark information

Feedback

Please direct any comments or suggestions about this document to:
gilles.bellaton@sun.com

Acknowledgments

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

Modifications and Updates

Date	*Description of Change*
July 20th 2006	Initial draft

Architecture Overview

The synchronization is built around a centralized publish/subscribe architecture, each Directory Server is configured to communicate with this central service and uses it to publish its own changes and be notified about the changes from the other servers. This central service is called the changelog service.

The changelog service can be configured as a highly-available service by using multiple instance that can run on multiple hosts. The instances providing the changelog service are called the changelog servers.

At startup time, each Directory Server choose one instance of the changelog server that it will use both for sending all its changes and receiving all the changes from the other servers.

All the changelog servers are always connected to each other changelog servers.

The Directory Servers sends all their change to the changelog server with which they are connected. When they receive a change from a directory server, the changelog servers forward this change to all the other changelog servers and to all the servers to which they are connected. When they receive a change from another changelog server, the changelog servers forward to change to the directory servers but not to the changelog servers.

The Directory Servers never send their changes directly to other Directory Servers.

This make sure that all changes are forwarded to all servers without requiring complex negociations.

Each change is assigned a change number by the Directory Server that originally processed this change. This change number is used to identify the change during all its processing.

The changelog servers keeps the changes in stable storage so that the older changes can be kept and resend to directory server in case they were not connected when the change happened or in case they get late and changes cannot be sent to them at the time they are processed.

Each Directory Server updatedness is kept by using a record of the last changes that they have processed. When connecting to a changelog server they use it to let the changelog server know from which change it should start sending updates to this Directory Server.

Because multiple Directory Servers instances can be updated simultaneously, it is possible that conflicting operations happens between server. The conflicts are solved by each Directory Server when it replays operation from other Directory Server in a way that makes sure that all Directory Servers eventually converge to the same data.
Conflict can happen between modify operations , this is called modify conflict or between add, delete and modrdn operations, this is called naming conflicts.
To be able to solve the conflicts in a coherent way the Directory Servers needs to keep an history of the successive changes that happened on each entry. This is called the historical information, it is stored as an operational attribute inside the entry itself.

The Change Numbers

The change numbers are used to uniquely identify the changes and to define a consistent order between the change. This order will be used by the conflict resolution procedures and when forwarding the changes.
The change numbers are therefore constituted with

- A time stamp in msec
- A sequence number
- A server identifier

The time stamps is generally generated based on the system clock, however, it is also necessary that each server make sure that it always generate change numbers that are greater than all the change numbers that it has already processed so that operations depending on other operations (like a modify after an add) are always replayed after the operation it is depending on.

The sequence number is a sequential number that is incremented for each change inside the same msec

The replica identifier is an integer identifier uniquely assigned to each server in the topology. It is useful to make sure that 2 servers don't assign the same identifier to tow different changes. At the present time it must be assigned by the administrator. In the future we may develop an algorithm to automatically assign a unique identifier to each server.

Sequential Numbers

In many situations it would be useful to have sequential numbers for the changes. Those sequential numbers have to be generated toward the end of the operation because it must be done at a time when the operation cannot be rejected and cannot failed anymore so that there will never be holes in the number sequence.
The Change numbers them self are used to store historical information and this must be done before applying the change to the entries and is therefore incompatible with being sequential.
Therefore an other sequential number will be used to identify the changes.

The Server State

When an ldap server connect to a changelog server, the changelog server must know what is the current state of updatedness of this ldap server so that it can start sending it the changes that it has not seen yet.

This state of updatedness must be kept as a vector because a server may have not seen relatively old changes from another server that is distant but may have seen more recent changes from a close-by server and may have processed very recent changes.
It is therefore kept by recording the last processed change number from each server identifier.

because servers can be stopped and restarted it is necessary to save this server state to stable storage.
Ideally this should be done after completion of each change (local or synchronized change).
However saving this information to the database after each change would add too much overhead, this information is therefore kept in memory and saved to the database on a regular basis only and when the server shutdowns properly. The drawback of this is that after a brutal interruption the server state can miss dome of the changes that have already been processed and may need to be fixed on restart. This process will be explain in the "LDAP server crash" part of this document.

Operation dependencies

Before replaying some operation it may be necessary to wait for other previous operation to be fully completed. For example, in the case of an add operation followed by a modify operation of the same entry, it is necessary to wait for the completion of the add operation before starting the modify operation. In such case, the master that processed the original change needs to generate some dependency information that the servers that replay the change will use to know what are the conditions for replaying a change.

The dependency information is the list of the change numbers of the operation on which the current operation is dependent. It is generated on the server that process the change from the application and transmitted to the servers replaying the operation as part of the changes.

Historical information and conflict resolution

There are two types of conflicts : modify conflicts and naming conflicts.

Modify conflicts are conflicts that happen between several modify operations done on different servers. For those conflicts the conflict resolution roles is to make sure that the processing done all the servers ends with the same values everywhere. Those types of conflicts can always be solved automatically.

Naming conflicts are conflicts that happen between other operations or between a modify and an other type of operations. Those types of conflicts are more complex than the modify conflicts because they cannot always be resolved automatically.
In this case the role of the conflict resolution code is first to make sure that the operations do apply to the correct entry everywhere even when the entries are renamed on one server while being modified on another server, but also to allow detection of the cases that cannot be resolved automatically.

Historical information for modify conflicts

When modify conflict happen, it is necessary to be able to decide if the current values in the entry must be kept or if the modification must be applied, therefore the current values of the attribute in the entries are not sufficient it is also necessary to know when (at which Change Number) they were added. Therefore the historical information

- Last date when the attribute was deleted.
- Last date when a given value was added.
- Last date when a given value was deleted.

Whenever an attribute is deleted or fully replaced, all the older information is not relevant anymore and can therefore be removed.

Historical information for naming conflicts

Since LDAP entries can be renamed the DN is not an immutable value of the entry and therefore cannot be used by the synchronization to identify the entry. An unique and immutable identifier is therefore generated at the entry creation time and added as an operational attribute in the entry. It is used as the entry identifier (instead of the DN) in the changes that are sent between the LDAP servers and the changelog servers.
The entryUUIDplugin is used to generate this immutable identifier.

When naming conflict occurs (like for example adds of two different entries on two different servers with the same DN at the same time), the synchronization needs to decide with operation is going to win and therefore needs to have the date when the DN was modified. The following information needs to be kept :

- Last date when a given DN was used for this entry.

Purging the historical information

The historical information is stored in the database and therefore waste some space, IO bandwidth and reduce cache efficiency. The historical information kept for each change can be removed as soon as more recent changes have been seen from each of other other servers.
This can happen

- When a new change is done on the same entry.
- By a purge process that can be triggered at regular interval. This would save space but at the cost of CPU processing and therefore must be configurable.

Processing a Change on a master

When a modification, is done by an application on the original master, the synchronization code needs to assign a change number, generate the historical information, forward the change to a changelog server and update its server state.

Because it is saved in the entry itself, the historical information must be set in the operation before writing to the backend. The change number is used when generating the historical information and this must therefore be done before generating the historical information. These two processing are therefore done in the pre-operation phase.

The send of the operation to the changelog server must be done before sending back the acknowledgment of the operation to the LDAP application so that a synchronous or an assured mode can be implemented. This is therefore done in the post-operation phase.

It is important to send the changes in the order that is defined by the change numbers so that the changelog servers can make sure that all the changes are forwarded to the other LDAP servers. Since the directory server is multi-threaded it is possible that the post-operation plugins are not called in the same order as the pre-operation phase. The synchronization code therefore needs to maintain a list of pending changes which contain the list of changes that have started and for which change numbers have already been generated but that have not yet been sent to the changelog server. The changes are added to this list in the pre-operation phase and removed from this list when they are sent to the changelog server. If a given operation reaches the post-operation phase ahead of its natural position then it will wait for send of the other previous operations before it is send to the changelog server.

The server state is updated at the same time the operation is sent to the changelog server.

Replaying changes on the LDAP servers

The replay of changes on the LDAP servers needs to be efficient on multi-code and multi CPU machines otherwise the synchronization mechanism would not be able to keep LDAP servers on synchronization on those types of platforms.
Several threads are used to read the changes sent by the changelog server.
The dependency information is then used to decide if the operation can start being replayed. This is done by comparing the list of operations it is dependent on and check with the server state if these operations have already been replayed. If not, the operation is put in a special queue containing the dependency operations.
If the operation can be replayed, an internal operation is built from the information sent by the changelog servers and run.
Those internal operations can be in conflict with previous operations and therefore cannot always be replayed as they were played on the original server. This is checked in a specific phase of the operation processing called the handleConflictResolution phase. In the huge majority of cases the operations will not be conflicting and there will be nothing to do in the handleConflictResolution phase and the code is therefore optimized for those cases. If this is not the case, appropriate actions must be taken to solve the conflicts. In case of a modify conflict, the handleConflictResolution change the modifications to kept the last changes. In case of naming conflict, the handleConflictResolution code may have to rename this entry or another entry. This last case is not fully investigated yet and it is not clear how to archive this in an atomic way.
Once the conflict resolution is done, the historical information must be updated in the same way as for local operations and the operation can then proceed in the core server.
At the end of the operation the server state is updated.
After completing an operation the thread needs to check if some operation in the dependency queue was waiting for tis operation to be completed and can now be replayed, if yes it starts the replay process for this operation, if not
start listening again to operations from the changelog server.

The Changelog Servers

The roles of the changelog servers are :

- Manage connection from ldap servers.
- Connect to other changelog servers, and listen for connections from other changelog servers.
- Receive changes from ldap servers.
- Forward changes to ldap server and other changelog servers.
- Save changes to stable storage (includes trimming of older operations).

The Changelog servers are not Directory Servers per see but like Directory Servers they need to have a configuration file, they need to have on-line configuration and monitoring, they need to have backup and restore facilities. Developing a separate daemon would require to develop all those features for the changelog servers. Instead it is much more handy to develop the changelog server as a plugin of the directory server daemon so that we can benefit from all the administration facilities.

Therefore the changelog servers should always also be LDAP servers or JMX servers even though they do not store LDAP data.

Making configuration simple

The LDAP servers need to know :

- The list of suffix-dn that must be synchronized
- For each base-dn the list of all the changelog servers that it can connect to

The changelog servers need to know :

- The list of all the other changelog servers

It would be user un-friendly to require to configure all these informations on each server because this would mean that adding a changelog server would require to change the configuration of all the other changelog servers and also on some of the LDAP servers.

To avoid this the servers needs to exchange topology information :
when adding a new LDAP or changelog server the administrator only needs to configure one or two changelog servers
when the new server connects to a changelog server, the changelog server sends back the list of all changelog server and the new server adds this list to its configuration.
In case several data center are used and the administrator wants the LDAP servers to connect only to the changelog server, a geography identifier can be set to identify the servers that stay in the same data-center. When they connect the LDAP servers tries to connect to the servers in their geography first.

Choosing the changelog server

When its starts (or when the changelog server that it is connected to stops) the LDAP servers need to choose an adequate changelog server for publishing and receiving changes. Two parameters must be used for choosing the best changelog server : the geography parameter and the updatedness state of the changelog server.
LDAP servers tries to find a changelog server that's match the following criteria :

1 : any changelog servers that stay in the same geography and has already seen all the changes that it has seen.
2 : any changelog servers that stay in the same geography.
3 : any changelog servers that has already seen all the changes that it has seen.
4 : any changelog server.

If they have to connect to servers that have not seen all the changes that they have seen, it is necessary to update the changelog servers with those changes. The LDAP servers can achieve this by browsing the historical informations and reconstructing fake operations from this state information.

Auto-repair

Despite all the efforts to keep servers synchronized it may happen that servers start to show incoherent data. Typically this can happen because of :

- A disk error caused the stored data to be tainted.
- A memory error caused error while handling some data.
- a software bug caused bad data or leaded to missing changes.
- etc...

In such case, the tracking and replay of changes alone is not sufficient for re-synchronization of those incoherent data.
The synchronization therefore provide an automatic repair mechanism that can leverage the historical information inside the entries to decide what the data should be and that can repair it in the servers where it is broken or missing.
This auto repair mechanism is implemented as an ldap application running from the same hosts as the changelog servers.
It can be started in several modes :

- repair an inconsistency that was raised by an error while replaying modifications
- repair an inconsistency that was detected by the administrator
- periodically scan the entries to detect and repair inconsistencies

Directory server crash

When a directory server crash, the connection to the changelog server is lost and it is possible that some of the changes that the directory server has processed and committed to the database have not yet been transmitted to any changelog server.

Therefore when it restart the directory servers needs to compare its state with the state of the changelog servers and if they detect that some changes are missing on the changelog servers it needs to create fake operations from the historical information and send those operations to the changelog servers that it has chosen.

Since the local server state is not saved after each operation, the directory server cannot trust the saved copy of the server state after a crash and therefore needs to recalculate it using the historical information.

Changelog Server crash

When a changelog server crash, the Directory Servers must connect to another changelog server, check if they are missing some of their changes and if necessary resend them.

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.