OpenDS

Directory Server

JE Backend Database Design

Version 1.0
July 2006

Contents

Overview

Through its backend database API, the OpenDS Directory Server can support multiple types of repositories to store LDAP entry data. A default backend implementation was required that would be suitable for the vast majority of uses, from those that contain only a handful of entries, up to those containing hundreds of millions of entries. It has to be very efficient and highly scalable, yet be easy to manage. This document describes the JE Backend, a backend implementation which uses the Berkeley DB Java Edition embedded storage engine to satisfy those requirements.

Copyright and trademark information

Feedback

Please direct any comments or suggestions about this document to:
issues@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*
07/20/06	Initial version.

Terminology

The Berkeley DB notion of a database is somewhat different to common usage. When we talk about a Directory Server backend database for example, we are referring to all the data in that backend instance. However, a database in Berkeley DB Java Edition is a single collection of keyed records, usually in b-tree form.

base DN – The DN of the entry in the backend that is closest to the root entry, for those entries in a backend instance sharing a common base entry. There may be multiple base DNs in a backend instance.

JE – Berkeley DB Java Edition.

JE backend – The primary Directory Server backend database built on JE.

JE database – A collection of records, where each record is a key/data pairing.

JE environment – A collection of JE databases, sharing a common cache. Transactions are created within the environment.

key comparator – A function, used for JE b-tree key comparisons, which determines the key order.

Data Organization

Each instance of this backend creates a single JE environment to store its data so that JE environments are not shared by backend instances. The backend supports multiple base DNs, so it is possible for data under multiple suffixes to share the same database environment, by declaring those suffixes as base DNs of a single JE backend instance.

The data for a base DN is kept in its own set of databases, so that each JE database contains data for only one base DN. Each JE database name is prefixed by the base DN it belongs to, where the DN is simplified by preserving only letters and digits.

For example, if you were to use the `com.sleepycat.je.util.DbDump` utility to list the databases in the environment corresponding to a backend instance containing just the base DN "dc=example,dc=com", you might see the following:

```
dc_example_dc_com_cn.equality
dc_example_dc_com_cn.presence
dc_example_dc_com_cn.substring
dc_example_dc_com_dn2id
dc_example_dc_com_givenName.equality
dc_example_dc_com_givenName.presence
dc_example_dc_com_givenName.substring
dc_example_dc_com_id2children
dc_example_dc_com_id2entry
dc_example_dc_com_id2subtree
dc_example_dc_com_mail.equality
dc_example_dc_com_mail.presence
dc_example_dc_com_mail.substring
dc_example_dc_com_member.equality
dc_example_dc_com_sn.equality
dc_example_dc_com_sn.presence
dc_example_dc_com_sn.substring
dc_example_dc_com_telephoneNumber.equality
dc_example_dc_com_telephoneNumber.presence
dc_example_dc_com_telephoneNumber.substring
dc_example_dc_com_uid.equality
```

Database Relocation

The data is stored in a format which is independent of system architecture, and is also independent of file system location because it contains no pathnames. The backend, and its backups, can be copied, moved and restored to a different location, within the same system or a different system.

The Entry ID

Each entry to be stored in the backend is assigned a 64-bit integer identifier called the entry ID. The first entry to be created is entry ID 1, the second is entry ID 2, etc. This ensures that the ID for any given entry is always greater than its superiors. The backend takes care to preserve this invariant, in particular during Modify DN operations where an entry can be given a new superior. Clients have come to expect child entries to be returned after their parent in search results, and the backend can ensure this by returning entries in ID order.

On disk, an entry ID is stored in eight bytes in big-endian format (from most significant byte to least significant byte). This facilitates binary copy of the backend from one system to another, regardless of the system architecture.

The IDs of deleted entries are not reused. The use of a 64-bit integer means it is implausible that the entry ID space will be exhausted.

The Entry Database

Entries are stored in a JE database referred to internally as id2entry. The key to the database is the entry ID, and the value is an ASN.1 encoding of the entry contents. The default JE b-tree key comparator is used for the entry database, such that a cursor will return entries in order of entry ID. When the backend starts it is able to determine the last assigned entry ID by reading the last key value in the entry database.

The format of the entry on disk is described by the following ASN.1.

```
DatabaseEntry ::= [APPLICATION 0] IMPLICIT SEQUENCE {
 uncompressedSize INTEGER,      -- A zero value means not compressed.
 dataBytes        OCTET STRING  -- Optionally compressed encoding of
                                   the data bytes.
}

ID2EntryValue ::= DatabaseEntry
 -- Where dataBytes contains an encoding of DirectoryServerEntry.

DirectoryServerEntry ::= [APPLICATION 1] IMPLICIT SEQUENCE {
 dn                       LDAPDN,
 objectClasses            SET OF LDAPString,
 userAttributes           AttributeList,
 operationalAttributes    AttributeList
}
```

Entry compression is optional and can be switched on or off at any time. Switching on entry compression only affects future writes, therefore the database can contain a mixture of compressed and not-compressed records. Either record type can be read regardless of the configuration setting. The compression algorithm is the default ZLIB implementation provided by the Java platform.

The ASN.1 types have application tags to allow for future extensions. The types may be extended with additional fields where this makes sense, or additional types may be defined.

Note that all entry attributes, great and small, are lumped together in the same record. This is inefficient when one or two small attributes are modified in an entry containing a large binary attribute value (such as a photo for example), since the entire entry record must be written. A design improvement could be to break out “large” attributes into their own database.

The entry count record

It is useful to be able to provide to the administrator the current number of entries stored in the backend. JE does not maintain database record counts, therefore requiring a full key traversal to count the number of records in a database, which is too time consuming for large numbers of entries. For this reason the backend maintains its own count of the number of entries in the entry database, storing this count in the special record whose key is entry ID zero.

The DN Database

Although each entry's DN is stored in the entry database, we need to be able to retrieve entries by DN. The DN database key is the normalized DN and the value is the entry ID corresponding to the DN. A normalized DN is one which may be compared for equality with another using a standard string comparison function. A given DN can have numerous string representations, due to insignificant white-space, or insignificant case of attribute names, etc., but it has only one normalized form. Use of the normalized form enables efficient key comparison.

A custom b-tree key comparator is applied to the DN database, which orders the keys such that a given entry DN comes after the DNs of its superiors, and ensures that the DNs below a given base DN are contiguous. This ordering is used to return entries for a non-indexed subtree or single level search. The comparator is just like the default lexicographic comparator except that it compares in reverse byte order.

For example, a cursor iteration through a range of the DN database might look like this:

```
dc=example,dc=com
ou=people,dc=example,dc=com
uid=user.1000,ou=people,dc=example,dc=com
uid=user.2000,ou=people,dc=example,dc=com
uid=user.3000,ou=people,dc=example,dc=com
uid=user.4000,ou=people,dc=example,dc=com
uid=user.100,ou=people,dc=example,dc=com
uid=user.1100,ou=people,dc=example,dc=com
uid=user.2100,ou=people,dc=example,dc=com
```

At first, it may seem strange that “user.1100” comes after “user.1000” but it becomes clear when considering the values in reverse byte order, since “0011.resu” indeed comes after “0001.resu”.

Index Databases

Index databases are used to efficiently process search requests. The children and subtree system indexes are dedicated to processing one-level and subtree search scope respectively. Additional attribute indexes are configured as needed to process components of a search filter. Each record in an index database maps a key to an Entry ID List.

Entry ID List

An entry ID list is a set of entry IDs, arranged in order of ID. On disk, the list is a concatenation of the 8-byte entry ID values, where the first ID is the lowest. The number of IDs in the list can be obtained by dividing the total number of bytes by eight.

Index Entry Limit

In some cases, the number of entries indexed by a given key is so large that the cost of maintaining the list during entry updates outweighs the benefit of the list during search processing. Each index therefore has an entry limit. Whenever a list reaches the entry limit, it is replaced with a zero length value to indicate that the list is no longer maintained.

Children Index

The children index is a system index which maps the ID of any non-leaf entry to entry IDs of the immediate children of the entry. This index is used to get the set of entries within the scope of a one-level search.

Subtree Index

The subtree index is a system index which maps the ID of any non-leaf entry to entry IDs of all descendants of the entry. This index is used to get the set of entries within the scope of a subtree search.

Attribute Equality Index

An attribute equality index maps the value of an attribute to entry IDs of all entries containing that attribute value. The database key is the attribute value after it has been normalized by the equality matching rule for that attribute. This index is used to get the set of entries matching an equality filter.

Attribute Presence Index

An attribute presence index contains a single record of entry IDs of all entries containing a value of the attribute. This index is used to get the set of entries matching an attribute presence filter.

Attribute Substring Index

An attribute substring index maps a substring of an attribute value to entry IDs of all entries containing that substring in one or more of its values of the attribute. This index is used to get a set of entries that are candidates for matching a substring filter.

The length of substrings in the index may be configured. For example, let's say the configured substring length is six (the default), and there is an entry containing the attribute value EXAMPLE. The ID for this entry would be indexed by the substrings of length six (EXAMPLE, XAMPLE), plus the remaining trailing substrings (AMPLE, MPLE, PLE, LE and E). To find entries containing a short substring such as AM, cursor through all keys with prefix AM, and those are exactly the required entries. To find entries containing a longer substring such as EXAMPLES, read keys EXAMPL, XAMPLE, AMPLES. In this case the resulting set will also include any entries containing those substrings in the wrong order.

Attribute Ordering Index

An attribute ordering index is similar to an equality index in that it maps the value of an attribute to entry IDs of all entries containing that attribute value. However, the values are normalized by the ordering matching rule for the attribute rather than the equality matching rule, and the b-tree key comparator is set to the ordering matching rule comparator. This index is used to get the set of entries matching inequality filters (less-than-or-equal, greater-than-or-equal).

Search Evaluation

To process an LDAP search operation, the JE backend first applies the search filter to the attribute indexes to obtain an initial set of candidate entry IDs. Then the candidates may be further refined by fetching subordinates of the base entry from either the children or subtree databases (depending on the search scope).

If a candidate set could be obtained, the search is deemed “indexed”. Each candidate entry is fetched from the entry database and returned to the client if it matches the search scope and filter.

If no candidate set could be obtained (due to a lack of indexes or some of the index values having exceeded the index entry limit), the search is deemed “not-indexed”. In this case, a cursor is opened on the DN database at the base entry, to iterate through the DN/ID records, fetching and filtering the corresponding entries, until all the entries under the search base have been processed.

Whenever the number of candidate entry IDs from the indexes is found to be 10 or less, no further attempt is made to reduce the number of candidates. Instead those entries are immediately fetched from the entry database and filtered, on the assumption that this will be quicker than continuing to read the index databases. This could pay off for “AND” search filters in which the first component is the most specific.

Search “AND” filters are also rearranged so that components that are slow to evaluate (greater-or-equal, less-or-equal) come after components that are generally faster (equality, etc.).

The Referral Database

The referral database is used to provide support for Named Subordinate References, described in RFC 3296. The database contains all the URIs from referral entries conforming to the specification. The key is the DN of the referral entry and the value is that contained in a labeled URI in the `ref` attribute for that entry. Duplicate keys are permitted in this database since a referral entry can contain multiple values of the `ref` attribute. Key order is the same as in the DN database so that all referrals in a subtree can be retrieved using a cursor through a range of the records.

The referral database is used during processing of LDAP search operations to return all applicable search result referrals, and it is used during processing of all LDAP operations when the target entry cannot be found, to return a referral if a referral entry is found at or above the target entry.

Caching

The JE backend makes the appropriate callbacks to the Directory Server entry cache API (see `org.opends.server.api.EntryCache`), so that an entry cache configured for the Directory Server will be used.

Other than that, JE backend performance depends largely on the JE environment cache. The JE environment cache is configured to use a certain amount of the Directory Server JVM heap, by specifying it either as a percentage of the available JVM heap, or the number of bytes.

The JE backend can be configured to preload the JE cache when it is started by the Directory Server. A configuration attribute specifies the length of time the JE backend may spend populating the cache, and the backend attempts to prioritize the data to get the most benefit from the preload time. The entry database comes first, followed by the DN database, then the other databases.

Configuration

Because of the number of configuration options that are available for this backend, its configuration is represented as a tree rather than a single entry:

```
cn=config
      cn=Backends (Container for all backends)
            ds-cfg-backend-id=userRoot (A backend instance)
                  cn=Index (Container for attribute index definitions)
                        ds-cfg-index-attribute=cn (An attribute index)
                        ...
                  cn=JE Database (Berkeley DB JE configuration options)
```

The `ds-cfg-backend-id=userRoot` entry includes the location on disk of the backend database (the JE environment), and the base DNs stored in the backend.

The `cn=JE Database` entry contains a selection of the most useful properties supported by Berkeley DB Java Edition, so that it can be configured and tuned through the Directory Server mechanisms.

In LDIF the configuration looks like this:

```
dn: ds-cfg-backend-id=userRoot, cn=Backends, cn=config
objectClass: top
objectClass: ds-cfg-backend
objectClass: ds-cfg-je-backend
ds-cfg-backend-enabled: true
ds-cfg-backend-class: org.opends.server.backends.jeb.BackendImpl
ds-cfg-backend-id: userRoot
ds-cfg-backend-writability-mode: enabled
ds-cfg-backend-base-dn: dc=example,dc=com
ds-cfg-backend-directory: db
ds-cfg-backend-index-entry-limit: 4000
...

dn: cn=Index, ds-cfg-backend-id=userRoot, cn=Backends, cn=config
objectClass: top
objectClass: ds-cfg-branch
cn: Index

dn: ds-cfg-index-attribute=cn, cn=Index, ds-cfg-backend-id=userRoot,
cn=Backends, cn=config
objectClass: top
objectClass: ds-cfg-je-index
ds-cfg-index-attribute: cn
ds-cfg-index-type: presence
ds-cfg-index-type: equality
```

```
ds-cfg-index-type: substring
ds-cfg-index-entry-limit: 4000

...

dn: cn=JE Database, ds-cfg-backend-id=userRoot, cn=Backends, cn=config
objectClass: top
objectClass: ds-cfg-je-database
cn: JE Database
ds-cfg-database-cache-percent: 10
ds-cfg-database-cache-size: 0 MB
...
```

Import and Export

If an import is not appending to existing data, a non-transactional, non-locking, deferred-write JE environment handle is opened to get the fastest performance. To recover from an import that fails part way through, the import can be simply restarted from the beginning.

The import process is multi-threaded. A foreman thread does the following:

1. Reads the next entry from the input file,
2. Checks the DN database to ensure the entry does not already exist.
3. Checks the DN database to ensure the parent entry exists,
4. Assigns a new entry ID,
5. Writes a new record to the DN database,
6. Places the entry on a queue.

A number of worker threads are polling the queue. The worker thread:

1. Reads an entry from the queue,
2. Writes the entry to the entry database,
3. Writes index data for that entry to temporary files.

The temporary index files are binary files containing a series of records, ordered by key, where each record contains:

1. The number of bytes in the key,
2. The key,
3. The number of bytes of IDs to be inserted,
4. The IDs to be inserted, a series of long integers ordered by ID.

Each index has its own set of files. The worker thread buffers up as much data as it can in the memory it has been allocated, before sorting the keys, merging IDs for a given key, and writing a new temporary file. There is no need to sort the list of IDs since each entry is new and already in order of ID.

When all entries have been processed, and optionally whenever a specified number of entries have been processed, the temporary index files are merged into the final database files. To merge a set of files, the files are first opened and one record is read from each file. Data for the lowest key is merged into the database, the next record from each contributing file is read, and this is repeated until there is no more input. This process was designed to write the JE database files efficiently in key order, and with each record rewritten as few times as possible.

Import in append mode

If the option is set to allow existing entries to be replaced and the foreman finds that an entry exists upon checking the DN database, then the foreman:

1. Reads the existing entry ID from the DN database,
2. Reads the existing entry contents from the entry database,
3. Places the new entry contents on a queue, with the existing ID and contents as attachments.

Also, the temporary index file records are extended:

1. The number of bytes in the key,
2. The key,
3. The number of bytes of IDs to be inserted,
4. The IDs to be inserted, a series of long integers ordered by ID.
5. The number of bytes of IDs to be removed.
6. The IDs to be removed, a series of long integers ordered by ID.

Since we are dealing with both existing and new data, IDs are no longer collected in order of ID, so each list of IDs must be sorted when writing a temporary file.

This design of append-mode requires a backup of the existing data in case the import fails part way through. Even though the environment is written in transactional mode, the resulting database files may be left in an inconsistent state, which would not be fixed by restarting the import from the beginning. A better append-mode import and recovery process is for further study.

Export to LDIF

To export entries, a read-only database environment is opened and then a single thread uses a cursor to traverse the entry database to obtain all the entries. Opening a read-only environment results in a temporary snapshot of the database at that time.

A multi-threaded export is for further study. Note that LDIF files are generally expected to satisfy the constraint that a parent entry comes before its children. A requirement is that the output of an export must be valid input to an import.

Backup and Restore

The JE backend supports both full and incremental backup. The backend may be in use, even for writes, while a backup is taken. A full backup consists of a compressed archive of all the transaction log files in the environment. An incremental backup consists of a compressed archive of just those transaction log files that have been written since the previous backup, together with a list of names of log files that are unchanged since the previous backup.

The extra properties that are required to be stored in the backup info file are:

> `last-logfile-name` – The name of the latest (highest numbered) log file at the time the backup was created.
>
> `last-logfile-size` – The size in bytes of the latest log file at the time the backup was created.

Since JE will never rewrite a log file numbered less than (i.e. written before) the last log file, there is no need to know anything about those earlier files when taking an incremental backup.

When writing an incremental backup, if it is discovered that the database cleaner has deleted a log file between the time that the listing of log files is obtained and the time that the log file is to be copied into the archive, a new listing of log files must be obtained. This ensures that live data from the deleted log file is captured.

Source code organization

The source for the JE backend is in the package:

`org.opends.server.backends.jeb`

Some of the more significant classes are:

- `BackupManager` – Backup and restore.
- `EntryContainer` – Backend operations on the collection of entries in a single base DN.
- `ImportJob` – Import entries from LDIF.
- `ConfigurableEnvironment` – Implementation of `org.opends.server.api.ConfigurableComponent` for a JE environment.
- `BackendImpl` – The implementation of `org.opends.server.api.backend` and `org.opends.server.api.ConfigurableComponent` for the JE backend.
- `VerifyJob` – Verify the integrity of the backend data.
- `IndexFilter` – Applies a search filter to the indexes to generate a set of candidate entry IDs.
- `ExportJob` – Export entries to LDIF.
- `JebFormat` – Methods to serialize and deserialize data to and from the database.
- `Config` – The JE backend configuration.

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