

LDAP Innovations in the OpenDS project

Ludovic Poitou
OpenDS Community Manager
Sun Microsystems
ludovic.poitou@sun.com

Introduction

The OpenDS project was initiated about 3 years ago, with the goal of building a complete set of LDAPv3 directory services. While the focus with OpenDS 1.0 was to get a running server that implemented most of the LDAP standards and experimental RFCs, the OpenDS 2.0 release introduced some concepts and features new to LDAP servers. After a brief overview of the OpenDS project, we present two innovations that are aimed at the directory administrators : Scheduled and Recurrent Tasks and Assured Replication, and some new features in syntaxes and matching rules allowing application developers to get more of their OpenDS directory server.

The OpenDS project

The OpenDS project is an open source project initiated by Sun in July 2006, with the goal to build a complete new generation of directory services, based on LDAP and DSML. The project is written entirely in Java, and available under the CDDL license at <https://opends.dev.java.net/>. The OpenDS server is designed with 3 principles : Ease of Use, Extensibility and Performance. The first stable version, released in July 2008, delivered a complete LDAPv3 directory server and a DSMLv2 gateway, installable in less than 3 minutes through a Java Web Start install wizard.

Released in July 2009, OpenDS 2.0 improves the ease of use, the performance and scalability and introduced the new Assured Replication mode as well as Scheduled and Recurrent Tasks. The coming release, OpenDS 2.2, planned to be available in October adds another set of functionalities, including new LDAP extensibility features like new syntaxes and matching rules described here after.

Innovations for Directory Administrators

Assured Replication

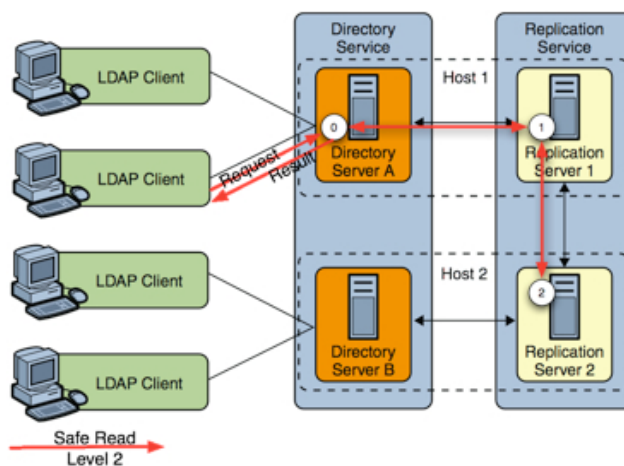
The OpenDS replication model is based on a loose consistency model where changes are applied on a server, acknowledged to the client and then replicated to all the replicas in the topology. OpenDS supports multi-master replication and applies the same conflict resolution procedures on all replicas so eventually the data is consistent on every host. This is the same model that is used by Sun Directory Server, 389 Directory Server (formerly known as Fedora Directory Server) and

has been successfully deployed and used by customers for years.

Assured Replication is an extension to this multi-master replication model to bring a tighter consistency of the data between replicas. When Assured Replication is enabled, the response to the client application is delayed until the LDAP update has been received or applied by other servers, in a best effort mode. This is not to be mistaken for a full synchronous and transactional replication mechanism, as it just ensures that the change is copied to other machines before sending the response to the client application. In the case of the server or the machine crashes in the middle of the commit, or if there is a very unlikely database file corruption, the change is present and will be applied to the other servers. To put it simple, it ensures there is no data loss.

Assured Replication can function in 2 modes :

Safe Data Mode: an update must be propagated to a defined number of Replication Servers before returning a response to the client. So if the server or the replication server is stopped, the data is still available to all other replicas. The figure on the right shows the interaction between the Directory services and replication services when a update request is received by the Directory Server A.



Safe Read Mode: an update must be propagated to all directory servers in the domain before the client is returned a response for the update.

In both modes, it's possible to configure a timeout interval to prevent LDAP clients to be waiting indefinitely if some servers are not available.

From a client point of view, there should be no difference, except that the server might take a little longer to return the response to an update request. In our measures, we found that the response time increased by 25% for Safe Data Level 2, which seems a lot, but honestly, when the response time is in the order of 2ms, it's hard to notice !

Scheduled and Recurrent Tasks

Being a Directory Server administrator often implies that you have to perform some administrative tasks on a regular basis. For example, one of the tasks that an administrator has to do is a regular backup of the database. With most directory servers, the administrator would write a script to be run on a specific time of the day (or rather the night) that would proceed with the backup.

In OpenDS, it is possible to initiate a backup, a restore, an export to LDIF or an import from LDIF by creating a specific entry under the cn=tasks naming context. Both the management

command line utilities and the graphical Control Panel make use of tasks. A task can be initiated immediately or can be scheduled for later, so the administrator does not need to stay in front of his computer to run a backup at midnight.

With OpenDS 2.0, we've added the possibility to define a repeated schedule when creating the task. A recurrent task will be execute on schedule until the task entry is deleted from the server. Recurrent tasks are handy but one can argue they are not necessary as the same function can be scripted out of the server. This is true. However there are some benefits with tasks and especially recurrent tasks. The recurrent tasks are run whenever the server is running. Which means that there is no error if it isn't, and if the server is restarted, the tasks are rescheduled automatically. Being defined in the OpenDS directory server, the tasks are backup with the rest of the data, and can be restored on the same server instance or on other instances. The recurrent backup is the obvious use for the recurrent tasks but with a little imagination, one can see a lot of advantages, especially for services in the cloud.

Innovations for the application developers

New Syntaxes

Defining new syntaxes for a Directory Server is usually very complex as it involves providing the code to handle and validate the values.

OpenDS has a couple of Syntax Factories that allow to define new types of syntaxes, for a better control of the values: A Regex Syntax factory and an Enum Syntax factory.

Those two are responding to frequent customers requests for a better control of the values provided by client applications. In the past, one would need to specifically write a server extension to be able to provide the functionality.

The Regex Syntax

To define a new syntax with values respecting specific pattern, one would simply add the syntax to the current OpenDS schema. This is done by simply adding a value to the cn=schema entry :

```
dn: cn=schema
changetype: modify
add: ldapsyntaxes
ldapSyntaxes: ( 1.3.6.1.4.1.32473.1 DESC 'Host and Port in the format of
HOST:PORT'
  X-PATTERN '^ [a-zA-Z][a-zA-Z0-9-]+:[0-9]+$' )
```

And then you can define attributes making use of the syntax, and object classes for the newly defined attributes.

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.4.1.32473.2 NAME 'test-attr-regex' SYNTAX
  1.3.6.1.4.1.32473.1 )
```

```

-
add: objectclasses
objectclasses: ( 1.3.6.1.4.1.32473.3 NAME 'testOCregex' SUP top AUXILIARY
MUST
  test-attr-regex)
-

```

Now when adding or modifying values of the test-attr-regex attributes, the server will reject values that are not matching the regular expression.

The Enum Syntax

In a similar way to the Regex syntax, it's possible to define syntaxes which allow an enumerated list of values.

```

dn: cn=schema
changetype: modify
add: ldapsyntaxes
ldapSyntaxes: ( 1.3.6.1.4.1.32473.4 DESC 'Day Of The Week'
  X-ENUM ( 'monday' 'tuesday' 'wednesday' 'thursday'
  'friday' 'saturday' 'sunday' ) )

```

```

dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.4.1.32473.5 NAME 'test-attr-enum' SYNTAX
  1.3.6.1.4.1.32473.4 )
-
add: objectclasses
objectclasses: ( 1.3.6.1.4.1.32473.6 NAME 'testOCenum' SUP top AUXILIARY
MUST test-attr-enum)

```

With the enum syntax, any value not part of the list will be rejected. Values are not case sensitive. And there is a default ordering implied which mean it's possible to do filters like this: (test-attr-enum>=friday)

New Matching Rules

The I18N Collation Rules

In LDAP, most of the data is made of Directory Strings which are UTF-8 encoded strings. LDAPv3 specifications and more precisely RFC 4518, defines the way to prepare UTF-8 strings to be compared in LDAP and OpenDS is fully compliant with this RFC.

But the case insensitive rules are generic and do not take into account language specific rules when comparing characters. A typical example is in French, the characters “e” and “é” do match equal when compared in a case insensitive way.

In Europe, many people's name contain non ascii characters, Øystein, Graß or Hélène... When a person from another country search for a colleague in the corporate directory, it may not know the exact spelling nor be able to easily compose the characters. So most of the searches are done

with plain ascii. I'm sure similar examples can be found in Asian languages as well.

OpenDS supports an extensive set of locale specific case ignore matching rules (they are one hundred and eight locale), allowing users to match entries according to a specific language. Each locale has been assigned an OID and then there are 6 different matching rules per locale : LowerThan, LowerOrEqual, Equality, GreaterOrEqual, GreaterThan , Substring.

So if one would like to match all entries with the givenname “Hélène” for equality according to the French collation rules, the filter would be the following:

```
(givenname:1.3.6.1.4.1.42.2.27.9.4.76.1.3:=Helene)
```

And to match by Substring:

```
(givenname:1.3.6.1.4.1.42.2.27.9.4.76.1.6:=hel*)
```

But remembering OID for each locale and type of matching is not easy. So we've also provided some shortcuts in the form of the locale name and a short string representing the different matching; lt, lte, eq, gte, gt, sub

And the examples above can also be expressed as:

```
(givenname:fr.eq:=Helene)
(givenname:fr.sub:=hel*)
```

The Relative Time Matching Rules

With attributes that have a GeneralizedTime, it's possible to search for all values that are greater than a specific value. So client applications can issue a search such as (modifyTimeStamp>=200909211200001Z). To compare with the current time, the client application has to translate the current time into a GTM string and put it in the filter. And it has to repeat the computation for every request.

Having to compute the current time to express it in a filter prevents such attributes to be used to express access control decisions (especially if those attributes are frequently modified).

So we've defined two matching rules: relativeTimeLTOrderingMatch and relativeTimeGTOrderingMatch as below:

```
( 1.3.6.1.4.1.26027.1.4.6
  NAME ( 'relativeTimeLTOrderingMatch' 'relativeTimeOrderingMatch.lt' )
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )

( 1.3.6.1.4.1.26027.1.4.5
  NAME ( 'relativeTimeGTOrderingMatch' 'relativeTimeOrderingMatch.gt' )
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

The syntax applies to attributes with a GeneralizedTime syntax but does not take a generalized time string. Instead it takes an Offset in the format of [+|-]Number[Unit] where

Number is a positive integer and Unit is a single letter chosen among s,m,h,d,w, respectively for Seconds, Minutes, Hours, Days and Weeks.

When processing the filter, the server computes the current GMT time, adds the Offset and compares the attribute value with the new computed value.

So a positive offset computes a Time in the future compared to the current time and a negative offset a time in the past compared to the current time.

(pwdExpirationTime:1.3.6.1.4.1.26027.1.4.5:=5d) can be translated in pwdExpirationTime >= (Now + 5 days).

(pwdExpirationTime:1.3.6.1.4.1.26027.1.4.6:=5d) can be translated in pwdExpirationTime <= (Now + 5 days).

The true power of these matching rules are really when used in Access controls or in scripts that are repeated regularly.

The Partial Date Or Time Matching Rule

Another issue when dealing with the GeneralizedTime attributes, is the inability to do a “Substring” match on the date.

The typical example is the birthdate attribute. How can I search for all entries whose birthdate is today, i.e. September 21 ?

In OpenDS, we added a new MatchingRule to be able to provide the answer of such request.

```
( 1.3.6.1.4.1.26027.1.4.7 NAME 'partialDateAndTimeMatchingRule' SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

The matching rule applies to attributes with a GeneralizedTime syntax but the value is not a generalized time. It's a pattern for the date, composed of one or more sequences of Integer followed by a Tag. The currently supported tags are Y, M, D, h, m, s .

Examples:

The following filter will match all users who were born on September 21st.

```
(birthDate:1.3.6.1.4.1.26027.1.4.7:=09M21D)
```

The following filter will match all users who were born in 1965.

```
(birthDate:1.3.6.1.4.1.26027.1.4.7:=1965Y)
```

Conclusion

The hype on LDAP is long gone, the directory servers are becoming commodities, but there is still development and innovation going on in the various products and projects. OpenDS is one of the most active LDAP related open source project, trying to change the directory landscape,

making servers easier to install and manage, more powerful than before, reducing the total cost of ownership of running directory services. You can download or install with Java Web Start the OpenDS server from <http://www.openss.org/>.