Network Working Group
Request for Comments: 2589
Category: Standards Track

Y. Yaacovi
Microsoft
M. Wahl
Innosoft International, Inc.
T. Genovese
Microsoft

May 1999

Lightweight Directory Access Protocol (v3): Extensions for Dynamic Directory Services

Status of this Memo

Copyright Notice

1. Abstract

This document defines the requirements for dynamic directory services and specifies the format of request and response extended operations for supporting client-server interoperation in a dynamic directories environment.

The Lightweight Directory Access Protocol (LDAP) [1] supports lightweight access to static directory services, allowing relatively fast search and update access. Static directory services store information about people that persists in its accuracy and value over a long period of time.

Dynamic directory services are different in that they store information that only persists in its accuracy and value when it is being periodically refreshed. This information is stored as dynamic entries in the directory. A typical use will be a client or a person that is either online - in which case it has an entry in the directory, or is offline - in which case its entry disappears from the directory. Though the protocol operations and attributes used by dynamic directory services are similar to the ones used for static directory services, clients that store dynamic information in the directory need to periodically refresh this information, in order to prevent it from disappearing. If dynamic entries are not refreshed

within a given timeout, they will be removed from the directory. For example, this will happen if the client that set them goes offline.

A flow control mechanism from the server is also described that allows a server to inform clients how often they should refresh their presence.

2. Requirements

The protocol extensions must allow accessing dynamic information in a directory in a standard LDAP manner, to allow clients to access static and dynamic information in the same way.

By definition, dynamic entries are not persistent and clients may go away gracefully or not. The proposed extensions must offer a way for a server to tell if entries are still valid, and to do this in a way that is scalable. There also must be a mechanism for clients to reestablish their entry with the server.

There must be a way for clients to find out, in a standard LDAP manner, if servers support the dynamic extensions.

Finally, to allow clients to broadly use the dynamic extensions, the extensions need to be registered as standard LDAP extended operations.

3. Description of Approach

The Lightweight Directory Access Protocol (LDAP) [1] permits additional operation requests and responses to be added to the protocol. This proposal takes advantage of these to support directories which contain dynamic information in a manner which is fully integrated with LDAP.

The approach described in this proposal defines dynamic entries in order to allow implementing directories with dynamic information. An implementation of dynamic directories, must be able to support dynamic directory entries.

3.1. Dynamic Entries and the dynamicObject object class

A dynamic entry is an object in the directory tree which has a time-to-live associated with it. This time-to-live is set when the entry is created. The time-to-live is automatically decremented, and when it expires the dynamic entry disappears. By invoking the refresh extended operation (defined below) to re-set the time-to-live, a client can cause the entry to remain present a while longer.

A dynamic entry is created by including the objectClass value given in section 5 in the list of attributes when adding an entry. This method is subject to standard access control restrictions.

The extended operation covered here, allows a client to refresh a dynamic entry by invoking, at intervals, refresh operations containing that entry's name. Dynamic entries will be treated the same as non-dynamic entries when processing search, compare, add, delete, modify and modifyDN operations. However if clients stop sending refresh operations for an entry, then the server will automatically and without notification remove that entry from the directory. This removal will be treated the same as if the entry had been deleted by an LDAP protocol operation.

There is no way to change a static entry into a dynamic one and vice-versa. If the client is using Modify with an objectClass of dynamicObject on a static entry, the server must return a service error either "objectClassModsProhibited" (if the server does not allow objectClass modifications at all) or "objectClassViolation" (if the server does allow objectClass modifications in general).

A dynamic entry may be removed by the client using the delete operation. This operation will be subject to access control restrictions.

A non-dynamic entry cannot be added subordinate to a dynamic entry: the server must return an appropriate update or service error if this is attempted.

The support of dynamic attributes of an otherwise static object, are outside the scope of this document.

3.2 Dynamic meetings (conferences)

The way dynamicObject is defined, it has a time-to-live associated with it, and that's about it. Though the most common dynamic object is a person object, there is no specific type associated with the dynamicObject as defined here. By the use of the dynamic object's attributes, one can make this object represent practically anything.

Specifically, Meetings (conferences) can be represented by dynamic objects. While full-featured meeting support requires special semantics and handling by the server (and is not in the scope of this document), the extensions described here, provide basic meetings support. A meeting object can be refreshed by the meeting participants, and when it is not, the meeting entry disappears. The one meeting type that is naturally supported by the dynamic extensions is creator-owned meeting.

3.2.1 Creator-owned meetings

Creator-owned meetings are created by a client that sets the time-to-live attribute for the entry, and it is this client's responsibility to refresh the meeting entry, so that it will not disappear. Others might join the meeting, by modifying the appropriate attribute, but they are not allowed to refresh the entry. When the client that created the entry goes away, it can delete the meeting entry, or it might disappear when its time-to-live expires. This is consistent with the common model for dynamicObject as described above.

4. Protocol Additions

4.1 Refresh Request

Refresh is a protocol operation sent by a client to tell the server that the client is still alive and the dynamic directory entry is still accurate and valuable. The client sends a Refresh request periodically based on the value of the client refresh period (CRP). The server can request that the client change this value. As long as the server receives a Refresh request within the timeout period, the directory entry is guaranteed to persist on the server. Client implementers should be aware that since the intervening network between the client and server is unreliable, a Refresh request packet may be delayed or lost while in transit. If this occurs, the entry may disappear, and the client will need to detect this and re-add the entry.

A client may request this operation by transmitting an LDAP PDU containing an ExtendedRequest. An LDAP ExtendedRequest is defined as follows:

```
ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
        requestName             [0] LDAPOID,
        requestValue            [1] OCTET STRING OPTIONAL }
```

The requestName field must be set to the string "1.3.6.1.4.1.1466.101.119.1".

The requestValue field will contain as a value the DER-encoding of the following ASN.1 data type:

```
SEQUENCE {
        entryName  [0] LDAPDN,
        requestTtl [1] INTEGER
}
```

The entryName field is the UTF-8 string representation of the name of the dynamic entry [3]. This entry must already exist.

The requestTtl is a time in seconds (between 1 and 31557600) that the client requests that the entry exists in the directory before being automatically removed. Servers are not required to accept this value and might return a different TTL value to the client. Clients must be able to use this server-dictated value as their CRP.

4.2 Refresh Response

If a server implements this extension, then when the request is made it will return an LDAP PDU containing an ExtendedResponse. An LDAP ExtendedResponse is defined as follows:

```
ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
        COMPONENTS OF LDAPResult,
        responseName     [10] LDAPOID OPTIONAL,
        response         [11] OCTET STRING OPTIONAL }
```

The responseName field contains the same string as that present in the request.

The response field will contain as a value the DER-encoding of the following ASN.1 data type:

```
SEQUENCE {
        responseTtl [1] INTEGER
}
```

The responseTtl field is the time in seconds which the server chooses to have as the time-to-live field for that entry. It must not be any smaller than that which the client requested, and it may be larger. However, to allow servers to maintain a relatively accurate directory, and to prevent clients from abusing the dynamic extensions, servers are permitted to shorten a client-requested time-to-live value, down to a minimum of 86400 seconds (one day).

If the operation was successful, the errorCode field in the standardResponse part of an ExtendedResponse will be set to success.

In case of an error, the responseTtl field will have the value 0, and the errorCode field will contain an appropriate value, as follows: If the entry named by entryName could not be located, the errorCode field will contain "noSuchObject". If the entry is not dynamic, the errorCode field will contain "objectClassViolation". If the requester does not have permission to refresh the entry, the

errorCode field will contain "insufficientAccessRights". If the requestTtl field is too large, the errorCode field will contain "sizeLimitExceeded".

If a server does not implement this extension, it will return an LDAP PDU containing an ExtendedResponse, which contains only the standardResponse element (the responseName and response elements will be absent). The LDAPResult element will indicate the protocolError result code.

This request is permitted to be invoked when LDAP is carried by a connectionless transport.

When using a connection-oriented transport, there is no requirement that this operation be on the same particular connection as any other. A client may open multiple connections, or close and then reopen a connection.

4.3 X.500/DAP Modify(97)

X.500/DAP servers can map the Refresh request and response operations into the X.500/DAP Modify(97) operation.

5. Schema Additions

All dynamic entries must have the dynamicObject value in their objectClass attribute. This object class is defined as follows (using the ObjectClassDescription notation of [2]):

```
( 1.3.6.1.4.1.1466.101.119.2 NAME 'dynamicObject'
  DESC 'This class, if present in an entry, indicates that this entry
        has a limited lifetime and may disappear automatically when
        its time-to-live has reached 0. There are no mandatory
        attributes of this class, however if the client has not
        supplied a value for the entryTtl attribute, the server will
        provide one.'
  SUP top AUXILIARY )
```

Furthermore, the dynamic entry must have the following operational attribute. It is described using the AttributeTypeDescription notation of [2]:

```
( 1.3.6.1.4.1.1466.101.119.3 NAME 'entryTtl'
  DESC 'This operational attribute is maintained by the server and
        appears to be present in every dynamic entry. The attribute
        is not present when the entry does not contain the
        dynamicObject object class. The value of this attribute is
        the time in seconds that the entry will continue to exist
```

before disappearing from the directory. In the absence of intervening refresh operations, the values returned by reading the attribute in two successive searches are guaranteed to be nonincreasing. The smallest permissible value is 0, indicating that the entry may disappear without warning. The attribute is marked NO-USER-MODIFICATION since it may only be changed using the refresh operation.'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE
NO-USER-MODIFICATION USAGE dSAOperation)

To allow servers to support dynamic entries in only a part of the DIT, the following operational attribute is defined. It is described using the AttributeTypeDescription notation of [2]:

(1.3.6.1.4.1.1466.101.119.4 NAME 'dynamicSubtrees'
DESC 'This operational attribute is maintained by the server and is present in the Root DSE, if the server supports the dynamic extensions described in this memo. The attribute contains a list of all the subtrees in this directory for which the server supports the dynamic extensions.'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 NO-USER-MODIFICATION
USAGE dSAOperation)

6. Client and Server Requirements

6.1 Client Requirements

Clients can find out if a server supports the dynamic extensions by checking the supportedExtension field in the root DSE, to see if the OBJECT IDENTIFIER described in section 4 is present. Since servers may select to support the dynamic extensions in only some of the subtrees of the DIT, clients must check the dynamicSubtrees operational attribute in the root DSE to find out if the dynamic extensions are supported on a specific subtree.

Once a dynamic entry has been created, clients are responsible for invoking the refresh extended operation, in order to keep that entry present in the directory.

Clients must not expect that a dynamic entry will be present in the DIT after it has timed out, however it must not require that the server remove the entry immediately (some servers may only process timing out entries at intervals). If the client wishes to ensure the entry does not exist it should issue a RemoveRequest for that entry.

Initially, a client needs to know how often it should send refresh requests to the server. This value is defined as the CRP (Client Refresh Period) and is set by the server based on the entryTtl.

Since the LDAP AddRequest operation is left unchanged and is not modified in this proposal to return this value, a client must issue a Refresh extended operation immediately after an Add that created a dynamic entry. The Refresh Response will return the CRP (in responseTtl) to the client.

Clients must not issue the refresh request for dynamic entries which they have not created. If an anonymous client attempts to do so, a server is permitted to return insufficientAccessRights (50) in the RefreshResponse, enforcing the client to bind first. Please note that servers which allow anonymous clients to create and refresh dynamic entries will not be able to enforce the above.

Clients should always be ready to handle the case in which their entry timed out. In such a case, the Refresh operation will fail with an error code such as noSuchObject, and the client is expected to re-create its entry.

Clients should be prepared to experience refresh operations failing with protocolError, even though the add and any previous refresh requests succeeded. This might happen if a proxy between the client and the server goes down, and another proxy is used which does not support the Refresh extended operation.

6.2 Server Requirements

Servers are responsible for removing dynamic entries when they time out. Servers are not required to do this immediately.

Servers must enforce the structural rules listed in above section 3.

Servers must ensure that the operational attribute described in section 5 is present in dynamic entries

Servers may permit anonymous users to refresh entries. However, to eliminate the possibility of a malicious use of the Refresh operation, servers may require the refreshing client to bind first. A server implementation can achieve this by presenting ACLs on the entryTtl attribute, and returning insufficientAccessRights (50) in the RefreshResponse, if the client is not allowed to refresh the entry. Doing this, though, might have performance implications on the server and might impact the server’s scalability.

Servers may require that a client which attempts to create a dynamic entry have a remove permission for that entry.

Servers which implement the dynamic extensions must have the OBJECT IDENTIFIER, described above in section 4 for the request and

response, present as a value of the supportedExtension field in the root DSE. They must also have as values in the attributeTypes and objectClasses attributes of their subschema subentries, the AttributeTypeDescription and ObjectClassDescription from section 5.

Servers can limit the support of the dynamic extensions to only some of the subtrees in the DIT. Servers indicate for which subtrees they support the extensions, by specifying the OIDs for the supported subtrees in the dynamicSubtrees attribute described in section 5. If a server supports the dynamic extensions for all naming contexts it holds, the dynamicSubtrees attribute may be absent.

7. Implementation issues

7.1 Storage of dynamic information

Dynamic information is expected to change very often. In addition, Refresh requests are expected to arrive at the server very often. Disk-based databases that static directory services often use are likely inappropriate for storing dynamic information. We recommend that server implementations store dynamic entries in memory sufficient to avoid paging. This is not a requirement.

We expect LDAP servers to be able to store static and dynamic entries. If an LDAP server does not support dynamic entries, it should respond with an error code such as objectClassViolation.

7.2 Client refresh behavior

In some cases, the client might not get a Refresh response. This may happen as a result of a server crash after receiving the Refresh request, the TCP/IP socket timing out in the connection case, or the UDP packet getting lost in the connection-less case.

It is recommended that in such a case, the client will retry the Refresh operation immediately, and if this Refresh request does not get a response as well, it will resort to its original Refresh cycle, i.e. send a Refresh request at its Client Refresh Period (CRP).

7.3 Configuration of refresh times

We recommend that servers will provide administrators with the ability to configure the default client refresh period (CRP), and also a minimum and maximum CRP values. This, together with allowing administrators to request that the server will not change the CRP dynamically, will allow administrators to set CRP values which will enforce a low refresh traffic, or - on the other extreme, an highly up-to-date directory.

8. Replication

Replication is only partially addressed in this memo. There is a separate effort in progress at the IETF on replication of static and dynamic directories.

it is allowed to replicate a dynamic entry or a static entry with dynamic attributes. Since the entryTtl is expressed as a relative time (how many seconds till the entry will expire), replicating it means that the replicated entry will be "off" by the replication time.

9. Localization

The are no localization issues for this extended operation.

10. Security Considerations

Standard LDAP security rules and support apply for the extensions described in this document, and there are no special security issues for these extensions. Please note, though, that servers may permit anonymous clients to refresh entries which they did not create. Servers are also permitted to control a refresh access to an entry by requiring clients to bind before issuing a RefreshRequest. This will have implications on the server performance and scalability.

Also, Care should be taken in making use of information obtained from directory servers that has been supplied by client, as it may now be out of date. In many networks, for example, IP addresses are automatically assigned when a host connects to the network, and may be reassigned if that host later disconnects. An IP address obtained from the directory may no longer be assigned to the host that placed the address in the directory. This issue is not specific to LDAP or dynamic directories.

11. Acknowledgments

Design ideas included in this document are based on those discussed in ASID and other IETF Working Groups.

12. Authors' Addresses

Yoram Yaacovi
Microsoft
One Microsoft way
Redmond, WA 98052
USA

Phone: +1 206-936-9629
EMail: yoramy@microsoft.com

Mark Wahl
Innosoft International, Inc.
8911 Capital of Texas Hwy #4140
Austin, TX 78759
USA

Email: M.Wahl@innosoft.com

Tony Genovese
Microsoft
One Microsoft way
Redmond, WA 98052
USA

Phone: +1 206-703-0852
EMail: tonyg@microsoft.com

13. Bibliography

[1] Wahl, M., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (Version 3)", RFC 2251, December 1997.

[2] Wahl, M. Coulbeck, A., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, December 1997.

[3] Wahl, M. and S. Kille, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, December 1997.

14. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.