ASID Working Group   Y. Yaacovi
INTERNET-DRAFT   Microsoft
                                                               M. Wahl
                                                   Critical Angle Inc.
                                                           T. Genovese
                                                             Microsoft

Expires in six months from   1 July 1997
Intended Category: Standards Track

Lightweight Directory Access Protocol: Dynamic Attributes
<draft-ietf-asid-ldap-dynatt-00.txt>

1. Status of this Memo

2. Abstract

This document defines dynamic attributes and the fashion in which they are being set and used on entries in the directory. This document builds heavily on the dynamic extensions to LDAP infrastructure as defined in [1].

The Lightweight Directory Access Protocol (LDAP) [2] 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.

The dynamic extension to LDAP as defined in [1] added the concept of dynamic entries that only persist in the directory, when they are periodically refreshed. This document takes this approach one step farther and defines how dynamic attributes can be used, on either static or dynamic entries, to handle up-to-date and dynamic information about an entry in the directory. An example use will be a client or a person that has a static entry in the directory and

sometimes goes online, which is reflected in the 'online' attribute for the entry. To specify whether this person is online or offline - an attribute that changes frequently, a client application will have to modify this attribute relatively frequently. The current location of a person is another attribute that may change frequently.

Dynamic attributes must be periodically refreshed. Otherwise, they will disappear from the directory over time.

3. Requirements

Dynamic attributes must be set and used in a standard LDAP manner, to allow clients to access static and dynamic attributes in the same way.

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

Dynamic attributes must build on the same infrastructure that was design into LDAP in the dynamic extensions draft ([1]). It should take advantage of the extensions and operational attributes defined [1].

Finally, to allow clients to broadly use dynamic attributes, they must be able to find out if a server supports such attributes.

4. Description of Approach

The Lightweight Directory Access Protocol (LDAP) [2] supports the use of attributes on entries in the directory. The dynamic extensions draft ([1]) added support for dynamic entries. This proposal takes the attributes model and the dynamic model and merges the two to provide support for dynamic attributes in a manner which is fully compatible with LDAP.

The approach taken here is to mark dynamic attributes as dynamic (described below), and require clients to refresh these attributes periodically in order to guarantee that they will continue to be present on the entry. Dynamic attributes can be set on either a static or dynamic entry, and the time-to-live associated with them applies to ALL the dynamic attributes on an entry, i.e. there is no time-to-live per dynamic attribute. This provides for a more scalable design, and a less complex server implementation.

4.1. Dynamic Attributes

Dynamic attributes behave the same as ordinary user attributes to Search, Compare and Modify requests, except that if they time out they disappear from the entry in which they are located. Unlike the dynamicObject class - which is used to mark entries as dynamic, the entry itself does not disappear, and non-dynamic attributes are unaffected. Like dynamic entries, a static entry with dynamic attributes must have the entryTtl attribute which is described in [1].

Dynamic attributes do not necessarily introduce a new set of attributes in the schema. Any static attribute in the schema can be made dynamic by including the ;dynamic attribute modifier with the typename. It is allowed to introduce new dynamic attributes which are not useful in the static domain. Such attributes will still require including the ;dynamic attribute modifier with the typename to define them as dynamic.

A dynamic attribute may be a subtype of a non-dynamic attribute, but a non-dynamic attribute must not be a subtype of a dynamic attribute. Operational attributes and collective attributes must not be dynamic. All dynamic attributes are considered to be user attributes, however they are not guaranteed to be present in shadow copies of entries.

A client may introduce dynamic attributes into an entry by using a Modify request to add them, or by including them in the attribute list of an Add Request. If the attribute type is permitted in the entry then the dynamic attribute is also permitted. The client would specify that the attribute is dynamic by including the tag ";dynamic" with the typename. Dynamic values may be changed, and the attributes removed, by using the Modify request as normal. If the entry is deleted, dynamic attributes disappear immediately along with all the non-dynamic. A ;dynamic modifier on an attribute must not be used in a compare request, or in a search request (either in the filter or attributes requested for return). This is in keeping with the philosophy that dynamic attributes be indistinguishable from static attributes as much as possible. In such a case, a server is expected to reject the request, returning 'invalidAttributeSyntax'.

The granularity of a dynamic attribute is the entire attribute including all values. Dynamic and static values may not be mixed within a single attribute. For example, suppose an existing static entry had an attribute called "ipAddress" with the value "1.2.3.4". A modification of the entry to add the value "5.4.3.2" to the attribute "ipAddress;dynamic" would fail with a "constraintViolation" error.

The addition and modification of dynamic attributes are subject to schema and access control restrictions, as with non-dynamic attributes. Thus unless the extensibleObject object class is present, generally an object class or content rule must be defined to permit those attributes to be present in an entry. If their presence is controlled by an object class, then just as with non-dynamic attributes, the object class value must have already been added before the attributes are added. The dynamicObject object class described in [1] does not itself permit any particular dynamic attributes.

Dynamic attributes in a particular entry are refreshed using the Refresh extended operation. All of the entry's dynamic attributes are refreshed by a single refresh request. The TTL given in the refresh response applies to all of the entry's dynamic attributes. There is no way to refresh particular dynamic attributes within an entry at different times, or to have different TTLs apply to different dynamic attributes or different values of the same multi-value dynamic attribute.

If not refreshed, all dynamic attributes in an entry time out simultaneously. All the attributes which are dynamic with all their values disappear atomically, as if the server had done a ModifyEntry specifying that all the dynamic types were to be deleted from that entry. The client must not expect any dynamic attributes to be present in an entry after the time-to-live for that entry has reached zero. However the attributes may not disappear immediately as servers may only process timing out attributes at intervals (e.g. every few minutes).

Note that if an object class definition references a dynamic attribute as a mandatory attribute, the attributes of the entry will still time out, but a schema inconsistency will be present in that entry. (The objectClass attribute and its values always remain since objectClass is not a dynamic attribute.) Thus it is strongly recommended that designers not specify dynamic attributes as anything other than optional attributes of auxiliary classes.

Dynamic attributes may be used within dynamic entries (i.e., entries containing object class "dynamicObject", defined in [1]), but since all of such an entry's attributes are implicitly dynamic, such use is superfluous.

5. Protocol Additions

Support of dynamic attributes in LDAP does not require any protocol additions beyond the Refresh request and response which were introduced in [1].

6. Schema Additions

An entry in the directory which has dynamic attributes must have the entryTtl operational attribute. In addition, the dynamicSubtrees attribute, if present in the root DSE, indicates which subtrees of the directory support the dynamic extensions. The entryTtl attribute and the dynamicSubtrees attribute are defined in [1].

The following attributes are introduced as a result of allowing dynamic attributes on static entries. It is described using the AttributeTypeDescription notation of [3]:

```
( 1.3.6.1.4.1.1466.101.119.5 NAME 'dynamicAttributesSubtrees'
  DESC 'This operational attribute is maintained by the server and
        is present in the Root DSE, if the server supports dynamic
        attributes as described in this draft. The attribute contains
        a list of all the subtrees in this directory for which the
        server supports dynamic attributes.'
  SYNTAX 'DN' NO-USER-MODIFICATION USAGE dSAOperation )

( 1.2.840.113556.1.4.612 NAME 'online'
  DESC 'This attribute is used to indicate whether a entry is the
        directory is currently online or offline. This attribute
        should only be used with objects in the directory for which
        being online or offline makes sense. For example, a
        person object or a meeting object. Dynamic entries are
        always expected to be online. This attribute is optional.
  SYNTAX 'BOOLEAN' SINGLE-VALUE )

( 1.2.840.113556.1.4.613 NAME 'onlineServer'
  DESC 'This attribute is used to indicate the DNS names of server
        where this entry is online. This attribute should only
        be used with objects in the directory for which being
        online or offline makes sense. If an entry is not online,
        the server names in this attribute are meaningless. This
        attribute is maintained by the client and is optional.
  SYNTAX 'LDAPString' )
```

Note that the 'online' and 'onlineServer' attributes are only examples for one particular application of dynamic attributes, and that not all entries with dynamic attributes are required to have these attributes.

7. Client and Server Requirements

7.1 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 [1] for Refresh request and response, is present. Farther, and as described in [1], clients are expected to check the dynamicSubtrees operational attribute in the root DSE to find out in which subtrees of the directory, the server selected to support the dynamic extensions.

To find out if a server specifically supports dynamic attributes, and in which subtrees in the directory, clients must check the dynamicAttributesSubtrees operational attribute in the root DSE.

Once an entry has been created in the directory, that has dynamic attributes, clients are responsible for invoking the refresh extended operation, in order to keep these attributes present in the entry. The same holds true for dynamic attributes that were added to an entry after its creation.

Clients must not expect that a dynamic attributes will be present in an entry after they have timed out. However it must not either require that the server remove these attributes immediately (some servers may only process timing out attributes at intervals). If the client wishes to ensure an attribute does not exist it should issue a ModifyRequest to remove this attribute explicitly.

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 AddRequest and ModifyRequest are left unchanged and are not modified in this proposal to return this value, a client must issue a Refresh extended operation immediately after an Add or Modify that introduced a dynamic attribute to an entry. The Refresh Response will return the CRP (in responseTtl) to the client as described in [1].

Clients must not issue the refresh request for entries to which they did not add dynamic attributes. Please note that when a client refreshes a static entry to which it added dynamic attribute(s), it refreshes ALL the dynamic attributes in this entry, including ones added by other clients. Also note that servers which allow anonymous clients to create and refresh dynamic entries and attributes will not be able to enforce the above.

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

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.

7.2 Server Requirements

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

Servers must enforce the schema rules listed in above section 4.

Servers must ensure that the entryTtl operational attribute described in [1] is present in entries that contain dynamic attributes.

Servers are permitted to check the authentication of the client invoking a refresh extended operation, and only permit the operation if it matches that of the client which created the entry or added dynamic attributes to this entry (please see a note about that in section 7.1). Servers may permit anonymous users to refresh entries.

Servers which implement support for dynamic attributes must have the OBJECT IDENTIFIER, described in [1] 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 attribute of their subschema subentries, the AttributeTypeDescription for entryTtl and dynamicSubtrees, as described in [1].

8. Implementation issues

8.1 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 attributes 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. Such a server might still support setting dynamic attributes on a static entry.

8.2 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).

9. Localization

The are no localization issues for this extended operation.

10. Security Considerations

Security issues are not addressed in this document. Please note, though, that anonymous clients are able to refresh attributes which they did not add to an entry.

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
Critical Angle Inc.
4815 W. Braker Lane #502-385
Austin, TX 78759
USA

EMail: M.Wahl@critical-angle.com

Tony Genovese
Microsoft
One Microsoft way
Redmond, WA 98052
USA

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

13. Bibliography

[1] Y.Yaacovi, M.Wahl, T.Genovese, "Lightweight Directory Access Protocol (v3): Extensions for Dynamic Directory Services". INTERNET DRAFT. <draft-ietf-asid-ldapv3-dynamic-04.txt>

[2] M.Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (Version 3)". INTERNET DRAFT <draft-ietf-asid-ldapv3-protocol-01.txt>

[3] M.Wahl, A.Coulbeck, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions". INTERNET DRAFT <draft-ietf-asid-ldapv3-attributes-04.txt>

Expires on six months from the post date (see top).