INTERNET-DRAFT
<draft-armijo-ldap-treedelete-02.txt>
August 24, 2000
Expires February, 2001

Michael P. Armijo
Microsoft Corporation

Tree Delete Control
draft-armijo-ldap-treedelete-02.txt

1. Status of this Memo

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Distribution of this memo is unlimited. It is filed as <draft-armijo-ldap-treedelete-02.txt>, and expires on February 24, 2001. Please send comments to the authors.

2. Abstract

This document defines an LDAPv3 control that deletes an entire subtree of a container entry. This control extends the scope of the LDAPv3 delete operation as defined in RFC 2251. This control is beneficial in extending the functionality of the LDAP protocol and may be useful in administration in an LDAP environment.

3. RFC Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

4. Tree Delete Control

The control is included in the DelRequest message as part of the controls field of the LDAPMessage. The controlType is " 1.2.840.113556.1.4.805", the criticality field may be TRUE or FALSE, and the controlValue field is absent.

This control allows a client to delete an entire subtree. This can only be done if the authenticated user has appropriate permissions to complete the operation. The server MUST check to see if the authenticated user has appropriate permissions to delete the object and all of its descendants. This control MUST only be used with a DelRequest message. A server MUST ignore the control if used with any other message unless the criticality field is set to True, in which case the entire operation MUST fail and MUST instead return the resultCode unsupportedCriticalExtension as per section 4.1.12 of [RFC 2251]. The server MUST list that it recognizes this control in the supportedControl attribute in the root DSE.

4.1 Tree Delete Semantics

The tree delete request may be processed as a single atomic transaction or as a sequence of atomic transactions. In the latter case, all the client-visible states produced by tree delete are consistent with the execution of a sequence of normal delete requests. The server SHOULD NOT leave process the objects in a manner that could orphan objects.

If a tree delete request fails for any transient reason, it can be retried and, if the retry runs to a successful conclusion, the result is the same as if the initial request had succeeded. The client SHOULD resubmit the delRequest until the server returns a result code of success(0).

If a client submits a request to move an object into (or create an object in) or out of the subtree concurrently with the execution of a tree delete request, the effect is undefined. The server is NOT required to lock out such requests.

The server MUST NOT chase referrals stored in the tree. If information about referrals is stored in this section of the tree, this pointer will be deleted.

4.2 Error Messages with this Control

When the Tree Delete Control is invoked, the server MUST check to see if the authenticated user has appropriate permissions to delete the object and all of its descendants. If the user does not have appropriate permissions, an insufficientAccessRights(50) error SHOULD

be returned.

The server MUST identify all the objects to be deleted before deletions begin. If the server has a problem identifying the objects to delete, the server MAY return an operationsError(1). The operation MAY be retried if this error is returned. If the server is not the authority for any objects in the selected tree the server SHOULD return unwillingToPerform(53).

Server implementations may have other restraints on which containers may or may not use the Tree Delete control. If you attempt to delete a container that cannot be deleted due to a platform specific restraint, the server MUST return the error unwillingToPerform(53). The Tree Delete control will not work under these circumstances and the operation SHOULD NOT be retried on this container.

If the limit to the number of objects that can be deleted in one operation is reached, the server SHOULD return adminLimitExceeded(11). Objects processed up to the point of the limit SHOULD be deleted. The DelRequest with the Tree Delete Control SHOULD be resubmitted until a successful response is returned to the server.

5. References

[RFC 2251]
M. Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997. 1997.

[RFC 2119]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels,"
RFC 2119, Harvard University, March 1997.

6. Authors Address

Michael P. Armijo
One Microsoft Way
Redmond, WA
98052
USA

(425)882-8080
micharm@microsoft.com

Expires February 24,2001