ASID Working Group
INTERNET-DRAFT

S. Judd
Microsoft Corporation
S. Thatte
Microsoft Corporation
P. Leach
Microsoft Corporation
W. Hopkins
Hewlett-Packard Corporation

Expires August 28, 1997

February 28, 1997

Lightweight Directory Access Protocol:
Schema for Storing RPC Entries in a Directory Service
< draft-ietf-asid-ldap-rpcschema-00.txt >

1. Status of this Memo

This document is an Internet-Draft. 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."

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

2. Abstract

The Lightweight Directory Access Protocol [1][2] defines a standard protocol for accessing diverse directory services. One common use of a directory service is the location of application services, among which are RPC services.

This document defines a set of object classes and attributes for storing metadata and binding information for RPC services that are DCE-compliant or closely follow the DCE model, in directories that support LDAP.

3. Table of Contents

4. Overview

This document defines a schema that conforms closely to the DCE conceptual model for RPC entries. This schema allows an RPC Name Service Interface (NSI) implementation to use LDAP to store RPC entries, and to use LDAP queries to implement the RPC Name Service Interface (NSI) lookup APIs.

Note on terminology: in this document the terms "property" and "attribute" are used interchangeably.

The requirement is to support three kinds of RPC Name Service Entries:

@ Server Entries: server entries support the retrieval of a set of string bindings for any combination of Entry Name, Interface ID and version, Object ID, Transfer Syntax, and transfer syntax version
@ Group Entries: a set of RPC entries identified by an Entry name
@ Profile Entries: a profile establishes a priority-based search order through a set of entries. This is essentially a "list of sets" with the outer list ordered by priority and each inner set being a set of entries at the current priority

DCE RPC defines the concept of a "mixed entry" in which a single entry serves multiple purposes - for example, entries that serve as both Group and Server entries. Mixed entries are not supported by this schema. This seldom-used DCE RPC feature leads to unnecessary complexity for both implementers and users of the RPC Name Service Interface.

To meet these requirements the schema defines seven object

```
classes:
@ container
@ rpcEntry
@ rpcGroup
```

@ rpcServer
@ rpcServerElement
@ rpcProfile
@ rpcProfileElement

and 9 attributes:

@ rpcNsObjectID
@ rpcNsGroup
@ rpcNsPriority
@ rpcNsProfileEntry
@ rpcNsInterrfaceId
@ rpcNsAnnotation
@ rpcNsCodeset
@ rpcNsBindings
@ rpcNsTransferSyntax

Taken together these object classes and attributes implement the DCE-RPC concept of an "Entry".

The rpcEntry object class is an abstract class from which all other RPC objects derive, so that they may be easily located in a search.

An rpcGroup, rpcServer, or rpcProfile object forms the "root" of an entry. The type of entry is determined by the object class. Note that the types are mutually exclusive; an entry cannot serve multiple purposes. Separating the entry types into distinct object classes simplifies the task of the NSI provider in determining how to handle a given entry.

Table 1: RPC Entry Types

Entry Type	Object Class(es)
Group	rpcGroup, holds a set of references to other RPC-Entry objects
Profile	rpcProfile, a container holding a set of rpcProfileEntry objects, each holding a list of references to entries with a given priority.
Server	rpcServer, a container holding a set of rpcServerElement objects, each holding the identification of one or more interfaces (and/or objects) offered by a given server

5. Objects and Attributes

The 'top', 'country', 'locality', 'organization' and 'organizationalUnit' object classes are included in the "LDAPv3 Standard and Pilot Attribute Definitions", a work in progress [2].

5.1. Notation

The notation used in this document is the same as that used in [2], with the following difference: the referenced notation does not allow the expression of both permissible parentage and class inheritance. The BNF in [2] for defining object classes is therefore extended as follows:

```
<ObjectClassDescription> ::= "("
    <oid>   -- ObjectClass identifier
    [ "NAME" <DirectoryStrings> ]
    [ "DESC" <DirectoryString> ]
    [ "OBSOLETE" ]
    [ "SUP" <oids> ]    -- ObjectClass[es] from which this
                           class is derived
    [ "PARENT" <oids> ] -- Permissible parents of this
                           object class
    [ ( "ABSTRACT" | "STRUCTURAL" | "AUXILIARY" ) ]
    [ "MUST" <oids> ]   -- AttributeTypes
    [ "MAY" <oids> ]    -- AttributeTypes
")"
```

5.2. Object Naming

All objects have 'cn' (Common-name) as their naming attribute; this attribute provides the RDN for the object. 'cn' is included in [2] and defined in [4].

5.3. Object Definitions

5.3.1. Container

The Container object is a general purpose container for use in building containment hierarchies, and is not specific to RPC.

```
(1.2.840.113556.1.3.23
      NAME 'container'
      SUP top
      PARENT (country $organization $ organizationalUnit $
              locality $ container)
)
```

5.3.2. RPC Entry

The RPC Entry is an abstract class from which all other RPC

classes are derived.

```
(  1.2.840.113556.1.5.27
    NAME 'rpcEntry'
    SUP top
    ABSTRACT
    MUST cn
 )
```

5.3.3. RPC Group

The rpcGroup object defines an RPC Group. The cn is the RDN component of the entry name provided by the user in the NSI API call that creates the group. The rpcNsObjectID attribute contains string UUIDs of objects added to the group entry by applications. These Object IDs are not used by the NSI provider during lookup operations.

```
(  1.2.840.113556.1.5.80
    NAME 'rpcGroup'
    SUP rpcEntry
    PARENT (country $organization $ organizationalUnit $
            locality $ container)
    STRUCTURAL
    MUST rpsNsGroup
    MAY rpcNsObjectID
 )
```

5.3.4. RPC Profile

RPC Profile entries are implemented by two object classes. The rpcProfile object class is a container. It is used to gather profile elements into a single profile instance. The cn is the RDN component of the entry name provided by the user in the NSI API call that creates the profile.

```
(  1.2.840.113556.1.5.82
    NAME 'rpcProfile'
    SUP rpcEntry
    PARENT (country $ organization $ organizationalUnit $
            locality $ container)
    STRUCTURAL
 )
```

The rpcProfileElement object describes a single element in the profile. The entire profile is retrieved with a single-level LDAP search rooted at the parent rpcProfile container. The cn is a string UUID generated by the NSI provider when the

rpcProfileElement instance is created.

```
  (  1.2.840.113556.1.5.26
      NAME 'rpcProfileElement'
      SUP rpcEntry
```

```
      PARENT rpcProfile
      STRUCTURAL
 MUST ( rpcNsPriority $ rpcNsProfileEntry $ rpcNsInterfaceId )
      MAY rpcNsAnnotation
   )
```

5.3.5. RPC Server

RPC Server entries are implemented by two object classes. The rpcServer object class is a container. It is used to gather rpcServerElement entries into a single server instance. The CN is the RDN component of the entry name provided by the user in the NSI API call that creates the server entry.

```
  (  1.2.840.113556.1.5.81
     NAME 'rpcServer'
     SUP rpcEntry
     PARENT (country $ organization $ organizationalUnit $
             locality $ container)
     STRUCTURAL
     MAY ( rpcNsObjectID $ rpcNsCodeSet )
   )
```

The rpcServerElement object describes a single interface in the server entry. The entire Server entry is retrieved with a single-level LDAP search rooted at the parent rpcServer container. The properties of the rpcServerElement object allow for efficient searching using straightforward LDAP query expressions. The CN is a string UUID generated by the NSI provider when the rpcServerElement instance is created.

```
  (  1.2.840.113556.1.5.73
     NAME 'rpcServerElement'
     SUP rpcEntry
     PARENT rpcServer
     STRUCTURAL
 MUST ( rpcNsInterfaceID $ rpcNsBindings $ rpcNsTransferSyntax )
  )
```

5.4. Attribute Definitions

RPC Name Service implementations search on a well-known set of properties.Implementations of this schema are advised to index the following properties for performance reasons:

- rpcNsObjectID

- rpcNsInterfaceID

5.4.1. rpcNsObjectID

A set of string UUIDs for objects (in the DCE RPC sense of objects).

```
(  1.2.840.113556.1.4.312
    NAME 'rpcNsObjectID'
    EQUALITY caseIgnoreListMatch
    SYNTAX directoryString
    USAGE userApplications
)
```

5.4.2. rpcNsGroup

A set of DN's for RPC entries that are members of a given RPC group.

```
(  1.2.840.113556.1.4.114
    NAME 'rpcNsGroup'
    EQUALITY distinguishedNameMatch
    SUBSTRING distinguishedNameMatch
    SYNTAX DN
    USAGE userApplications
)
```

5.4.3. rpcNsPriority

An integer value indicating the priority of a given RPC profile element.

```
(  1.2.840.113556.1.4.117
    NAME 'rpcNsPriority'
    EQUALITY integerMatch
    SYNTAX INTEGER
    USAGE userApplications
)
```

5.4.4. rpcNsProfileEntry

The DN of a single RPC entry that is a member of a given RPC profile.

```
(  1.2.840.113556.1.4.118
    NAME 'rpcNsGroup'
    EQUALITY distinguishedNameMatch
    SUBSTRING distinguishedNameMatch
    SYNTAX DN
    SINGLE-VALUE
```

```
    USAGE userApplications
)
```

5.4.5. rpcNsInterfaceId

A string composed of the UUID for an interface exported by an RPC server,
and the interface major and minor version numbers in the form:
<string-UUID>","<major>"."<minor>.

```
(  1.2.840.113556.1.4.115
    NAME 'rpcNsInterfaceID'
    EQUALITY caseIgnoreMatch
    SYNTAX directoryString
    SINGLE-VALUE
    USAGE userApplications
)
```

5.4.6. rpcNsAnnotation

A string describing a given RPC Profile element.

```
(  1.2.840.113556.1.4.366
    NAME 'rpcNsAnnotation'
    EQUALITY caseIgnoreMatch
    SUBSTRING caseIgnoreMatch
    SYNTAX directoryString
    SINGLE-VALUE
    USAGE userApplications
)
```

5.4.7. rpcNsCodeSet

A set of strings identifying the character sets supported by a given RPC
server.

```
(  1.2.840.113556.1.4.367
    NAME 'rpcNsCodeset'
    EQUALITY caseIgnoreListMatch
    SYNTAX directoryString
    USAGE userApplications
)
```

5.4.8. rpcNsBindings

A set of binding strings for a given interface and transfer syntax, in the
form:
<ProtocolSequence>":"<NetworkAddress>"[]" or
<ProtocolSequence>":"<NetworkHostName>"[]"

```
(  1.2.840.113556.1.4.113
    NAME 'rpcNsBindings'
    EQUALITY caseIgnoreMatch
    SYNTAX directoryString
    USAGE userApplications
)
```

5.4.9. rpcNsTransferSyntax

A set of strings composed of the string UUID for a transfer syntax supported by an RPC server, and the transfer syntax major and minor version numbers in the form:

```
<string-UUID>","<major>"."<minor>.
```

```
(  1.2.840.113556.1.4.314
    NAME 'rpcNsTransferSyntax'
    EQUALITY caseIgnoreListMatch
    SYNTAX directoryString
    USAGE userApplications
)
```

6. Usage Model

Instantiating an rpcGroup, rpcProfile, or RPCServer object and any necessary child objects (e.g. rpcServerElement or rpcProfileElement) can create any RPC entry type. Searching is simplified because there is a well-known set of object classes and attributes for each entry type.

A group entry contains the rpcNsGroup property listing the entries in the group. Each group is a single object and can be retrieved in a single operation. An rpcGroup object may have rpcNsObjectID present - the list of object ID's, if present, are explicitly stored and retrieved by applications and are not used by the NSI provider in locating rpcNsGroup objects.

A profile entry consists of an rpcProfile container with one or more rpcProfileEntry objects as children, one for each priority level defined. The complete profile is retrieved in a single operation by performing a single-level LDAP search for objects of class rpcProfileElement rooted at the rpcProfile entry.

A server entry consists of an rpcServer container with one or more rpcServerElement objects as children. The complete entry is retrieved in a single operation by performing a single-level LDAP search for objects of class rpcServerElement rooted at the rpcProfile entry. The NSI provider creates a new rpcServerElement entry when the interface and transfer syntax

provided by the caller do not match an existing rpcServerElement in the named server entry. If a matching rpcServerElement exists, the NSI provider updates it with the string bindings provided by the caller.

This schema allows many discrete rpcServerElement objects to be stored in a given entry. This avoids a number of problems in trying to store multiple interfaces with their versions and transfer syntaxes in a single entry while providing convenient access and searching via LDAP. Indexing the Interface ID and Object UUID's reduces the performance cost for retrieving multiple objects.

6.1. DCE Relative Names

DCE RPC allows names presented to the RPC Name Service Interface to be absolute or relative. An absolute name contains the full DN of the entry in question. A relative name is relative to the DCE Cell where the name is stored. The full DN is the DN of the cell with the relative name appended. When using an LDAP directory to store RPC entries as defined by this schema, the implementation of relative names is implementation dependent, but should be consistent. A suggested approach is the creation of a container at the root of the namespace (e.g. directly below the first instantiated object, such as Organization or organizationalUnit) called RpcServices, which forms the root for 'cell-relative' names.

7. Authors' Addresses

Steven G. Judd
Microsoft Corporation
1 Microsoft Way
Redmond, WA. 98053
USA

e-mail: stevenju@microsoft.com

Paul Leach
Microsoft Corporation
1 Microsoft Way
Redmond, WA. 98053
USA

e-mail: paulle@microsoft.com

Satish Thatte
Microsoft Corporation
1 Microsoft Way
Redmond, WA. 98053
USA

e-mail: satisht@microsoft.com

William S. Hopkins
Hewlett-Packard Corporation
300 Apollo Drive
Chelmsford, MA. 01824

e-mail: hopkins@apollo.hp.com

8. Bibliography

[1] W. Yeong, T. Howes, S. Kille, 'Lightweight Directory Access Protocol', RFC 1777, March 1995, <URL:ftp://ds.internic.net/rfc/rfc1777.txt>

[2] M. Wahl, T. Howes, S. Kille, et al. "Lightweight Directory Access Protocol (Version 3)", INTERNET-DRAFT <draft-ietf-asid-ldapv3-protocol-04.txt>, February 1997.

[3] M. Wahl, A. Coulbeck, T. Howes, S. Kille, 'Lightweight Directory Access Protocol: Standard and Pilot Attribute Definitions', INTERNET DRAFT <draft-ietf-asid-ldapv3-attributes-03.txt> October 1996

[4] The Directory: Selected Attribute Types. ITU-T Recommendation X.520, 1993.

[5] The Directory: Selected Object Classes. ITU-T Recommendation X.521, 1993.