                Schema Publishing in X.500 Directory

Status of this Memo

   This memo defines an Experimental Protocol for the Internet
   community. This memo does not specify an Internet standard of any
   kind. Discussion and suggestions for improvement are requested.
   Distribution of this memo is unlimited.

Abstract

   The X.500 directory provides a powerful mechanism for storing and
   retrieving information about objects of interest. To interpret the
   information stored in the directory, the schema must be known to all
   the components of the directory. Presently, there are no means other
   than ftp to distribute schema information across the Internet. This
   is proving to be a severe constraint as the Directory is growing.
   This document presents a solution to the schema distribution problem
   using the existing mechanisms of the directory. A naming scheme for
   naming schema objects and a meta-schema for storing schema objects is
   presented. The procedures for fetching unknown schema from the
   directory at runtime are described.

Table of Contents

1. Introduction

   The X.500 Directory [1] is now used for a wide range of applications
   from name/address lookup to network management, from restaurant
   information to bibliographic information services. This information
   is distributed and managed across a network of many autonomous sites.
   In order to interpret the information stored in the directory, the
   components of the directory must have knowledge about the structure
   and representation (schema) of the information held within the
   directory.

   The distributed nature of the network and the relatively slow process
   of standardization have given rise to the challenging task of making
   accessible the information about the schema rules themselves. A
   mechanism for making the schema accessible to the functional
   components of the directory is urgently required.

   The 1993 X.500 Directory Standard [2] has attempted to address the
   problem of schema management and distribution. The 1993 framework
   does provide the means for storing and retrieving schema information
   in the directory. However, the resolution of unknown OIDs will
   require both the DUA and the DSA to be compliant with [2].

   In this document we propose a solution using the existing mechanisms
   of the directory [1] itself. We present a naming scheme for naming
   schema objects and a meta-schema for storing schema objects in the
   directory. The proposal allows the algorithmic resolution of unknown
   objects in the directory and in the absence of 1993 X.500 Directory
   Standard implementations provides an interim solution to the schema
   publishing problem.

2. Schema Management

   The storage and retrieval mechanism provided by the directory is
   powerful and flexible. However, the key to the directory is the
   knowledge of the schema rules defined for the objects represented in
   the directory. To facilitate the diffusion of this knowledge
   appropriate schema management mechanisms need to be designed. Schema
   management involves:

        o Storage of schema information in the directory
        o Algorithmic access to and retrieval of schema information
          in the directory
        o Definition of rules for schema modification
        o Propagation of schema information from one component of the
          directory to other components of directory

In this document we concentrate on the aspect of schema
access/retrieval from the directory. Since schema objects are defined
and employed, the modification , addition and deletion of schema
objects can be carried out using existing directory mechanisms. But
the operational issue of synchronizing the schema with the DIB will
require further attention. Similarly the issue of schema propagation
requires further work and is outside the scope of this document. The
strategy proposed in this document has a very simple and workable
approach. No added DAP/DSP functionality is envisaged. At the same
time by using the directory's distributed framework scalability
problems are avoided. In essence, it allows the distributed storage
of schema objects and proposes a naming scheme which allows
algorithmic schema retrieval. Of course, on the down side, more than
one directory read operation may be required to retrieve the
information about an object and its attributes, as objects and
attributes are stored as separate entries in the directory.

As schema information of all objects in a naming context are stored
below the root entry of that naming context, the same DSA will be
able to supply the schema information stored in that DSA. Thus there
is no need to contact another DSA for resolving the schema of an
object stored in the local DSA.

3. Storage of Schema Information in the Directory

The schema information may be stored and distributed using mechanisms
external to the X.500 directory standard [5]. This document proposes
storing schema information in the directory. It has the following
advantages:

     o The components of the directory can access the schema
       information using the standard directory protocols.

     o The nature of the directory naturally allows the schema
       to be distributed. Schema used locally can be kept in the
       local DSA itself whereas schema for general objects like
       person, organization etc can be made available to all
       components of the directory by publishing it.

In the operational model, the schema information in the directory is
expected to complement the schema information held in central
repositories.

3.1 Naming Scheme for the Schema

   The schema information is stored in a distributed manner. We propose
   a model in which each naming context stores the schema relevant to
   it.

```
                            Root
                               \
                                \
             +------------\--------------------+
             |          C=IN         DSA-1     |
             |           / \                   |
             |          /   \                  |
             |         /     \                 |
             |        /       \                |
             |       /      cn=subschema       |
             |      /       / / | \ \ \        |
             |     /       / / | \ \ \         |
             |    /       oid= oid=            |
             +--/-----------------------------+
               /
              /
   +---------------------/--------------------+
   |          o=IIT, Madras      DSA-2        |
   |             / \                          |
   |            /   \                         |
   |           /     \                        |
   |          /       \                       |
   |       ou=CSE      cn=subschema           |
   |       / \         / /| \ \ \             |
   |      /   \       / / | \ \ \             |
   |ipni=spark  cn=Rajeev oid=ipni  oid=      |
   +------------------------------------------+
```
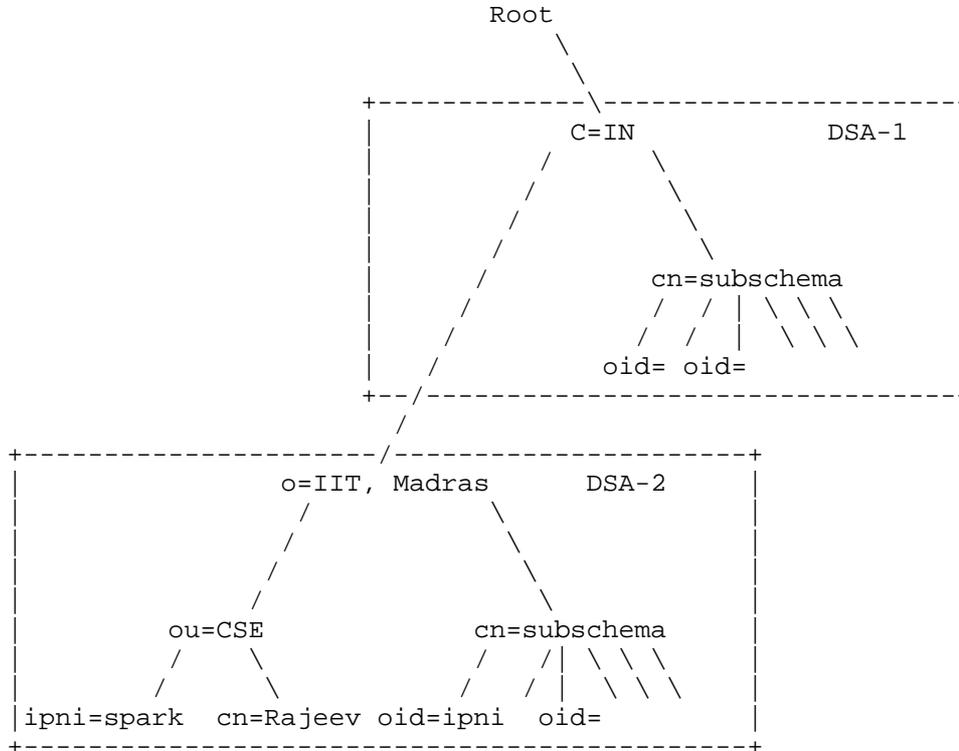
              Figure 1: DIT with schema objects

   To store the schema information, an object called subschema object is
   defined. This object can come anywhere in the Directory Information
   Tree (DIT). The subschema is defined as a subclass of Top. The
   subschema entry is stored below the root entry of a naming context.
   The root entry of a naming context must contain a subschema subentry,
   named {CN= Subschema}. This standard naming methodology is necessary
   so that the components of the directory can easily and
   algorithmically locate the schema entries. All schema information
   relevant to that naming context is stored below the subschema entry.
   Children of the subschema entry store information about objects,
   attribute types, attribute syntaxes or matching rules. The DIT

structure for storing schema information is shown in Figure 1.
Schema for these objects are given in section 5.

4. Retrieval of Schema from the Directory

When an unknown object is encountered by any component of directory
during a directory operation, it proceeds the following way to
resolve the schema.

The RDN component at the leaf-end of the name of the object whose
schema is to be resolved is replaced by the RDNs "oid=<object
identifier of the new object>, CN=subschema" and a read request is
initiated for the newly formed name. If the entry is not found, two
RDN components from the leaf-end of the name of the object are
replaced by the RDNs "oid=<object identifier of the new object>,
CN=subschema" and another read is attempted. The process continues
until the read succeeds. For example, while resolving the schema of
the object "IPNI=spark, OU=Department of Computer Science, O=Indian
Institute of Technology, Madras , C=IN", if the schema of the object
IPNI (IP Node Image) is not known to a component of the directory,
the following procedure will be adopted.

Let the object id for the object IPNI be ipni. The RDN "IPNI=spark"
is removed from the distinguished name of the entry and the RDNs
"oid=ipni, CN= Subschema" is appended. The name thus formed is
"oid=ipni, CN=subschema, OU=Department of Computer Science, O=Indian
Institute of Technology, Madras, C=IN" A read request is initiated on
this name. If the distinguished name "OU= Department of Computer
Science, O=Indian Institute of Technology, Madras, C=IN" is the
context prefix of a naming context, this read request will result in
the directory returning the schema for the object IPNI. If it is not,
the read operation will fail. In that case, a read operation is
initiated with distinguished name "oid=ipni, CN= subschema, O=Indian
Institute of Technology, Madras, C=IN". For the DIT structure shown
in Figure-1, this query will succeed and the schema information will
be returned. The schema for the requested object will always be
located below the starting entry of the naming context in which the
entry is located.

5. The Meta-Schema

experimental = 1.3.6.1.3

schema OBJECT IDENTIFIER
        ::= {experimental 65}

schemaObjectClass OBJECT IDENTIFIER
        ::= {schema.1}

schemaAttribute OBJECT IDENTIFIER
        ::= {schema.2}

subschema OBJECT CLASS
    Subclass of TOP
        MUST CONTAIN {
            commonName
            - - For naming
        }
        ::= {schemaObjectClass.1}

objectClass OBJECT CLASS
    Subclass of TOP
        MUST CONTAIN {
            objectIdentifier
                - - This field stores the object identifier of object
                - - represented by an object class entry. This attribute
                - - is used for naming an object class entry.
        }
        MAY CONTAIN {
            commonName,
                - - This field is used to store the name of the object
            mandatoryNamingAttributes,
            mandatoryAttributes,
            optionalNamingAttibutes,
            optionalAttributes,
            obsolete,
            description,
            subClassOf
        }
        ::= {schemaObjectClass.2}

attributeType OBJECT CLASS
    Subclass of Top
        MUST CONTAIN {
            objectIdentifier
        }
        MAY CONTAIN {

```
               commonName,
                  - - used to store the name of the attribute type
               constraint,
               attributeSyntax,
               multivalued,
               obsolete,
               matchRules,
               description
           }
           ::= {schemaObjectClass.3}

matchingRule OBJECT CLASS
      Subclass of Top
         MUST CONTAIN {
               objectIdentifier
          }
          MAY CONTAIN {
               commonName,
               matchtype,
               description,
               obsolete
          }
          ::= {schemaObjectClass.4}

objectIdentifier ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX
           objectIdentifierSyntax
        ::= {schemaAttribute.1}

mandatoryNamingAttributes ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX
            SET OF OBJECT IDENTIFIER
        ::= {schemaAttribute.2}

mandatoryAttributes ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX
            SET OF OBJECT IDENTIFIER
        ::= {schemaAttribute.3}

optionalNamingAttibutes ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX
            SET OF OBJECT IDENTIFIER
        ::= {schemaAttribute.4}

optionalAttibutes ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX
            SET OF OBJECT IDENTIFIER
        ::= {schemaAttribute.5}
```

```
obsolete ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX
            BOOLEAN
                            -- DEFAULT FALSE
        ::= {schemaAttribute.6}

subClassOf      ATTRIBUTE
         WITH ATTRIBUTE-SYNTAX
                SET OF OBJECT IDENTIFIER
        ::= {schemaAttribute.7}

constraint ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX
            Constraint
        ::= {schemaAttribute.8}

Constraint ::=Choice {
            StringConstraint,
            IntegerConstraint
        }

StringConstraint ::= SEQUENCE {
            shortest INTEGER,
            longest  INTEGER
        }

IntegerConstraint ::= SEQUENCE {
            lowerbound INTEGER,
            upperbound INTEGER OPTIONAL
        }

attributeSyntax ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX
            ASN1DataType
        ::= {schemaAttribute.9}

multivalued ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX
            BOOLEAN              -- DEFAULT FALSE
        ::= {schemaAttribute.10}

matchRules ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX
            SET OF OBJECT IDENTIFIER
        ::= {schemaAttribute.11}

matchtype ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX
```

```
        INTEGER {
         PRESENT                    (0),
         EQUALITY                   (1),
         ORDERING                   (2),
         CASESENSITIVEMATCH         (3),
         CASEINSENSITIVEMATCH       (4)
        }
    ::= {schemaAttribute.12}
```

6. References

   [1] CCITT. "Data Communication Networks: Directory", Recommendations
       X.500 - X.521 1988.

   [2] CCITT. "Data Communication Networks: Directory", Recommendations
       X.500 - X.525 1993.

   [3] Barker, P., and S. Kille, "The COSINE and Internet X.500 Schema",
       RFC 1274, University College London, November 1991.

   [4] Howes, T., "Schema Information in the X.500 Directory", Work in
       Progress, University of Michigan, July 1992.

   [5] Howes, T., Rossen, K., Sataluri, S., and R. Wright, "Procedures
       for Formalization, Evolution, and Maintenance of the Internet
       X.500 Directory Schema", Work in Progress, June 1995.

7. Security Considerations

   Security issues are not discussed in this memo.

8. Authors' Addresses

Glenn Mansfield
AIC Systems Laboratories,
6-6-3, Minami Yoshinari, Aoba-Ku, Sendai,
Japan

Phone: +81 (22) 279-3310
Fax: +81 (22) 279-3640
EMail: glenn@aic.co.jp

P. V. Rajeev
Hughes Software Systems,
2nd Floor, International Trade Tower,
Nehru Place, New Delhi,
India

EMail: rajeev%hss@lando.hns.com

S. V. Raghavan
Department of Computer Science and Engineering,
Indian Institute of Technology, Madras 600 036,
India

EMail: svr@iitm.ernet.in

Tim Howes
University of Michigan
ITD Research Systems
535 W William St.
Ann Arbor, MI 48103-4943, USA

Phone: +1 (313) 747-4454
EMail: tim@umich.edu