Representing Tables and Subtrees in the X.500 Directory

Status of this Memo

Abstract

   This document defines techniques for representing two types of
   information mapping in the OSI Directory [1].

   1. Mapping from a key to a value (or set of values), as might be
       done in a table lookup.

   2. Mapping from a distinguished name to an associated value (or
       values), where the values are not defined by the owner of the
       entry. This is achieved by use of a directory subtree.

   These techniques were developed for supporting MHS use of Directory
   [2], but are specified separately as they have more general
   applicability.

1. Representing Flat Tables

   Before considering specific function, a general purpose technique for
   representing tables in the directory is introduced. The schema for
   this is given in Figure 1.

   A table can be considered as an unordered set of key to (single or
   multiple) value mappings, where the key cannot be represented as a
   global name. There are four reasons why this may occur:

   1. The object does not have a natural global name.

   2. The object can only be named effectively in the context of being
       a key to a binding. In this case, the object will be given a
       natural global name by the table.

   3. The object has a global name, and the table is being used to
       associate parameters with this object, in cases where they cannot
       be placed in the objects global entry. Reasons why they might
       not be so placed include:

        o The object does not have a directory entry

        o There is no authority to place the parameters in the global
           entry

        o The parameters are not global --- they only make sense in the
           context of the table.

   4. It is desirable to group information together as a performance
       optimisation, so that the block of information may be widely
       replicated.

   A table is represented as a single level subtree. The root of the
   subtree is an entry of object class Table. This is named with a
   common name descriptive of the table. The table will be located
   somewhere appropriate to its function. If a table is private to an
   MTA, it will be below the MTA's entry. If it is shared by MTA's in
   an organisation, it will be located under the organisation.

   The generic table entry contains only a description. All instances
   will be subclassed, and the subclass will define the naming
   attribute. Two subclasses are defined:

----
table OBJECT-CLASS ::= {
    SUBCLASS OF {top}
    MUST CONTAIN {commonName}
    MAY CONTAIN {manager}
    ID oc-table}

tableEntry OBJECT-CLASS ::= {
    SUBCLASS OF {top}
    MAY CONTAIN {description}   10
    ID oc-table-entry}

textTableEntry OBJECT-CLASS ::= {
    SUBCLASS OF {tableEntry}
    MUST CONTAIN {textTableKey}
    MAY CONTAIN {textTableValue}
    ID oc-text-table-entry}

textTableKey ATTRIBUTE ::= {

```
    SUBTYPE OF name                                                  20
    WITH SYNTAX DirectoryString {ub-name}
    ID at-text-table-key}

textTableValue ATTRIBUTE ::= {
    SUBTYPE OF name
    WITH SYNTAX  DirectoryString {ub-description}
    ID at-text-table-value}

distinguishedNameTableEntry OBJECT-CLASS ::= {
    SUBCLASS OF {tableEntry}                                         30
    MUST CONTAIN {distinguishedNameTableKey}
    ID oc-distinguished-name-table-entry}

distinguishedNameTableKey ATTRIBUTE ::= {
    SUBTYPE OF distinguishedName
    ID at-distinguished-name-table-key}
```

                        Figure 1: Representing Tables

1. TextEntry, which define table entries with text keys, which may
    have single or multiple values of any type. An attribute is
    defined to allow a text value, to support the frequent text key to
    text value mapping. Additional values may be defined.

2. DistinguishedNameEntry. This is used for associating information
    with globally defined objects. This approach should be used where
    the number of objects in the table is small or very sparsely
    spread over the DIT. In other cases where there are many objects
    or the objects are tightly clustered in the DIT, the subtree
    approach defined in Section 2 will be preferable. No value
    attributes are defined for this type of entry. An application of
    this will make appropriate subtyping to define the needed values.

This is best illustrated by example. Consider the MTA:

CN=Bells, OU=Computer Science,
O=University College London, C=GB

Suppose that the MTA needs a table mapping from private keys to fully
qualified domain names (this example is fictitious). The table might
be named as:

CN=domain-nicknames,
CN=Bells, OU=Computer Science,
O=University College London, C=GB

To represent a mapping in this table from "euclid" to
"bloomsbury.ac.uk", the entry:

CN=euclid, CN=domain-nicknames,
CN=Bells, OU=Computer Science,
O=University College London, C=GB

will contain the attribute:

TextTableValue=bloomsbury.ac.uk

A second example, showing the use of DistinguishedNameEntry is now
given. Consider again the MTA:

CN=Bells, OU=Computer Science,
O=University College London, C=GB

Suppose that the MTA needs a table mapping from MTA Name to bilateral
agreement information of that MTA. The table might be named as:

CN=MTA Bilateral Agreements,
CN=Bells, OU=Computer Science,
O=University College London, C=GB

To represent information on the MTA which has the Distinguished Name:

CN=Q3T21, ADMD=Gold 400, C=GB

   There would be an entry in this table with the Relative Distinguished
   Name of the table entry being the Distinguished Name of the MTA being
   referred to. The MTA Bilateral information would be an attribute in
   this entry. Using a non-standard notation, the Distinguished Name of
   the table entry is:

   DistinguishedNameTableValue=<CN=Q3T21, ADMD=Gold 400, C=GB>,
   CN=MTA Bilateral Agreements,
   CN=Bells, OU=Computer Science,
   O=University College London, C=GB

2. Representing Subtrees

   A subtree is similar to a table, except that the keys are constructed
   as a distinguished name hierarchy relative to the location of the
   subtree in the DIT. The subtree effectively starts a private "root",
   and has distinguished names relative to this root. Typically, this
   approach is used to associate local information with global objects.
   The schema used is defined in Figure 2. Functionally, this is
   equivalent to a table with distinguished name keys. The table
   approach is best when the tree is very sparse. This approach is
   better for subtrees which are more populated.

   The subtree object class defines the root for a subtree in an
   analogous means to the table. Information within the subtree will
   generally be defined in the same way as for the global object, and so

   ----
   subtree OBJECT-CLASS ::= {
       SUBCLASS OF {top}
       MUST CONTAIN {commonName}
       MAY CONTAIN {manager}
       ID oc-subtree}

                    Figure 2: Representing Subtrees

   no specific object classes for subtree entries are needed.

   For example consider University College London.

   O=University College London, C=GB

   Suppose that the UCL needs a private subtree, with interesting
   information about directory objects. The table might be named as:

   CN=private subtree,
   O=University College London, C=GB

   UCL specific information on Inria might be stored in the entry:

   O=Inria, C=FR,
   CN=private subtree,
   O=University College London, C=GB

   Practical examples of this mapping are given in [2].

3. Acknowledgements

    Acknowledgements for work on this document are given in [2].

References

    [1] The Directory --- overview of concepts, models and services,
        1993. CCITT X.500 Series Recommendations.

    [2] Kille, S., "MHS use of the X.500 Directory to Support MHS
        Routing", RFC 1801, ISODE Consortium, June 1995.

4. Security Considerations

    Security issues are not discussed in this memo.

5. Author's Address

    Steve Kille
    ISODE Consortium
    The Dome
    The Square
    Richmond
    TW9 1DT
    England

    Phone: +44-81-332-9091
    Internet EMail: S.Kille@ISODE.COM
    X.400: I=S; S=Kille; O=ISODE Consortium; P=ISODE;
    A=Mailnet; C=FI;
    DN: CN=Steve Kille,
    O=ISODE Consortium, C=GB
    UFN: S. Kille, ISODE Consortium, GB

A. Object Identifier Assignment

```
-----------------------------------------------------------------------
mhs-ds OBJECT IDENTIFIER ::= {iso(1) org(3) dod(6) internet(1)
         private(4) enterprises(1) isode-consortium (453) mhs-ds (7)}

tables OBJECT IDENTIFIER ::= {mhs-ds 1}

oc OBJECT IDENTIFIER ::= {tables 1}
at OBJECT IDENTIFIER ::= {tables 2}

oc-subtree OBJECT IDENTIFIER ::= {oc 1}
oc-table OBJECT IDENTIFIER ::= {oc 2}                                10
oc-table-entry OBJECT IDENTIFIER ::= {oc 3}
oc-text-table-entry OBJECT IDENTIFIER ::= {oc 4}
oc-distinguished-name-table-entry  OBJECT IDENTIFIER ::= {oc 5}

at-text-table-key OBJECT IDENTIFIER ::= {at 1}
at-text-table-value OBJECT IDENTIFIER ::= {at 2}
at-distinguished-name-table-key OBJECT IDENTIFIER ::= {at 3}
```

                Figure 3: Object Identifier Assignment