Network Working Group
INTERNET-DRAFT

S.E. Hardcastle-Kille
ISODE Consortium

November 1992
Expires: June 1993

Representing Tables and Subtrees in the Directory

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. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

Please check the I-D abstract listing contained in each Internet Draft directory to learn the current status of this or any other Internet Draft.

Abstract

This document defines techniques for representing two types of information mapping in the OSI Directory [CCI88].

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.

This techniques were developed for supporting MHS use of Directory [HK92], but are specified separately as they have more general applicability.

This draft document will be submitted to the RFC editor as a protocol standard. Distribution of this memo is unlimited. Please send comments to the author or to the discussion group `<mhs-ds@mercury.udev.cdc.com>`.

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:
   - The object does not have a directory entry
   - There is no authority to place the parameters in the global entry
   - 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:

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

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


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

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

textTableKey ATTRIBUTE
    WITH ATTRIBUTE−SYNTAX
            caseIgnoreStringSyntax
    ::= at−text−table−key

textTableValue ATTRIBUTE
    WITH ATTRIBUTE−SYNTAX
            caseIgnoreStringSyntax
    ::= at−text−table−value

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

distinguishedNameTableKey ATTRIBUTE
    WITH ATTRIBUTE−SYNTAX
            distinguishedNameSyntax
    ::= at−distinguished−name−table−key
```

Figure 1: Representing Tables

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:

```
TextValue=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:

```
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.

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

Figure 2: Representing Subtrees

2 Representing Subtrees

A subtree is similar to a table, except that the keys are constructed 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 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 [HK92].

References

[CCI88] The Directory — overview of concepts, models and services, December 1988. CCITT X.500 Series Recommendations.

[HK92] S.E. Hardcastle-Kille. MHS use of the directory to support MHS routing, April 1992. Internet Draft.

3 Security Considerations

Security considerations are not discussed in this INTERNET–DRAFT .

4 Author's Address

Steve Hardcastle-Kille
ISODE Consortium
PO Box 505
London
SW11 1DX
England

Phone: +44-71-223-4062

EMail: S.Kille@ISODE.COM

DN: CN=Steve Hardcastle-Kille,
O=ISODE Consortium, C=GB

UFN: S. Hardcastle-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 (3)}

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}
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