

Network Working
Group
INTERNET-DRAFT

S.E. Kille
ISODE Consortium
June 1994
Expires: December 1994
File:
[draft-ietf-mhsds-routdirectory-05.txt](#), ps

MHS use of Directory to support MHS Routing

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 specifies an approach for X.400 Message Handling Systems to perform application level routing using the OSI Directory [[20](#), [1](#)]. Use of the directory in this manner is fundamental to enabling large scale deployment of X.400.

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>.

INTERNET--DRAFT

MHS Routing using Directory

June 1994

Contents

1	Introduction	6
2	Goals	6

<u>3</u>	Approach	9
<u>4</u>	Direct vs Indirect Connection	9
<u>5</u>	X.400 and RFC 822	12
<u>6</u>	Objects	12
<u>7</u>	Communities	14
<u>8</u>	Routing Trees	15
8.1	Routing Tree Definition	16
8.2	The Open Community Routing Tree	16
8.3	Routing Tree Location	17
8.4	Example Routing Trees	17
8.5	Use of Routing Trees to look up Information	18
<u>9</u>	Routing Tree Selection	19
9.1	Routing Tree Order	19
9.2	Example use of Routing Trees	20
9.2.1	Fully Open Organisation	20
9.2.2	Open Organisation with Fallback	20
9.2.3	Minimal-routing MTA	20
9.2.4	Organisation with Firewall	21
9.2.5	Well Known Entry Points	21
9.2.6	ADMD using the Open Community for Advertising	21
9.2.7	ADMD/PRMD gateway	22
<u>10</u>	Routing Information	22

Kille Expires: December 1994 Page 1

INTERNET--DRAFT MHS Routing using Directory June 1994

10.1	Multiple routing trees	26
10.2	MTA Choice	27
10.3	Routing Filters	30
10.4	Indirect Connectivity	32

11	Local Addresses (UAs)	33
11.1	Searching for Local Users	34
12	Direct Lookup	35
13	Alternate Routes	35
13.1	Finding Alternate Routes	35
13.2	Sharing routing information	35
14	Looking up Information in the Directory	36
15	Naming MTAs	37
15.1	Naming 1984 MTAs	39
16	Attributes Associated with the MTA	39
17	Bilateral Agreements	41
18	MTA Selection	42
18.1	Dealing with protocol mismatches	42
18.2	Supported Protocols	43
18.3	MTA Capability Restrictions	43
18.4	Subtree Capability Restrictions	43
19	MTA Pulling Messages	44
20	Security and Policy	44
20.1	Finding the Name of the Calling MTA	44
20.2	Authentication	45
20.3	Authentication Information	46

Kille Expires: December 1994 Page 2

INTERNET--DRAFT MHS Routing using Directory June 1994

21	Policy and Authorisation	47
21.1	Simple MTA Policy	47
21.2	Complex MTA Policy	48

<u>22</u>	Delivery	49
22.1	Redirects	49
22.2	Underspecified O/R Addresses	50
22.3	Non Delivery	50
22.4	Bad Addresses	50
<u>23</u>	Submission	51
23.1	Normal Derivation	51
23.2	Roles and Groups	52
<u>24</u>	Access Units	52
<u>25</u>	The Overall Routing Algorithm	53
<u>26</u>	Performance	54
<u>27</u>	Acknowledgements	54
<u>28</u>	Security Considerations	56
<u>29</u>	Author's Address	56
A	Object Identifier Assignment	58
B	Community Identifier Assignments	60
C	Protocol Identifier Assignments	61
D	ASN.1 Summary	62
E	Regular Expression Syntax	73

Kille Expires: December 1994 Page 3

INTERNET--DRAFT MHS Routing using Directory June 1994

E.1	Pseudo Code	75
-----	-----------------------	----

List of Figures

1	Location of Routing Trees	17
2	Routing Tree Use Definition	19
3	Routing Information at a Node	24
4	Indirect Access	32

6	MTA Definitions	38
19	Object Identifier Assignment	59
22	ASN.1 Summary	72
5	UA Attributes	76
7	MTA Bilateral Table Entry	76
8	Bilateral Table Attribute	77
9	Supported MTS Extensions	77
10	Subtree Capability Restriction	77
11	Pulling Messages	78
12	Authentication Requirements	79
13	MTA Authentication Parameters	80
14	Simple MTA Policy Specification	81
15	Redirect Definition	81
16	Non Delivery Information	82
17	Bad Address Pointers	82
18	Access Unit Attributes	83
20	Transport Community Object Identifier Assignments	83
21	Protocol Object Identifier Assignments	84

List of Tables

1	Possible target end to end delays	7
---	---	---

Kille

Expires: December 1994 Page 5

INTERNET--DRAFT

MHS Routing using Directory

June 1994

1 Introduction

MHS Routing is the problem of controlling the path of a message as it traverses one or more MTAs to reach its destination recipients. Routing starts with a recipient O/R Address, and parameters associated with the message to be routed. It is assumed that this is known a priori, or is derived at submission time as described in [Section 23](#).

The key problem in routing is to map from an O/R Address onto an MTA (next hop). This shall be an MTA which in some sense is ``nearer'' to the destination UA. This is done repeatedly until the message can be directly delivered to the recipient UA. There are a number of things which need to be considered to determine this. These are discussed in the subsequent sections. A description of the overall routing process is given in [Section 25](#).

2 Goals

Application level routing for MHS is a complex procedure, with many requirements. The following goals for the solution are set:

- o Straightforward to manage. Non-trivial configuration of routing for current message handling systems is a black art, often involving gathering and processing many tables, and editing complex configuration files. Many problems are solved in a very ad hoc manner. Managing routing for MHS is the most serious headache for most mail system managers.
- o Economic, both in terms of network and computational resources.
- o Robust. Errors and out of date information shall cause minimal and localised damage.
- o Deal with link failures. There needs to be some ability to choose alternative routes. In general, it is desirable that the routing approach be redundant.
- o Load sharing. Information on routes shall allow ``equal'' routes to be specified, and thus facilitate load sharing.
- o Support format and protocol conversion

Kille

Expires: December 1994 Page 6

INTERNET--DRAFT

MHS Routing using Directory

June 1994

	High_Priority_	Normal_Priority_	Low_Priority_	
50%_max_delay_	0.5_hour_____	1_hour_____	12_hours_____	
98%_max_delay_	3_hours_____	12_hours_____	24_hours_____	
max_delay_____	6_hours_____	72_hours_____	96_hours_____	

Table 1: Possible target end to end delays

- o Dynamic and automatic. There shall be no need for manual propagation of tables or administrator intervention.
- o Policy robust. It shall not allow specification of policies which cause undesirable routing effects.
- o Reasonably straightforward to implement.
- o Deal with X.400, [RFC 822](#), and their interaction.
- o Extensible to other mail architectures
- o Recognise existing [RFC 822](#) routing, and coexist smoothly.
- o Improve [RFC 822](#) routing capabilities. This is particularly important for [RFC 822](#) sites not in the SMTP Internet.
- o Deal correctly with different X.400 protocols (P1, P3, P7), and with 1984, 1988 and 1992 versions.
- o Support X.400 operation over multiple protocol stacks (TCP/IP, CONS, CLNS) and in different communities.
- o Messages shall be routed consistently. Alternate routing strategies, which might introduce unexpected delay, shall be used with care (e.g. routing through a protocol converter due to unavailability of an MTA).
- o Delay between message submission and delivery shall be minimised. Table 1 indicates the sort of target which might be aimed for. The figures are illustrative of the sort of target which might be set. They are better than achieved in most current Research Networks. The long tail-offs on Normal and Low priority recognise the fact that some end systems will not get 24 hour operator coverage (whereas any MTAs providing ADMD service will usually be required to do this). In the case of a high and normal priority

Kille

Expires: December 1994 Page 7

INTERNET--DRAFT

MHS Routing using Directory

June 1994

messages, it is a target that a non-delivery notification be returned within the suggested time.

- o Interact sensibly with ADMD services.
- o Be global in scope

- o Routing strategy shall deal with a scale of order of magnitude 1,000,000 -- 100,000,000 MTAs.
- o Routing strategy shall deal with of order 1,000,000 -- 100,000,000 Organisations.
- o Information about alterations in topology shall propagate rapidly to sites affected by the change.
- o Removal, examination, or destruction of messages by third parties shall be difficult. This is hard to quantify, but ``difficult'' shall be comparable to the effort needed to break system security on a typical MTA system.
- o As with current Research Networks, it is recognised that prevention of forged mail will not always be possible. However, this shall be as hard as can be afforded.
- o Sufficient tracing and logging shall be available to track down security violations and faults.
- o Optimisation of routing messages with multiple recipients, in cases where this involves selection of preferred single recipient routes.

The following are not initial goals:

- o Advanced optimisation of routing messages with multiple recipients, noting dependencies between the recipients to find routes which would not have been chosen for any of the single recipients.
- o Dynamic load balancing. The approach does not give a means to determine load. However, information on alternate routes is provided, which is the static information needed for load balancing.

Kille

Expires: December 1994 Page 8

INTERNET--DRAFT

MHS Routing using Directory

June 1994

3 Approach

A broad problem statement, and a survey of earlier approaches to the problem is given in the COSINE Study on MHS Topology and Routing [8]. The interim (table-based) approach suggested in this study, whilst not

being followed in detail, broadly reflects what the research X.400 (GO-MHS) community is doing. The evolving specification of the RARE table format is defined in [5]. This document specifies the envisaged longer term approach.

Some documents have made useful contributions to this work:

- o A paper by the editor on MHS use of directory, which laid out the broad approach of mapping the O/R Address space on to the DIT [7].
- o Initial ISO Standardisation work on MHS use of Directory for routing [21]. Subsequent ISO work in this area has drawn from earlier drafts of this specification.
- o The work of the VERDI Project [3].
- o Work by Kevin Jordan of CDC [6].
- o The routing approach of ACSNet [4, 19] paper. This gives useful ideas on incremental routing, and replicating routing data.
- o A lot of work on network routing is becoming increasingly relevant. As the MHS routing problem increases in size, and network routing increases in sophistication (e.g., policy based routing), the two areas have increasing amounts in common. For example, see [2].

4 Direct vs Indirect Connection

Two extreme approaches to routing connectivity are:

- 1. High connectivity between MTAs.** An example of this is the way the Domain Name Server system is used on the DARPA/NSF Internet. Essentially, all MTAs are fully interconnected.
- 2. Low connectivity between MTAs.** An example of this is the UUCP network.

Kille

Expires: December 1994 Page 9

INTERNET--DRAFT

MHS Routing using Directory

June 1994

In general an intermediate approach is desirable. Too sparse a connectivity is inefficient, and leads to undue delays. However, full connectivity is not desirable, for the reasons discussed below. A number of general issues related to relaying are now considered.

The reasons for avoiding relaying are clear. These include.

- o Efficiency. If there is an open network, it is desirable that it be used.
- o Extra hops introduce delay, and increase the (very small) possibility of message loss. As a basic principle, hop count shall be minimised.
- o Busy relays or Well Known Entry points can introduce high delay and lead to single point of failure.
- o If there is only one hop, it is straightforward for the user to monitor progress of messages submitted. If a message is delayed, the user can take appropriate action.
- o Many users like the security of direct transmission. It is an argument often given very strongly for use of SMTP.

Despite these very powerful arguments, there are a number of reasons why some level of relaying is desirable:

- o Charge optimisation. If there is an expensive network/link to be traversed, it may make sense to restrict its usage to a small number of MTAs. This would allow for optimisation with respect to the charging policy of this link.
- o Copy optimisation. If a message is being sent to two remote MTAs which are close together, it is usually optimal to send the message to one of the MTAs (for both recipients), and let it pass a copy to the other MTA.
- o To access an intermediate MTA for some value added service. In particular for:
 - Message Format Conversion
 - Distribution List expansion

Kille

Expires: December 1994 Page 10

INTERNET--DRAFT

MHS Routing using Directory

June 1994

- o Dealing with different protocols. The store and forward approach allows for straightforward conversion. Relevant cases include:
 - Provision of X.400 over different OSI Stacks (e.g.

Connectionless Network Service).

-- Use of a different version of X.400.

-- Interaction with non-X.400 mail services

- o To compensate for inadequate directory services: If tables are maintained in an ad hoc manner, the manual effort to gain full connectivity is too high.
- o To hide complexity of structure. If an organisation has many MTAs, it may still be advantageous to advertise a single entry point to the outside world. It will be more efficient to have an extra hop, than to (widely) distribute the information required to connect directly. This will also encourage stability, as organisations need to change internal structure much more frequently than their external entry points. For many organisations, establishing such firewalls is high priority.
- o To handle authorisation, charging and security issues. In general, it is desirable to deal with user oriented authorisation at the application level. This is essential when MHS specific parameters shall be taken into consideration. It may well be beneficial for organisations to have a single MTA providing access to the external world, which can apply a uniform access policy (e.g. as to which people are allowed access). This would be particularly true in a multi-vendor environment, where different systems would otherwise have to enforce the same policy --- using different vendor-specific mechanisms.

In summary there are strong reasons for an intermediate approach. This will be achieved by providing mechanisms for both direct and indirect connectivity. The manager of a configuration will then be able to make appropriate choices for the environment. Two models of managing large scale routing have evolved:

- 1. Use of a global directory/database.** This is the approach proposed here.

Kille

Expires: December 1994 Page 11

INTERNET--DRAFT

MHS Routing using Directory

June 1994

- 2. Use of a routing table in each MTA, which is managed either by a management protocol or by directory.** This is coupled with means to exchange routing information between MTAs. This approach is

more analogous to how network level routing is commonly performed. It has good characteristics in terms of managing links and dealing with link related policy. However, it assumes limited connectivity and does not adapt well to a network environment with high connectivity available.

5 X.400 and [RFC 822](#)

This document defines mechanisms for X.400 routing. It is important that this can be integrated with [RFC 822](#) base routing, as many MTAs will work in both communities. This routing document is written with this problem in mind, and support for [RFC 822](#) routing using the same basic infrastructure is defined in a companion document [[12](#)]. In addition support for X.400/RFC 822 gatewaying is needed, to support interaction. Directory based mechanisms for this are defined in [[16](#)]. The advantages of the approach defined by this set of specifications are:

- o Uniform management for sites which wish to support both protocols.
- o Simpler management for gateways.
- o Improved routing services for [RFC 822](#) only sites.

For sites which are only X.400 or only [RFC 822](#), the mechanisms associated with gatewaying or with the other form of addressing are not needed.

6 Objects

It is useful to start with a manager's perspective. Here is the set of object classes used in this specification. It is important that all information entered relates to something which is being managed. If this is achieved, configuration decisions are much more likely to be correct. In the examples, distinguished names are written using the String Syntax for Distinguished Names [[10](#)]. The list of objects used in this specification is:

Kille

Expires: December 1994 Page 12

User An entry representing a single human user. This will typically be named in an organisational context. For example:

CN=Edgar Smythe,
O=Zydeco Services, C=GB

This entry would have associated information, such as telephone number, postal address, and mailbox.

MTA A Message Transfer Agent. In general, the binding between machines and MTAs will be complex. Often a small number of MTAs will be used to support many machines, by use of local approaches such as shared filestores. MTAs may support multiple protocols, and will identify separate addressing information for each protocol.

To achieve support for multiple protocols, an MTA is modelled as an Application Process, which is named in the directory. Each MTA will have one or more associated Application Entities. Each Application Entity is named as a child of the Application Process, using a common name which conveniently identifies the Application Entity relative to the Application Process. Each Application Entity supports a single protocol, although different Application Entities may support the same protocol. Where an MTA only supports one protocol or where the addressing information for all of the protocols supported have different attributes to represent addressing information (e.g., P1(88) and SMTP) the Application Entity(ies) may be represented by the single Application Process entry.

User Agent (Mailbox) This defines the User Agent (UA) to which mail may be delivered. This will define the account with which the UA is associated, and may also point to the user(s) associated with the UA. It will identify which MTAs¹ are able to access the UA.

Role Some organisational function. For example:

1. In the formal X.400 model, there will be a single MTA delivering to a UA. In many practical configurations, multiple MTAs can deliver to a single UA. This will increase robustness, and is desirable.

Kille

Expires: December 1994 Page 13

CN=System Manager, OU=Sales,

O=Zydeco Services, C=GB

The associated entry would indicate the occupant of the role.

Distribution Lists There would be an entry representing the distribution list, with information about the list, the manager, and members of the list.

7 Communities

There are two basic types of agreement in which an MTA may participate in order to facilitate routing:

Bilateral Agreements An agreement between a pair of MTAs to route certain types of traffic. This MTA pair agreement usually reflects some form of special agreement and in general bilateral information shall be held for the link at both ends. In some cases, this information shall be private.

Open Agreements An agreement between a collection of MTAs to behave in a cooperative fashion to route traffic. This may be viewed as a general bilateral agreement.

It is important to ensure that there are sufficient agreements in place for all messages to be routed. This will usually be done by having agreements which correspond to the addressing hierarchy. For X.400, this is the model where a PRMD connects to an ADMD, and the ADMD provides the inter PRMD connectivity, by the ability to route to all other ADMDs. Other agreements may be added to this hierarchy, in order to improve the efficiency of routing. In general, there may be valid addresses, which cannot be routed to, either for connectivity or policy reasons.

We model these two types of agreements as communities. A community is a scope in which an MTA advertises its services and learns about other services. Each MTA will:

1. Register its services in one or more communities.

Kille

Expires: December 1994 Page 14

2. Look up services in one or more communities.

In most cases an MTA will deal with a very small number of communities --- very often one only. There are a number of different types of community.

The open community This is a public/global scope. It reflects routing information which is made available to any MTA which wishes to use it.

The local community This is the scope of a single MTA. It reflects routing information private to the MTA. It will contain an MTA's view of the set of bilateral agreements in which it participates, and routing information private and local to the MTA.

Hierarchical communities A hierarchical community is a subtree of the O/R Address tree. For example, it might be a management domain, an organisation, or an organisational unit. This sort of community will allow for firewalls to be established. A community can have complex internal structure, and register a small subset of that in the open community.

Closed communities A closed community is a set of MTAs which agrees to route amongst themselves. Examples of this might be ADMs within a country, or a set of PRMs representing the same organisation in multiple countries.

Formally, a community indicates the scope over which a service is advertised. In practice, it will tend to reflect the scope of services offered. It does not make sense to offer a public service, and only advertise it locally. Public advertising of a private service makes more sense, and this is shown below. In general, having a community offer services corresponding to the scope in which they are advertised will lead to routing efficiency. Examples of how communities can be used to implement a range of routing policies are given in [Section 9.2](#).

8 Routing Trees

Communities are a useful abstract definition of the routing approach taken by this specification. Each community is represented in the

directory as a routing tree. There will be many routing trees instantiated in the directory. Typically, an MTA will only be registered in and make use of a small number of routing trees. In most cases, it will register in and use the same set of routing trees.

8.1 Routing Tree Definition

Each community has a model of the O/R address space. Within a community, there is a general model of what to do with a given O/R Address. This is structured hierarchically, according to the O/R address hierarchy. A community can register different possible actions, depending on the depth of match. This might include identifying the MTA associated with a UA which is matched fully, and providing a default route for an O/R address where there is no match in the community --- and all intermediate forms.

The name structure of a routing tree follows the O/R address hierarchy, which is specified in a separate document [[14](#)]. Where there is any routing action associated with a node in a routing tree, the node is of object class routingInformation, as defined in [Section 10](#).

8.2 The Open Community Routing Tree

The routing tree of the open community starts at the root of the DIT. This routing tree also serves the special function of instantiating the global O/R Address space in the Directory. Thus, if a UA wishes to publish information to the world, this hierarchy allows it to do so.

The O/R Address hierarchy is a registered tree, which may be instantiated in the directory. Names at all points in the tree are valid, and there is no requirement that the namespace is instantiated by the owner of the name. For example, a PRMD may make an entry in the DIT, even if the ADMD above it does not. In this case, there will be a ``skeletal'' entry for the ADMD, which is used to hang the PRMD entry in place. The skeletal entry contains the minimum number of entries which are needed for it to exist in the DIT (Object Class and Attribute information needed for the relative distinguished name). This entry may be placed there solely to support the subordinate entry, as its existence is inferred by the subordinate entry. Only the owner of the entry may place information into it. An analogous situation in current operational practice is to make DIT entries for

```
routingTreeRoot OBJECT-CLASS ::= {  
    SUBCLASS OF {routingInformation|subtree}  
    ID oc-routing-tree-root}
```

Figure 1: Location of Routing Trees

Countries and US States.

8.3 Routing Tree Location

All routing trees follow the same O/R address hierarchy. Routing trees other than the open community routing tree are rooted at arbitrary parts of the DIT. These routing trees are instantiated using the subtree mechanism defined in the companion document ``Representing Tables and Subtrees in the Directory'' [14]. A routing tree is identified by the point at which it is rooted. An MTA will use a list of routing trees, as determined by the mechanism described in [Section 9](#). Routing trees may be located in either the organisational or O/R address structured part of the DIT. All routing trees, other than the open community routing tree, are rooted by an entry of object class routingTreeRoot, as defined in Figure 1.

8.4 Example Routing Trees

Consider routing trees with entries for O/R Address:

P=ABC; A=XYZMail; C=GB;

In the open community routing tree, this would have a distinguished name of:

PRMD=ABC, ADMD=XYZMail, C=GB

Consider a routing tree which is private to:

Kille

Expires: December 1994 Page 17

O=Zydeco Services, C=GB

They might choose to label a routing tree root ``Zydeco Routing Tree'', which would lead to a routing tree root of:

CN=Zydeco Routing Tree, O=Zydeco Services, C=GB

The O/R address in question would be stored in this routing tree as:

PRMD=ABC, ADMD=XYZMail
C=GB, CN=Zydeco Routing Tree,
O=Zydeco Services, C=GB

8.5 Use of Routing Trees to look up Information

Lookup of an O/R address in a routing tree is done as follows:

- 1. Map the O/R address onto the O/R address hierarchy described in [\[14\]](#) in order to generate a Distinguished Name.**
- 2. Append this to the Distinguished Name of the routing tree, and then look up the whole name.**
- 3. Handling of errors will depend on the application of the lookup, and is discussed later.**

Note that it is valid to look up a null O/R Address, as the routing tree root may contain default routing information for the routing tree. This is held in the root entry of the routing tree, which is a subclass of routingInformation. The open community routing tree does not have a default.

Routing trees may have aliases into other routing trees. This will typically be done to optimise lookups from the first routing tree which a given MTA uses. Lookup needs to take account of this.

```
routingTreeList ATTRIBUTE ::= {  
    WITH SYNTAX RoutingTreeList  
    SINGLE VALUE  
    ID at-routing-tree-list}  
  
RoutingTreeList ::= SEQUENCE OF RoutingTreeName  
  
RoutingTreeName ::= DistinguishedName
```

Figure 2: Routing Tree Use Definition

9 Routing Tree Selection

The list of routing trees which a given MTA uses will be represented in the directory. This uses the attribute defined in Figure 2. This attribute defines the routing trees used by an MTA, and the order in which they are used. Holding these in the directory eases configuration management. It also enables an MTA to calculate the routing choice of any other MTA which follows this specification, provided that none of its routing trees have access restrictions. This will facilitate debugging routing problems.

9.1 Routing Tree Order

The order in which routing trees are used will be critical to the operation of this algorithm. A common approach will be:

- 1. Access one or more shared private routing trees to access private routing information.**
- 2. Utilise the open routing tree.**
- 3. Fall back to a default route from one of the private routing trees.**

Initially, the open routing tree will be very sparse, and there will be little routing information in ADMD level nodes. Access to many services will only be via ADMD services, which in turn will only be

accessible via private links. For most MTAs, the fallback routing will be important, in order to gain access to an MTA which has the right private connections configured.

In general, for a site, UAs will be registered in one routing tree only, in order to avoid duplication. They may be placed into other routing trees by use of aliases, in order to gain performance. For some sites, Users and UAs with a 1:1 mapping will be mapped onto single entries by use of aliases.

9.2 Example use of Routing Trees

Some examples of how this structure might be used are now given. Many other combinations are possible to suit organisational requirements.

9.2.1 Fully Open Organisation

The simplest usage is to place all routing information in the open community routing tree. An organisation will simply establish O/R addresses for all of its UAs in the open community tree, each registering its supporting MTA. This will give access to all systems accessible from this open community.

9.2.2 Open Organisation with Fallback

In practice, some MTAs and MDs will not be directly reachable from the open community (e.g., ADMDs with a strong model of bilateral agreements). These services will only be available to users/communities with appropriate agreements in place. Therefore it will be useful to have a second (local) routing tree, containing only the name of the fallback MTA at its root. In many cases, this fallback would be to an ADMD connection.

Thus, open routing will be tried first, and if this fails the message will be routed to a single selected MTA.

9.2.3 Minimal-routing MTA

The simplest approach to routing for an MTA is to deliver messages to associated users, and send everything else to another MTA (possibly with backup).

An organisation using MTAs with this approach will register its users as for the fully open organisation. A single routing tree will be established, with the name of the organisation being aliased into the open community routing tree. Thus the MTA will correctly identify local users, but use a fallback mechanism for all other addresses.

9.2.4 Organisation with Firewall

An organisation can establish an organisation community to build a firewall, with the overall organisation being registered in the open community. This is an important structure, which cannot be achieved easily with current technology (e.g., DNS with MX records).

- o Some MTAs are registered in the open community routing tree to give access into the organisation. This will include the O/R tree down to the organisational level. Full O/R Address verification will not take place externally.
- o All users are registered in a private (organisational) routing tree.
- o All MTAs in the organisation are registered in the organisation's private routing tree, and access information in the organisation's community. This gives full internal connectivity.
- o Some MTAs in the organisation access the open community routing tree. These MTAs take traffic from the organisation to the outside world. These will often be the same MTAs that are externally advertised.

9.2.5 Well Known Entry Points

Well known entry points will be used to provide access to countries and MDs which are oriented to private links. A private routing tree will be established, which indicates these links. This tree would be shared by the well known entry points.

9.2.6 ADMD using the Open Community for Advertising

An ADMD uses the open community for advertising. It advertises its existence and also restrictive policy. This will be useful for:

- o Address validation
- o Advertising the mechanism for a bilateral link to be established

[9.2.7](#) ADMD/PRMD gateway

An MTA provides a gateway from a PRMD to an ADMD. It is important to note that many X.400 MDs will not use the directory. This is quite legitimate. This technique can be used to register access into such communities from those that use the directory.

- o The MTA registers the ADMD in its local community (private link)
- o The MTA registers itself in the PRMD's community to give access to the ADMD.

[10](#) Routing Information

Routing trees are defined in the previous section, and are used as a framework to hold routing information. Each node, other than a skeletal one, in a routing tree has information associated with it, which is defined by the object class routingInformation in Figure 3. This structure is fundamental to the operation of this specification, and it is recommended that it be studied with care. _____

```
routingInformation OBJECT-CLASS ::= {
    SUBCLASS OF top
    MAY CONTAIN {
        subtreeInformation|
        routingFilter|
        routingFailureAction|
        mTAInfo|
        accessMD|
        nonDeliveryInfo|
        badAddressSearchPoint|
        badAddressSearchAttributes}
    ID oc-routing-information}
    -- No naming attributes as this is not a
    -- structural object class
```

10

```
subtreeInformation ATTRIBUTE ::= {  
    WITH SYNTAX SubtreeInfo  
    SINGLE VALUE  
    ID at-subtree-information}
```

20

```
SubtreeInfo ::= ENUMERATED {  
    all-children-present(0),  
    not-all-children-present(1) }
```

```
routingFilter ATTRIBUTE ::= {  
    WITH SYNTAX RoutingFilter  
    ID at-routing-filter}
```

30

```
RoutingFilter ::= SEQUENCE{  
    attribute-type OBJECT-IDENTIFIER,  
    weight RouteWeight,  
    dda-key String OPTIONAL,  
    regex-match IA5String OPTIONAL,  
    node DistinguishedName }
```

40

```
String ::= CHOICE {PrintableString, TeletexString}
```

```
routingFailureAction ATTRIBUTE ::= {  
    WITH SYNTAX RoutingFailureAction  
    SINGLE VALUE  
    ID at-routing-failure-action}
```

```
RoutingFailureAction ::= ENUMERATED {  
    next-level(0),  
    next-tree-only(1),  
    next-tree-first(2),  
    stop(3) }
```

50

```
mTAInfo ATTRIBUTE ::= {  
    WITH SYNTAX MTAInfo  
    ID at-mta-info}
```

```
MTAInfo ::= SEQUENCE {  
    name DistinguishedName,
```

60


```

    weight [1] RouteWeight DEFAULT preferred-access,
    mta-attributes [2] SET OF Attribute OPTIONAL,
    ae-info SEQUENCE OF SEQUENCE {
        aEqualifier PrintableString,
        ae-weight RouteWeight DEFAULT preferred-access,
        ae-attributes SET OF Attribute OPTIONAL} OPTIONAL
}

RouteWeight ::= INTEGER {endpoint(0),
    preferred-access(5),
    backup(10)} (0..20)

```

70

Figure 3: Routing Information at a Node

For example, information might be associated with the (PRMD) node:

PRMD=ABC, ADMD=XYZMail, C=GB

If this node was in the open community routing tree, then the information represents information published by the owner of the PRMD relating to public access to that PRMD. If this node was present in another routing tree, it would represent information published by the owner of the routing tree about access information to the referenced PRMD. The attributes associated with a routingInformation node provide the following information:

Implicit That the node corresponds to a partial or entire valid O/R address. This is implicit in the existence of the entry.

Object Class If the node is a UA. This will be true if the node is of object class routedUA. This is described further in [Section 11](#). If it is not of this object class, it is an intermediate node in the O/R Address hierarchy.

routingFilter A set of routing filters, defined by the routingFilter attribute. This attribute provides for routing on information in the unmatched part of the O/R Address. This is described in [Section 10.3](#).

subtreeInformation Whether or not the node is authoritative for the level below is specified by the subtreeInformation attribute. If

it is authoritative, indicated by the value all-children-present, this will give the basis for (permanently) rejecting invalid O/R Addresses. The attribute is encoded as enumerated, as it may be later possible to add partial authority (e.g., for certain attribute types). If this attribute is missing, the node is assumed to be non-authoritative (not-all-children-present). For example, consider the node:

MHS-O=Zydeco, PRMD=ABC, ADMD=XYZMail, C=GB

An organisation which has a bilateral agreement with this organisation has this entry in its routing tree, with no children entries. This is marked as non-authoritative. There is a second routing tree maintained by Zydeco, which contains all of the children of this node, and is marked as authoritative. When considering an O/R Address

MHS-G=Random + MHS-S=Unknown, MHS-O=Zydeco,
PRMD=ABC, ADMD=XYZMail, C=GB

only the second, authoritative, routing tree can be used to determine that this address is invalid. In practice, the manager configuring the non-authoritative tree, will be able to select whether an MTA using this tree will proceed to full verification, or route based on the partially verified information.

mTAInfo A list of MTAs and associated information defined by the mTAInfo attribute. This information is discussed further in Sections [15](#) and [18](#). This information is the key information associated with the node. When a node is matched in a lookup, it indicates the validity of the route, and a set of MTAs to connect to. Selection of MTAs is discussed in Sections [18](#) and [Section 10.2](#).

routingFailureAction An action to be taken if none of the MTAs can be used directly (or if there are no MTAs present) is defined by the routingFailureAction attribute. Use of this attribute and multiple routing trees is described in [Section 10.1](#).

accessMD The accessMD attribute is discussed in [Section 10.4](#). This attribute is used to indicate MDs which provide indirect access to

the part of the tree that is being routed to.

badAddressSearchPoint/badAddressSearchAttributes The

badAddressSearchPoint and badAddressSearchAttributes are discussed in [Section 17](#). This attribute is for when an address has been rejected, and allows information on alternative addresses to be found.

10.1 Multiple routing trees

A routing decision will usually be made on the basis of information contained within multiple routing trees. This section describes the algorithms relating to use of multiple routing trees. Issues relating to use of X.500 and handling of errors is discussed in Section 14. The routing decision works by examining a series of entries (nodes) in one or more routing trees. Each entry may contain information on possible next-hop MTAs. When an entry is found which enables the message to be routed, one of the routing options determined at this point is selected, and a routing decision is made. It is possible, that further entries may be examined, in order to determine other routing options. This sort of heuristic is not discussed here. When a single routing tree is used, the longest possible match based on the O/R address to be routed to is found. This entry, and then each of its parents in turn is considered, ending with the routing tree root node (except in the case of the open routing tree, which does not have such a node). When multiple routing trees are considered, the basic approach is to treat them in a defined order. This is supplemented by a mechanism whereby if a matched node cannot be used directly, the routing algorithm will have the choice to move up a level in the current routing tree, or to move on to the next routing tree with an option to move back to the first tree later. This option to move back is to allow for the common case where a tree is used to specify two things:

- 1. Routing information private to the MTA (e.g., local UAs or routing info for bilateral links).**
- 2. Default routing information for the case where other routing has failed.**

The actions allow for a tree to be followed, for the private

INTERNET--DRAFT

MHS Routing using Directory

June 1994

information, then for other trees to be used, and finally to fall back to the default situation. For very complex configurations it might be necessary to split this into two trees. The options defined by `routingFailureAction`, to be used when the information in the entry does not enable a direct route, are:

`next-level` Move up a level in the current routing tree. This is the action implied if the attribute is omitted. This will usually be the best action in the open community routing tree.

`next-tree-only` Move to the next tree, and do no further processing on the current tree. This will be useful optimisation for a routing tree where it is known that there is no useful additional routing information higher in the routing tree.

`next-tree-first` Move to the next tree, and then default back to the next level in this tree when all processing is completed on subsequent trees. This will be useful for an MTA to operate in the sequence:

1. Check for optimised private routes
2. Try other available information
3. Fall back to a local default route

`stop` This address is unroutable. No processing shall be done in any trees.

For the root entry of a routing tree, the default action and `next-level` are interpreted as `next-tree-only`.

[10.2](#) MTA Choice

This section considers how the choice between alternate MTAs is made. First, it is useful to consider the conditions why an MTA is entered into a node of the routing tree are:

- o The manager for the node of the tree shall place it there. This is a formality, but critical in terms of overall authority.

INTERNET--DRAFT

MHS Routing using Directory

June 1994

- o The MTA manager shall agree to it being placed there. For a well operated MTA, the access policy of the MTA will be set to enforce this.
- o The MTA will in general (for some class of message) be prepared to route to any valid O/R address in the subtree implied by the address. The only exception to this is where the MTA will route to a subset of the tree which cannot easily be expressed by making entries at the level below. An example might be an MTA prepared to route to all of the subtree, with certain explicit exceptions.

Information on each MTA is stored in an mTAInfo attribute. This attribute contains:

name The Distinguished Name of the MTA (Application Process)

weight A weighting factor (Route Weight) which gives a basis to choose between different MTAs. This is described in [Section 10.2](#).

mta-attributes Attributes from the MTAs entry. Information on the MTA will always be stored in the MTA's entry. The MTA is represented here as a structure, which enables some of this entry information to be represented in the routing node. This is effectively a maintained cache, and can lead to considerable performance optimisation. For example if ten MTAs were represented at a node, another MTA making a routing decision might need to make ten directory reads in order to obtain the information needed. If any attributes are present here, all of the attributes needed to make a routing decision shall be included, and also all attributes at the Application Entity level.

Where an MTA supports a single protocol only, or the protocols it supports have address information that can be represented in non-conflicting attributes, then the MTA may be represented as an application process only. In this case, the ae-info structure which gives information on associated application entities may be omitted, as the MTA is represented by a single application entity which has the same name as the application process. In other cases, the names of all application entities shall be included. A weight is associated with each application entity to allow the MTA to indicate a preference between its application entities.

INTERNET--DRAFT

MHS Routing using Directory

June 1994

ae-qualifier A printable string (e.g. ``x400-88''), which is the value of the common name of the relative distinguished name of the application entity. This can be used with the application process name to derive the application entity title.

ae-weight A weighting factor (Route Weight) which gives a basis to choose between different Application Entities (not between different MTAs). This is described in [Section 10.2](#).

ae-attributes Attributes from the AEs entry.

Route weighting is a mechanism to distinguish between different route choices. A routing weight may be associated with the MTA in the context of a routing tree entry. This is because routing weight will always be context dependent. This will allow machines which have other functions to be used as backup MTAs. The Route Weight is an integer in range 0--20. The lower the value, the better the choice of MTA. Where the weight is equal, and no other factors apply, the choice between the MTAs shall be random to facilitate load balancing. If the MTA itself is in the list, it shall only route to an MTA of lower weight. The exact values will be chosen by the manager of the relevant part of the routing tree. For guidance, three fixed points are given:

- o 0. For an MTA which can deliver directly to the entire subtree implied by the position in the routing tree.
- o 5. For an MTA which is preferred for this point in the subtree.
- o 10. For a backup MTA.

When an organisation registers in multiple routing trees, the route weight used is dependent on the context of the subtree. In general it is not possible to compare weights between subtrees. In some cases, use of route weighting can be used to divert traffic away from expensive links.

Attributes present in an MTA Entry are defined in various parts of this specification. A summary and pointers to these sections is given in [Section 16](#).

Attributes that are available in the MTA entry and will be needed for making a routing choice are:

INTERNET--DRAFT

MHS Routing using Directory

June 1994

protocolInformation

applicationContext

mhs-deliverable-content-length

responderAuthenticationRequirements

initiatorAuthenticationRequirements

responderPullingAuthenticationRequirements

initiatorPullingAuthenticationRequirements

initiatorP1Mode

responderP1Mode

polledMTAs Current MTA shall be in list if message is to be pulled.

mTAsAllowedToPoll

supportedMTSExtensions

If any MTA attributes are present in the mTAInfo attribute, all of the attribute that may affect routing choice shall be present. Other attributes may be present. A full list of MTA attributes, with summaries of their descriptions are given in [Section 16](#), with a formal definition in Figure 6 on Page 37.

[10.3](#) Routing Filters

This attribute provides for routing on information in the unmatched part of the O/R Address, including:

- o Routing on the basis of an O/R Address component type
- o Routing on the basis of a substring match of an O/R address component. This might be used to route X121 addressed faxes to an appropriate MTA.

INTERNET--DRAFT

MHS Routing using Directory

June 1994

When present, the procedures of analysing the routing filters shall be followed before other actions. The routing filter overrides mTAInfo and accessMD attributes. The components of the routingFilter attribute are:

attribute-type This gives the attribute type to be matched, and is selected from the attribute types which have not been matched to identify the routing entry. The filter applies to this attribute type. If there is no regular expression present (as defined below), the filter is true if the attribute is present. The value is the object identifier of the X.500 attribute type (e.g., at-prmd-name).

weight This gives the weight of the filter, which is encoded as a Route Weight, with lower values indicating higher priority. If multiple filters match, the weight of each matched filter is used to select between them. If the weight is the same, then a random choice shall be made.

dda-key If the attribute is domain defined, then this parameter may be used to identify the key.

regex-match This string is used to give a regular expression match on the attribute value. The syntax for regular expressions is defined in [Appendix E](#).

node This defines where to get routing information for the filter. It will be an entry with object class routingInformation, which can be used to determine the MTA or MTA choice.

An example of use of routing filters is now given, showing how to route on X121 address to a fax gateway in Germany. Consider the routing point.

PRMD=ABC, ADMD=XYZMail, C=GB

The entry associated would have two routing filters:

- 1. One with type x121 and no regular expression, to route a default**

fax gateway.

Kille

Expires: December 1994 Page 31

INTERNET--DRAFT

MHS Routing using Directory

June 1994

```
accessMD ATTRIBUTE ::= {  
    SUBTYPE OF distinguishedName  
    ID at-access-md}
```

Figure_4: Indirect Access

- 2. One with type x121 and a regular expression ^9262 to route all German faxes to a fax gateway located in Germany with which there is a bilateral agreement. This would have a lower weight, so that it would be selected over the default fax gateway.**

10.4 Indirect Connectivity

In some cases a part of the O/R Address space will be accessed indirectly. For example, an ADMD without access from the open community might have an agreement with another MD to provide this access. This is achieved by use of the accessMD attribute defined in Figure 4. If this attribute is found, the routing algorithm shall read the entry pointed to by this distinguished name and route according to the information retrieved, in order to route to this access MD.

The attribute is called an MD, as this is descriptive of its normal use. It might point to a more closely defined part of the O/R Address space.

It is possible for both access MD and MTAs to be specified. This might be done if the MTAs only support access over a restricted set of transport stacks. In this case, the access MD shall only be routed to if it is not possible to route to any of the MTAs.

This structure can also be used as an optimisation, where a set of MTAs provides access to several parts of the O/R Address space. Rather than repeat the MTA information (list of MTAs) in each reference to the MD, a single access MD is used as a means of grouping the MTAs. The value of the Distinguished Name of the access MD will probably not be meaningful in this case (e.g., it might be the name ``Access MTA List'', within the organisation.)

If the MTA routing is unable to access the information in the Access

MD dues to directory security restrictions, the routing algorithm shall continue as if no MTA information was located in the routing

Kille

Expires: December 1994 Page 32

INTERNET--DRAFT

MHS Routing using Directory

June 1994

entry.

11 Local Addresses (UAs)

Local addresses (UAs) are a special case for routing: the endpoint. The definition of the routedUA object class is given in Figure 5. This identifies a User Agent in a routing tree. This is needed for several reasons:

- 1. To allow UAs to be defined without having an entry in another part of the DIT.**
- 2. To identify which (leaf and non-leaf) nodes in a routing tree are User Agents. In a pure X.400 environment, a UA (as distinct from a connecting part of the O/R address space) is simply identified by object class. Thus an organisation entry can itself be a UA. A UA need not be a leaf, and can thus have children in the tree.**
- 3. To allow UA parameters as defined in X.402 (e.g., the mhs-deliverable-eits) to be determined efficiently from the routing tree, without having to go to the user's entry.**
- 4. To provide access to other information associated with the UA, as defined below.**

The following attributes are defined associated with the UA.

supportedExtensions

supportingMTA The MTAs which support a UA directly are noted in the SupportingMTA attribute, which may be multi-valued. In the X.400 model, only one MTA is associated with a UA. In practice, it is possible and useful for several MTAs to be able to deliver to a single UA. This attribute is a subtype of MTAinfo, as it is possible for an MTA to be registered to route to a UA, without it actually being able to deliver messages to it. This attribute shall be present, unless the address is being non-delivered or redirected.

mandatoryRedirect/filteredRedirect The attributes mandatoryRedirect and filteredRedirect control redirects, as described in

Kille

Expires: December 1994 Page 33

INTERNET--DRAFT

MHS Routing using Directory

June 1994

[Section 22.1.](#)

userName The attribute userName points to the distinguished Name of the user, as defined by the mhs-user in X.402. The pointer from the user to the O/R Address is achieved by the mhs-or-addresses attribute. This makes the UA/User linkage symmetrical.

nonDeliveryInfo The attribute nonDeliveryInfo mandates non-delivery to this address, as described in [Section 22.3](#).

When routing to a UA, an MTA will read the supportingMTA attribute. If it finds its own name present, it will know that the UA is local, and invoke appropriate procedures for local delivery (e.g., co-resident or P3 access information). The cost of holding these attributes for each UA at a site will often be reduced by use of shared attributes (as defined in X.500(93)).

The linkage between the UA and User entries was noted above. It is also possible to use a single entry for both User and UA, as there is no conflict between the attributes in each of the objects. In this case, the entries shall be in one part of the DIT, with aliases from the other. Because the UA and User are named with different attributes, the aliases shall be at the leaf level.

[11.1](#) Searching for Local Users

The approach defined in this specification performs all routing by use of reads. This is done for performance reasons, as it is a reasonable expectation that all DSA implementations will support a high performance read operation. For local routing only, an MTA in cooperation with the provider of the local routing tree may choose to use a search operation to perform routing. The major benefit of this is that there will not be a need to store aliases for alternate names, and so the directory storage requirement and alias management will be reduced. The difficulty with this approach is that it is hard to define search criteria that would be effective in all situations and well supported by all DUAs. There are also issues about determining the validity of a route on the basis of partial matches.

INTERNET--DRAFT

MHS Routing using Directory

June 1994

12 Direct Lookup

Where an O/R address is registered in the open community and has one or more ``open'' MTAs which support it, this will be optimised by storing MTA information in the O/R address entry. In general, the Directory will support this by use of attribute inheritance or an implementation will optimise the storage of repeated information, and so there will not be a large storage overhead implied. This is a function of the basic routing approach.

As a further optimisation of this case, the User's distinguished name entry may contain the mTAInfo attribute. This can be looked up from the distinguished name, and thus routing on submission can be achieved by use of a single read.

13 Alternate Routes

13.1 Finding Alternate Routes

The routing algorithm selects a single MTA to be routed to. It could be extended to find alternate routes to a single MTA with possibly different weights. How far this is done is a local configuration choice. Provision of backup routing is desirable, and leads to robust service, but excessive use of alternate routing is not usually beneficial. It will often force messages onto convoluted paths, when there was only a short outage on the preferred path.

It is important to note that this strategy will lead to picking the first acceptable route. It is important to configure the routing trees so that the first route identified will also be the best route.

13.2 Sharing routing information

So far, only single addresses have been considered. Improving routing choice for multiple addresses is analogous to dealing with multiple routes. This section defines an optional improvement. When multiple addresses are present, and alternate routes are available, the preferred routes may be chosen so as to maximise the number of

recipients sent with each message.

Specification of routing trees can facilitate this optimisation.
Suppose there is a set of addresses (e.g., in an organisation) which

Kille

Expires: December 1994 Page 35

INTERNET--DRAFT

MHS Routing using Directory

June 1994

have different MTAs, but have access to an MTA which will do local switching. If each address is registered with the optimal MTA as preferred, but has the ``hub'' MTA registered with a higher route weight, then optimisation may occur when a message is sent to multiple addresses in the group.

14 Looking up Information in the Directory

The description so far has been abstract about lookup of information. This section considers how information is looked up in the Directory. Consider that an O/R Address is presented for lookup, and there is a sequence of routing trees. At any point in the lookup sequence, there is one of a set of actions that can take place:

Handle MTA Info Information from the node shall be examined. If suitable information is found, one of the following is done:

- o Use the information and finish the routing process
- o Continue the routing process in order to find more options to choose from

Unroutable Address Potentially valid address, which cannot be routed

Bad Address Return the failure to the routing algorithm

Temporary Reject Try again later. The MTA shall handle this in a similar manner to the way it would handle the failure to connect to a remote MTA.

Permanent Reject Administrative error on the directory which may be fixed in future, but which currently prevents routing.

Editor's Note: Add information on what to do and which diagnostic codes and handling of failed lookups.

Errors from the lookup (directory read) shall be handled as follows:

AttributeError This leads to a permanent reject.

Kille

Expires: December 1994 Page 36

INTERNET--DRAFT

MHS Routing using Directory

June 1994

NameError The matched parameter is used to determine the number of components of the name that have matched (possibly zero). The read is then repeated with this name. This is the normal case, and allows the ``best'' entry in the routing tree to be located with two reads.

Referral The referral shall be followed.

SecurityError Strip one component of the O/R address and continue.

ServiceError This leads to a temporary reject (see above).

15 Naming MTAs

MTAs need to be named in the DIT, but the name does not have routing significance, it is simply a unique key. Attributes associated with naming MTAs are given in Figure 6. This figure also gives a list of attributes, which may be present in the MTA entry. The use of most of these is explained in subsequent sections. The mTAName and globalDomainID attributes are needed to define the information that an MTA places in trace information. As noted previously, an MTA is represented as an Application Process, with one or more Application Entities.

```
mTAName ATTRIBUTE ::= {  
    SUBTYPE OF name  
    WITH SYNTAX DirectoryString{ub-mta-name-length}  
    SINGLE VALUE  
    ID at-mta-name}  
    -- used for naming when  
    -- MTA is named in O=R Address Hierarchy
```

```
globalDomainID ATTRIBUTE ::= {  
    WITH SYNTAX GlobalDomainIdentifier  
    SINGLE VALUE  
    ID at-global-domain-id}
```

10

-- both attributes present when MTA
-- is named outside O=R Address Hierarchy
-- to enable trace to be written

mTAApplicationProcess OBJECT CLASS ::= {

Kille

Expires: December 1994 Page 37

INTERNET--DRAFT

MHS Routing using Directory

June 1994

SUBCLASS OF {application-process}

MAY CONTAIN {

20

mTAWillRoute|

globalDomainID|

routingTreeList|

accessMD|

localAccessUnit|

accessUnitsUsed

}

ID oc-mta-application-process}

mTA OBJECT CLASS ::= { -- Application Entity

30

SUBCLASS OF {mhs-message-transfer-agent}

MAY CONTAIN {

mTAName|

globalDomainID| -- per AE variant

responderAuthenticationRequirements|

initiatorAuthenticationRequirements|

responderPullingAuthenticationRequirements|

initiatorPullingAuthenticationRequirements|

initiatorP1Mode|

responderP1Mode|

40

polledMTAs|

protocolInformation|

respondingRTSCredentials|

initiatingRTSCredentials|

callingPresentationAddress|

callingSelectorValidity|

bilateralTable|

mTAWillRoute|

mhs-deliverable-content-length|

routingTreeList|

50

supportedMTSExtensions|

mTAsAllowedToPoll

}

ID oc-mta}

In X.400, MTAs are named by MD and a single string. This style of naming is supported, with MTAs named in the O/R Address tree relative to the root of the DIT (or possibly in a different routing tree). The mTAName attribute is used to name MTAs in this case. For X.400(88) it

Kille

Expires: December 1994 Page 38

INTERNET--DRAFT

MHS Routing using Directory

June 1994

is assumed that the Distinguished Name may be passed as an AE Title. MTAs may be named with any other DN, which can be in the O/R Address or Organisational DIT hierarchy. There are several reasons why MTAs might be named differently.

- o The flat naming space is inadequate to support large MDs. MTA name assignment using the directory would be awkward.
- o An MD does not wish to register its MTAs in this way (essentially, it prefers to give them private names in the directory).
- o An organisation has a policy for naming application processes, which does not fit this approach.

In this case, the MTA entry shall contain the correct information to be inserted in trace. The mTAName and globalDomainID attributes are used to do this. They are single value. For an MTA which inserts different trace in different circumstances, a more complex approach would be needed.

An MD may choose to name its MTAs outside of the O/R address hierarchy, and then link some or all of them with aliases. A pointer from this space may help in resolving information based on MTA Trace.

[15.1](#) Naming 1984 MTAs

Some simplifications are necessary for 1984 MTAs, and only one naming approach may be used. In X.400, MTAs are named by MD and a single string. This style of naming is supported, with MTAs named in the O/R Address tree relative to the root of the DIT (or possibly in a different routing tree). The MTAName attribute is used to name MTAs in this case.

[16](#) Attributes Associated with the MTA

This section lists the attributes which may be associated with an MTA as defined in Figure 6, and gives pointers to the sections that describe them.

mTAName [Section 15](#).

Kille

Expires: December 1994 Page 39

INTERNET--DRAFT

MHS Routing using Directory

June 1994

globalDomainID [Section 15](#).

protocolInformation [Section 18.1](#).

applicationContext [Section 18.2](#).

mhs-deliverable-content-length [Section 18.3](#).

responderAuthenticationRequirements [Section 20.2](#).

initiatorAuthenticationRequirements [Section 20.2](#).

responderPullingAuthenticationRequirements [Section 20.2](#).

initiatorPullingAuthenticationRequirements [Section 20.2](#).

initiatorP1Mode [Section 19](#).

responderP1Mode [Section 19](#).

polledMTAs [Section 19](#).

mTAsAllowedToPoll [Section 19](#).

respondingRTSCredentials [Section 20.3](#).

initiatingRTSCredentials [Section 20.3](#).

callingPresentationAddress [Section 20.3](#).

callingSelectorValidity [Section 20.3](#).

bilateralTable [Section 17](#).

mTAWillRoute [Section 21](#).

routingTreeList [Section 9](#).

supportedMTSExtensions [Section 18.3](#).

Kille

Expires: December 1994 Page 40

INTERNET--DRAFT

MHS Routing using Directory

June 1994

[17](#) Bilateral Agreements

Each MTA has an entry in the DIT. This will be information which is globally valid, and will be useful for handling general information about the MTA and for information common to all connections. In many cases, this will be all that is needed. This global information may be restricted by access control, and so need not be globally available. In some cases, MTAs will maintain bilateral and multilateral agreements, which hold authentication and related information which is not globally valid. This section describes a mechanism for grouping such information into tables, which enables an MTA to have bilateral information or for a group of MTAs to share multilateral information. The description is for bilateral information, but is equally applicable to multilateral agreements. For the purpose of a bilateral agreement, the MTA is considered to be an application entity. This means that when this is distinct from the application process, that the agreements are protocol specific.

A bilateral agreement is represented by one entry associated with each MTA participating in the bilateral agreement. For one end of the bilateral agreement, the agreement information will be keyed by the name of the MTA at the other end. Each party to the agreement will set up the entry which represents its half of the agreed policy. The fact that these correspond is controlled by the external agreement. In many cases, only one half of the agreement will be in the directory. The other half might be in an ADMD MTA configuration file. MTA bilateral information is stored in a table, as defined in [\[14\]](#). An MTA has access to a sequence of such tables, each of which controls agreements in both directions for a given MTA. Where an MTA is represented in multiple tables, the first agreement shall be used. This allows an MTA to participate in multilateral agreements, and to have private agreements which override these. The definition of entries in this table are defined in Figure 7. This table will usually be access controlled so that only a single MTA or selected MTAs which appear externally as one MTA can access it.

Each entry in the table is of the object class distinguishedNameTableEntry, which is used to name the entry by the distinguished name of the MTA. In some cases discussed in [Section 20.1](#), there will also be aliases of type textTableEntry. The MTA attributes needed as a part of the bilateral agreement (typically MTA Name/Password pairs), as described in [Section 20.3](#), will always be present. Other MTA attributes (e.g., presentation address) may be present for one of two reasons:

Kille

Expires: December 1994 Page 41

INTERNET--DRAFT

MHS Routing using Directory

June 1994

[1.](#) As a performance optimisation

[2.](#) Because the MTA does not have a global entry

Every MTA with bilateral agreements will define a bilateral MTA table. When a connection from a remote MTA is received, its Distinguished Name is used to generate the name of the table entry. For 1984, the MTA Name exchanged at the RTS level is used as a key into the table. The location of the bilateral tables used by the MTA and the order in which they are used are defined by the bilateralTable attribute in the MTA entry, which is defined in Figure 8.

All of the MTA information described in [Section 16](#) may be used in the bilateral table entries. This will allow bilateral control of a wide range of parameters.

Editor's Note: Need to add in control for various ghastly things such as trace stripping and originator munging. Whilst this is needed, it is perhaps best swept under the carpet and left to implementation specific extensions. This area will be reviewed in light of implementation experience.

[18](#) MTA Selection

[18.1](#) Dealing with protocol mismatches

MTAs may operate over different stacks. This means that some MTAs cannot talk directly to each other. Even where the protocols are the same, there may be reasons why a direct connection is not possible. An environment where there is full connectivity over a single stack is known as a transport community [\[9\]](#). The set of transport communities supported by an MTA is specified by use of the protocolInformation

attribute defined in X.500(93). This is represented as a separate attribute for the convenience of making routing decisions. A community is identified by an object identifier, and so the mechanism supports both well known and private communities. A list of object identifiers corresponding to well known communities is given in Appendix B.

Kille

Expires: December 1994 Page 42

INTERNET--DRAFT

MHS Routing using Directory

June 1994

18.2 Supported Protocols

It is important to know the protocol capabilities of an MTA. This is done by the application context. There are standard definitions for the following 1988 protocols.

- o P3 (with and without RTS, both user and MTS initiated)
- o P7 (with and without RTS).
- o P1 (various modes). Strictly, this is the only one that matters for routing.

In order to support P1(1984) and P1(1988) in X.410 mode, application contexts which define these protocols are given in [Appendix C](#). This context is for use in the directory only, and would never be exchanged over the network.

For routing purposes, a message store which is not co-resident with an MTA is represented as if it had a co-resident MTA and configured with a single link to its supporting MTA.

In cases where the UA is involved in exchanges, the UA will be of object class mhs-user-agent, and this will allow for appropriate communication information to be registered.

18.3 MTA Capability Restrictions

In addition to policy restrictions, described in [Section 21](#), an MTA may have capability restrictions. The maximum size of MPDU is defined by the standard attribute mhs-deliverable-content-length. The supported MTS extensions are defined by a new attribute specified in Figure 9.

It may be useful to define other capability restrictions, for example to enable routing of messages around MTAs with specific deficiencies. It has been suggested using MTA capabilities as an optimised means of expressing capabilities of all users associated with the MTA. This is felt to be undesirable.

18.4 Subtree Capability Restrictions

Kille

Expires: December 1994 Page 43

INTERNET--DRAFT

MHS Routing using Directory

June 1994

In many cases, users of a subtree will share the same capabilities. It is possible to specify this by use of attributes, as defined in Figure 10. This will allow for restrictions to be determined in cases where there is no entry for the user or O/R Address. This will be a useful optimisation in cases where the UA capability information is not available from the directory, either for policy reasons or because it is not there. This information may also be present in the domain tree ([RFC 822](#)).

This shall be implemented as a collective attribute, so that it is available to all entries in the subtree below the entry. This can also be used for default setting in the subtree.

19 MTA Pulling Messages

Pulling messages between MTAs, typically by use of two way alternate, is for bilateral agreement. It is not the common case. There are two circumstances in which it can arise.

1. Making use of a connection that was opened to push messages.

2. Explicitly polling in order to pull messages

Attributes to support this are defined in Figure 11. These attributes indicate the capabilities of an MTA to pull messages, and allows a list of polled MTAs to be specified. If omitted, the normal case of push-only is specified. In the MTA Entry, the polledMTAs attribute indicates MTAs which are to be polled and the mTAsAllowedToPoll attribute indicates MTAs that may poll the current MTA.

20 Security and Policy

20.1 Finding the Name of the Calling MTA

A key issue for authentication is for the called MTA to find the name of the calling MTA. This is needed for it to be able to look up information on a bilateral agreement.

Where X.400(88) is used, the name is available as a distinguished name from the AE-Title provided in the A-Associate. For X.400(84), it will not be possible to derive a global name from the bind. The MTA Name

Kille

Expires: December 1994 Page 44

INTERNET--DRAFT

MHS Routing using Directory

June 1994

exchanged in the RTS Bind will provide a key into the private bilateral agreement table (or tables). Thus for X.400(1984) it will only be possible to have bilateral inbound links or no authentication of the calling MTA.

Editor's Note CDC use a search here, as a mechanism to use a single table and an 88/84 independent access. This may be considered for general adoption. It appears to make the data model cleaner, possibly at the expense of some performance. This will be considered in the light of implementation experience.

20.2 Authentication

The levels of authentication required by an MTA will have an impact on routing. For example, if an MTA requires strong authentication, not all MTAs will be able to route to it. The attributes which define the authentication requirements are defined in Figure 12.

The attributes specify authentication levels for the following cases:

Responder These are the checks that the responder will make on the initiator's credentials.

Initiator These are the checks that the initiator will make on the responders credentials. Very often, no checks are needed --- establishing the connection is sufficient.

Responder Pulling These are responder checks when messages are pulled. These will often be stronger than for pushing.

Initiator Pulling For completeness.

If an attribute is omitted, no checks are required. If multiple checks are required, then each of the relevant bits shall be set. If there are alternative acceptable checks, then multiple values of the attribute are used.

The values of the authentication requirements mean:

mta-name-present That an RTS level MTA parameter shall be present for logging purposes.

Kille

Expires: December 1994 Page 45

INTERNET--DRAFT

MHS Routing using Directory

June 1994

aet-present That a distinguished name application entity title shall be provided at the ACSE level

aet-valid As for aet-present, and that the AET be registered in the directory. This may be looked up as a part of the validation process. If mta-name-present is set, the RTS value of mta and password shall correspond to those registered in the directory.

network-address This can only be used for the responder. The AET shall be looked up in the directory, and the callingPresentationAddress attribute matched against the calling address. This shall match exactly at the network level. The validity of selectors will be matched according to the callingSelectorValidity attribute.

simple-authentication All MTA and password parameters needed for simple authentication shall be used. This will usually be in conjunction with a bilateral agreement.

strong-authentication Use of strong authentication.

bilateral-agreement-needed This means that this MTA will only accept connections in conjunction with a bilateral agreement. This link cannot be used unless such an agreement exists.

These attributes may also be used to specify UA/MTA authentication policy. They may be resident in the UA entry in environments where this information cannot be modified by the user. Otherwise, it will be present in an MTA table (represented in the directory).

An MTA could choose to have different authentication levels related to different policies ([Section 21](#)). This is seen as too complex, and so they are kept independent. The equivalent function can always be

achieved by using multiple Application Entities with the application process.

20.3 Authentication Information

This section specifies connection information needed by P1. This is essentially RTS parameterisation needed for authentication. This is defined in Figure 13. Confidential bilateral information is implied by these attributes, and this will be held in the bilateral information agreement. This shall have appropriate access control

Kille

Expires: December 1994 Page 46

INTERNET--DRAFT

MHS Routing using Directory

June 1994

applied. Note that in some cases, MTA information will be split across a private and public entry.

The parameters are:

Initiating Credentials The credentials to be used when the local MTA initiates the association. It gives the credentials to insert into the request, and those expected in the response.

Responding Credentials The credentials to be used when the remote MTA initiates the association. It gives the credential expected in the request, and those to be inserted into the response.

Remote Presentation Address Valid presentation addresses, which the remote MTA may connect from.

If an MTA/Password pair is omitted. The MTA shall default to the local MTA Name, and the password to a zero length OCTET STRING.

Note: It may be useful to add more information here relating to parameters required for strong authentication.

21 Policy and Authorisation

21.1 Simple MTA Policy

The routing trees will generally be configured in order to identify MTAs which will route to the destination. A simple means is identified to specify an MTA's policy. This is defined in Figure 14. If this attribute is omitted, the MTA shall route all traffic to the

implied destinations from the context of the routing tree for any MTAs that have valid access to the routing tree.

The multi-valued attribute gives a set of policies which the MTA will route. O/R Addresses are represented by a prefix, which identifies a subtree. A distinguished name encoding of O/R Address is used. There are three components:

from This gives a set of O/R addresses which are granted permission by this attribute value. If omitted, ``all'' is implied.

Kille

Expires: December 1994 Page 47

INTERNET--DRAFT

MHS Routing using Directory

June 1994

to This gives the set of acceptable destinations. If omitted, ``all'' is implied.

from-excludes This defines (by prefix) subtrees of the O/R address tree which are explicitly excluded from the ``from'' definition. If omitted, there are no exclusions.

to-excludes This defines (by prefix) subtrees of the O/R address tree which are explicitly excluded from the ``to'' definition. If omitted, there are no exclusions.

This simple policy will suffice for most cases. In particular, it gives sufficient information for most real situations where a policy choice is forced, and the application of this policy would prevent a message being routed.

This simple prefixing approach does not deal explicitly with alias dereferencing. The prefixes refer to O/R addresses where aliases have been dereferenced. To match against these prefixes, O/R addresses being matched need to be ``normalised'' by being looked up in the directory to resolve alias values. If the lookup fails, it shall be assumed that the provided address is already normalised. This means that policy may be misinterpreted for parts of the DIT not referenced in the directory.

The originator refers to the MTS originator, and the recipient to the MTS recipient, following any list expansion or redirect.

This simple policy does not apply to delivery reports. Any advertised route shall work for delivery reports, and it does not make sense to regulate this on the basis of the sender.

Editor's Note: It would be useful to include some examples here.

[21.2](#) Complex MTA Policy

MTAs will generally have a much more complex policy mechanism, such as that provided by PP [\[17\]](#). Representing this as a part of the routing decision is not done here, but may be addressed in future versions. Some of the issues which need to be tackled are:

- o Use of charging and non-charging nets

Kille

Expires: December 1994 Page 48

INTERNET--DRAFT

MHS Routing using Directory

June 1994

- o Policy dependent on message size
- o Different policy for delivery reports.
- o Policy dependent on attributes of the originator or recipient(e.g., mail from students)
- o Content type and encoded information types
- o The path which the message has traversed to reach the MTA
- o MTA bilateral agreements
- o Pulling messages
- o Costs. This sort of policy information may also be for information only.

Where an MTA applies more complex policies, it is highly undesirable that they lead to non-routability of messages for MTAs routing using the published information.

Policies relating to submission do not need to be public. They can be private to the MTA.

[22](#) Delivery

[22.1](#) Redirects

There is a need to specify redirects in the Directory. This will be done at the O/R Address level (i.e., in a routing tree). This will be

useful for alternate names where an equivalent name (synonym) defined by an alias is not natural. An example where this might be appropriate is to redirect mail to a new O/R address where a user had changed organisation. The definitions are given in Figure 15.

Mandatory redirects are specified by the mandatoryRedirect attribute. A filtered redirect is provided, to allow small messages, large messages, or messages containing specific EITs or content to be redirected. Message size is measured in kBytes. Where multiple elements are contained in the FilteredRedirect structure, they are treated as having a ``logical or'' relationship.

Kille

Expires: December 1994 Page 49

INTERNET--DRAFT

MHS Routing using Directory

June 1994

When a delivery report is sent to an address which would be redirected, X.400 would ignore the redirect. This means that every O/R address would need to have a valid means of delivery. This would seem to be awkward to manage. Therefore, the redirect shall be followed, and the delivery report delivered to the redirected address. These redirects are handled directly by the MTA. Redirects can also be initiated by the UA, for example in the context of a P7 interaction.

[22.2](#) Underspecified O/R Addresses

[X.400](#) requires that some underspecified O/R Addresses are handled in a given way (e.g., if a surname is given without initials or given name). Where an underspecified O/R Address is to be treated as if it were another O/R Address, an alias shall be used. If the O/R Address is to be rejected as ambiguous, an entry shall be created in the DIT, and forced non-delivery specified for this reason.

[22.3](#) Non Delivery

It is possible for a manager to define an address to non-deliver with specified reason and diagnostic codes. This might be used for a range of management purposes. The attribute to do this is defined in Figure 16.

[22.4](#) Bad Addresses

If there is a bad address, it is desirable to do a directory search to find alternatives. This is a helpful user service and may be

supported. This function is invoked after address checking has failed, and where this is no user supplied alternate recipient. This function would be an MTA-chosen alternative to administratively assigned alternate recipient. Attributes to support handling of bad addresses are defined in Figure 17. The attributes are:

badAddressSearchPoint This gives the point (or list of points) from which to search.

badAddressSearchAttributes This gives the set of attribute types to search on. The default is common name.

Kille

Expires: December 1994 Page 50

INTERNET--DRAFT

MHS Routing using Directory

June 1994

Searches are always single level, and always use approximate match. If a small number of matches are made, this is returned to the originator by use of the per recipient `AlternativeAddresssInformation` in the delivery report (DR). This shall be marked non-critical, so that it will not cause the DR to be discarded (e.g., in downgrading to X.400(1984)). This attribute allows the Distinguished Name and O/R Address of possible alternate recipients to be returned with the delivery report. There is also the possibility to attach extra information in the form of directory attributes. Typically this might be used to return attributes of the entry which were matched in the search. A summary of the information shall also be returned using the delivery report supplementary information field (e.g., ``your message could not be delivered to smith, try J. Smith or P. Smith''), so that the information is available to user agents not supporting this extension. Note the length restriction of this field is 256 (ub-supplementary-info-length).

If the directory search fails, or there are no matches returned, a delivery report shall be returned as if this extra check had not been made.

Editor's Note: It might be useful to allow control of search type, and also single level vs subtree. This issue is for further study.

23 Submission

A message may be submitted with Distinguished Name only. If the MTA to which the message is submitted supports this service, this section

describes how the mapping is done.

23.1 Normal Derivation

The Distinguished Name is looked up to find the attribute mhs-or-addresses. If the attribute is single value, it is straightforward. If there are multiple values, one O/R address shall be selected at random.

Kille

Expires: December 1994 Page 51

INTERNET--DRAFT

MHS Routing using Directory

June 1994

23.2 Roles and Groups

Some support for roles is given. If there is no O/R address, and the entry is of object class role, then the roleOccupant attribute shall be dereferenced, and the message submitted to each of the role occupants. Similarly, if the entry is of object class group, where the groupMember attribute is used.

24 Access Units

Attributes needed for support of Access Units, as defined in X.400(88), are defined in Figure 18.

The attributes defined are:

localAccessUnit This defines the list of access units supported by the MTA.

accessUnitsUsed This defines which access units are used by the MTA, giving the type and MTA. An O/R Address filter is provided to control which access unit is used for a given recipient. For a filter to match an address, all attributes specified in the filter shall match the given address. This is specified as an O/R Address, so that routing to access units can be filtered on the basis of attributes not mapped onto the directory (e.g., postal attributes). Where a remote MTA is used, it may be necessary to use source routing.

Editor's Note 1: This mechanism might be used to replace the routefilter mechanism of the MTS routing. Comments are solicited.

Editor's Note 2: It has been proposed to add a more powerful filter mechanism. Comments are solicited.

Editor's Note 3: The utility of this specification as a mechanism to route faxes and other non MHS messages has been noted, but not explored. Comments as to how and if this should be developed are solicited.

These three issues are for further study.

Kille

Expires: December 1994 Page 52

INTERNET--DRAFT

MHS Routing using Directory

June 1994

[25](#) The Overall Routing Algorithm

Having provided all the pieces, a summary of how routing works can be given.

The core of the X.400 routing is described in [Section 10](#). A sequence of routing trees are followed. As nodes of the routing tree are matched, a set of MTAs will be identified for evaluation as possible next hops. If all of these are rejected, the trees are followed further². A set of MTAs is evaluated on the following criteria:

- o If an MTA is the local MTA, deliver locally.
- o Supported protocols. The MTA shall support a protocol that the current MTA supports³, as described in [Section 18.2](#).
- o The protocols shall share a common transport community, as described in [Section 18.1](#).
- o There shall be no capability restrictions in the MTA which prevents transfer of the current message, as described in [Section 18.3](#).
- o There shall be no policy restrictions in the MTA which prevents transfer of the current message, as described in [Section 21](#).
- o The authentication requirements of the MTA shall be met by the local MTA, as described in [Section 20.2](#).
- o If the authentication ([Section 20.2](#)) indicates that a bilateral

agreement is present, the MTA shall be listed in the local set of bilateral agreements, as described in [Section 17](#).

- o In cases where the recipient UA's capabilities can be determined, there should either be no mismatch, or there shall be an ability to use local or remote reformatting capabilities, as described

2. It might be argued that the trees should be followed to find alternate routes in the case that only one MTA is acceptable. This is not proposed.

3. Note that this could be an [RFC 822](#) protocol, as well as an [X.400](#) protocol.

Kille

Expires: December 1994 Page 53

INTERNET--DRAFT

MHS Routing using Directory

June 1994

in[11].

[26](#) Performance

The routing algorithm has been designed with performance in mind. In particular, care has been taken to use only the read function, which will in general be optimised. Routing trees may be configured so that routing decisions can be made with only two directory reads. More complex configurations will not require a substantially larger number of operations.

[27](#) Acknowledgements

This INTERNET--DRAFT is the central document of a series of specifications [[13](#), [14](#), [16](#), [18](#), [12](#), [15](#), [11](#)]. The acknowledgements for all of this work is given here. Previous work, which significantly influenced these specifications is described in [Section 3](#). This led to an initial proposal by the editor, which was subsequently split into eight documents.

Work on these specifications has been done by the IETF MHS-DS working group. Special credit is given to the joint chairs of this group: Harald Alvestrand (Uninett) and Kevin Jordan (CDC). Credit is given to all members of the WG. Those who have made active contribution include: Piete Brooks (Cambridge University); Allan Cargille (University of Wisconsin); Jim Craigie (JNT); Dennis Doyle (SSS); Urs Eppenberger (SWITCH); Peter Furniss; Christian Huitema (Inria); Marko

Kaittola (Dante); Sylvain Langlois (EDF); Lucy Loftin (AT&T GIS); Julian Onions (NEXOR); Paul-Andre Pays (Inria); Colin Robbins (NEXOR); Michael Roe (Cambridge University); Jim Romaguera (Netconsult); Michael Storz (Leibniz Rechenzentrum); Mark Wahl (ISODE Consortium); Alan Young (ISODE Consortium).

This work was partly funded by the COSINE Paradise project.

References

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

[2] J.N. Chiappa. A new IP routing and addressing architecture,

Kille

Expires: December 1994 Page 54

INTERNET--DRAFT

MHS Routing using Directory

June 1994

1991.

[3] A. Consael, M. Tschicholz, O. Wenzel, K. Bonacker, and M. Busch. DFN-Directory nutzung durch MHS, April 1990. GMD Report.

[4] P. Dick-Lauder, R.J. Kummerfeld, and K.R. Elz. ACSNet - the Australian alternative to UUCP. In EUUG Conference, Paris, pages 60--69, April 1985.

[5] U. Eppenberger. Routing coordination for an X.400 MHS service within a multi protocol / multinetwork environment. [RFC 1465](#).

[6] K.E. Jordan. Using X.500 directory services in support of X.400 routing and address mapping, November 1991. Private Note.

[7] S.E. Kille. MHS use of directory service for routing. In IFIP 6.5 Conference on Message Handling, Munich, pages 157--164. North Holland Publishing, April 1987.

[8] S.E. Kille. Topology and routing for MHS. COSINE Specification Phase 7.7, RARE, 1988.

[9] S.E. Kille. Encoding network addresses to support operation over non-OSI lower layers. Request for Comments [RFC 1277](#), Department of Computer Science, University College London, November 1991.

[10] S.E. Kille. A string representation of distinguished name. Request for Comments in preparation, Department of Computer

Science, University College London, January 1992.

- [11] S.E. Kille. Mhs use of directory to support mhs content conversion, July 1993. Internet Draft.
- [12] S.E. Kille. Use of the directory to support routing for [RFC 822](#) and related protocols, July 1993. Internet Draft.
- [13] S.E. Kille. Representing tables and subtrees in the directory, June 1994. Internet Draft.
- [14] S.E. Kille. Representing the O/R Address hierarchy in the directory information tree, June 1994. Internet Draft.
- [15] S.E. Kille. A simple profile for MHS use of directory, March 1994. Internet Draft.

Kille

Expires: December 1994 Page 55

INTERNET--DRAFT

MHS Routing using Directory

June 1994

- [16] S.E. Kille. Use of the directory to support mapping between X.400 and [RFC 822](#) addresses, June 1994. Internet Draft.
- [17] S.E. Kille and J.P. Onions. The PP manual, December 1991. Version 6.0.
- [18] S.E. Kille and C.J. Robbins. MHS use of the directory to support distribution lists, March 1994. Internet Draft.
- [19] P. Lauder, R.J. Kummerfeld, and A. Fekete. Hierarchical network routing. In Tricom 91, 1991.
- [20] CCITT recommendations X.400 / ISO 10021, April 1988. CCITT SG 5/VII / ISO/IEC JTC1, Message Handling: System and Service Overview.
- [21] Zen and the ART of navigating through the dark and murky regions of the message transfer system: Working document on MTS routing, September 1991. ISO SC 18 SWG Messaging.

[28](#) Security Considerations

Security considerations are not discussed in this INTERNET--DRAFT.

[29](#) 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;

Kille

Expires: December 1994 Page 56

INTERNET--DRAFT

MHS Routing using Directory

June 1994

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)}

routing OBJECT IDENTIFIER ::= {mhs-ds 3}

oc OBJECT IDENTIFIER ::= {routing 1}

at OBJECT IDENTIFIER ::= {routing 2}

id OBJECT IDENTIFIER ::= {routing 3}

10

oc-mta OBJECT IDENTIFIER ::= {oc 1}

oc-mta-bilateral-table-entry OBJECT IDENTIFIER ::= {oc 2}

oc-routing-information OBJECT IDENTIFIER ::= {oc 3}

oc-restricted-subtree OBJECT IDENTIFIER ::= {oc 4}

oc-routed-ua OBJECT IDENTIFIER ::= {oc 5}

oc-routing-tree-root OBJECT IDENTIFIER ::= {oc 6}

oc-mta-application-process OBJECT IDENTIFIER ::= {oc 7}

at-access-md OBJECT IDENTIFIER ::= {at 1}

at-access-units-used OBJECT IDENTIFIER ::= {at 2}

20

at-subtree-information OBJECT IDENTIFIER ::= {at 3}

at-bad-address-search-attributes OBJECT IDENTIFIER ::= {at 4}

at-bad-address-search-point OBJECT IDENTIFIER ::= {at 5}

at-calling-selector-validity OBJECT IDENTIFIER ::= {at 7}

at-filtered-redirect OBJECT IDENTIFIER ::= {at 38}
 at-global-domain-id OBJECT IDENTIFIER ::= {at 10}
 at-initiating-rts-credentials OBJECT IDENTIFIER ::= {at 11}
 at-initiator-authentication-requirements OBJECT IDENTIFIER ::= {at 12}30
 at-initiator-p1-mode OBJECT IDENTIFIER ::= {at 13}
 at-initiator-pulling-authentication-requirements OBJECT IDENTIFIER ::= {at 14}
 at-local-access-unit OBJECT IDENTIFIER ::= {at 15}
 at-mandatory-redirect OBJECT IDENTIFIER ::= {at 16}
 at-mta-info OBJECT IDENTIFIER ::= {at 40}
 at-mta-name OBJECT IDENTIFIER ::= {at 19}

 at-mta-will-route OBJECT IDENTIFIER ::= {at 21}
 at-calling-presentation-address OBJECT IDENTIFIER ::= {at 22}

Kille

Expires: December 1994 Page 58

INTERNET--DRAFT

MHS Routing using Directory

June 1994

at-responder-authentication-requirements OBJECT IDENTIFIER ::= {at 23}40
 at-responder-p1-mode OBJECT IDENTIFIER ::= {at 24}
 at-responder-pulling-authentication-requirements OBJECT IDENTIFIER ::= {at 25}
 at-responding-rts-credentials OBJECT IDENTIFIER ::= {at 26}
 at-routing-failure-action OBJECT IDENTIFIER ::= {at 27}
 at-routing-filter OBJECT IDENTIFIER ::= {at 28}
 at-routing-tree-list OBJECT IDENTIFIER ::= {at 29}
 at-subtree-deliverable-content-length OBJECT IDENTIFIER ::= {at 30}
 at-subtree-deliverable-content-types OBJECT IDENTIFIER ::= {at 31}
 at-subtree-deliverable-eits OBJECT IDENTIFIER ::= {at 32}
 at-supporting-mta OBJECT IDENTIFIER ::= {at 33} 50
 at-transport-community OBJECT IDENTIFIER ::= {at 34}
 at-user-name OBJECT IDENTIFIER ::= {at 35}
 at-non-delivery-info OBJECT IDENTIFIER ::= {at 36}
 at-polled-mtas OBJECT IDENTIFIER ::= {at 37}
 at-bilateral-table OBJECT IDENTIFIER {at 41}
 at-supported-extension OBJECT IDENTIFIER {at 42}
 at-supported-mts-extension OBJECT IDENTIFIER {at 43}
 at-mtas-allowed-to-poll OBJECT IDENTIFIER {at 44}

 id-alternative-address-information OBJECT IDENTIFIER ::= {id 1} 60

Kille

Expires: December 1994 Page 59

INTERNET--DRAFT

MHS Routing using Directory

June 1994

B Community Identifier Assignments

Kille

Expires: December 1994 Page 60

INTERNET--DRAFT

MHS Routing using Directory

June 1994

C Protocol Identifier Assignments

INTERNET - - DRAFT

MHS Routing using Directory

June 1994

D ASN.1 Summary

MHS-DS-Definitions

DEFINITIONS ::=

BEGIN

```
-- assign OID to module
-- define imports and exports
```

```

routingTreeRoot OBJECT-CLASS ::= {
    SUBCLASS OF {routingInformation|subtree}
    ID oc-routing-tree-root}

```

```
routingTreeList ATTRIBUTE ::= {
    WITH SYNTAX RoutingTreeList
    SINGLE VALUE
    ID at-routing-tree-list}
```

```
RoutingTreeList ::= SEQUENCE OF RoutingTreeName
```

RoutingTreeName ::= DistinguishedName 20

```
routingInformation OBJECT-CLASS ::= {
    SUBCLASS OF top
    MAY CONTAIN {
        subtreeInformation|
        routingFilter|
        routingFailureAction|
        mTAInfo|
        accessMD|
        nonDeliveryInfo|
        badAddressSearchPoint|
    }
}
```

```

        badAddressSearchAttributes}
ID oc-routing-information}
    -- No naming attributes as this is not a
    -- structural object class

```

```

subtreeInformation ATTRIBUTE ::= {
    WITH SYNTAX SubtreeInfo
    SINGLE VALUE
    ID at-subtree-information}

```

40

Kille Expires: December 1994 Page 62

INTERNET--DRAFT MHS Routing using Directory June 1994

```

SubtreeInfo ::= ENUMERATED {
    all-children-present(0),
    not-all-children-present(1) }

```

```

routingFilter ATTRIBUTE ::= {
    WITH SYNTAX RoutingFilter
    ID at-routing-filter}

```

50

```

RoutingFilter ::= SEQUENCE{
    attribute-type OBJECT-IDENTIFIER,
    weight RouteWeight,
    dda-key String OPTIONAL,
    regex-match IA5String OPTIONAL,
    node DistinguishedName }

```

```

String ::= CHOICE {PrintableString, TeletexString}

```

60

```

routingFailureAction ATTRIBUTE ::= {
    WITH SYNTAX RoutingFailureAction
    SINGLE VALUE
    ID at-routing-failure-action}

```

```

RoutingFailureAction ::= ENUMERATED {
    next-level(0),
    next-tree-only(1),
    next-tree-first(2),
    stop(3) }

```

70


```

mTAInfo ATTRIBUTE ::= {
    WITH SYNTAX MTAInfo
    ID at-mta-info}

MTAInfo ::= SEQUENCE {
    name DistinguishedName,
    weight [1] RouteWeight DEFAULT preferred-access,      80
    mta-attributes [2] SET OF Attribute OPTIONAL,
    ae-info SEQUENCE OF SEQUENCE {
        aEqualifier PrintableString,
        ae-weight RouteWeight DEFAULT preferred-access,
        ae-attributes SET OF Attribute OPTIONAL} OPTIONAL
}

```

Kille Expires: December 1994 Page 63

INTERNET--DRAFT MHS Routing using Directory June 1994

```

}

RouteWeight ::= INTEGER {endpoint(0),
    preferred-access(5),
    backup(10)} (0..20)      90

```

```

accessMD ATTRIBUTE ::= {
    SUBTYPE OF distinguishedName
    ID at-access-md}

```

```

routedUA OBJECT-CLASS ::= {
    SUBCLASS OF {routingInformation}
    MAY CONTAIN {
        -- from X.402
        mhs-deliverable-content-length|      100
        mhs-deliverable-content-types|
        mhs-deliverable-eits|
        mhs-message-store|
        mhs-preferred-delivery-methods|
        -- defined here
        supportedExtensions|
        mandatoryRedirect|
        supportingMTA|
        filteredRedirect|
        userName|                             110
        nonDeliveryInfo}
    ID oc-routed-ua}

```

```

supportedExtensions ATTRIBUTE ::= {
    SUBTYPE OF objectIdentifier
    ID at-supported-extensions}

```

```

supportingMTA ATTRIBUTE ::= {
    SUBTYPE OF mtaInfo
    ID at-supporting-mta}

```

120

```

userName ATTRIBUTE ::= {
    SUBTYPE OF distinguishedName
    ID at-user-name}

```

```

mTAName ATTRIBUTE ::= {
    SUBTYPE OF name
    WITH SYNTAX DirectoryString{ub-mta-name-length}
    SINGLE VALUE

```

Kille Expires: December 1994 Page 64

INTERNET--DRAFT MHS Routing using Directory June 1994

```

ID at-mta-name}
-- used for naming when
-- MTA is named in O=R Address Hierarchy

```

130

```

globalDomainID ATTRIBUTE ::= {
    WITH SYNTAX GlobalDomainIdentifier
    SINGLE VALUE
    ID at-global-domain-id}
-- both attributes present when MTA
-- is named outside O=R Address Hierarchy
-- to enable trace to be written

```

140

```

mTAApplicationProcess OBJECT CLASS ::= {
    SUBCLASS OF {application-process}
    MAY CONTAIN {
        mTAWillRoute|
        globalDomainID|
        routingTreeList|
        accessMD|
        localAccessUnit|
        accessUnitsUsed
    }
    ID oc-mta-application-process}

```

150

```

mTA OBJECT CLASS ::= { -- Application Entity
    SUBCLASS OF {mhs-message-transfer-agent}
    MAY CONTAIN {
        mTAName|
        globalDomainID| -- per AE variant
        responderAuthenticationRequirements|

```

```

initiatorAuthenticationRequirements|
responderPullingAuthenticationRequirements|
initiatorPullingAuthenticationRequirements|
initiatorP1Mode|
responderP1Mode|
polledMTAs|
protocolInformation|
respondingRTSCredentials|
initiatingRTSCredentials|
callingPresentationAddress|
callingSelectorValidity|
bilateralTable|
mTAWillRoute|
mhs-deliverable-content-length|

```

Kille Expires: December 1994 Page 65

INTERNET--DRAFT MHS Routing using Directory June 1994

```

routingTreeList|
supportedMTSExtensions|
mTAsAllowedToPoll
}
ID oc-mta}

```

```

mTABilateralTableEntry OBJECT-CLASS ::=
    SUBCLASS OF {mTA| distinguishedNameTableEntry}
    ID oc-mta-bilateral-table-entry}

```

```

bilateralTable ATTRIBUTE ::= {
    WITH SYNTAX SEQUENCE OF DistinguishedName
    SINGLE VALUE
    ID at-bilateral-table}

```

```

supportedMTSExtensions ATTRIBUTE ::= {
    SUBTYPE OF objectIdentifier
    ID at-supported-mts-extensions}

```

```

restrictedSubtree OBJECT-CLASS ::= {
    SUBCLASS OF {top}
    MAY CONTAIN {
        subtreeDeliverableContentLength|
        subtreeDeliverableContentTypes|
        subtreeDeliverableEITs}
    ID oc-restricted-subtree}

```

```

subtreeDeliverableContentLength ATTRIBUTE ::= {
    SUBTYPE OF mhs-deliverable-content-length

```

200

ID at-subtree-deliverable-content-length}

subtreeDeliverableContentTypes ATTRIBUTE ::= {
 SUBTYPE OF mhs-deliverable-content-types
 ID at-subtree-deliverable-content-types}

subtreeDeliverableEITs ATTRIBUTE ::= {
 SUBTYPE OF mhs-deliverable-eits 210
 ID at-subtree-deliverable-eits}

initiatorP1Mode ATTRIBUTE ::= {
 WITH SYNTAX P1Mode
 SINGLE VALUE
 ID at-initiator-p1-mode}

Kille Expires: December 1994 Page 66

INTERNET--DRAFT MHS Routing using Directory June 1994

responderP1Mode ATTRIBUTE ::= {
 WITH SYNTAX P1Mode 220
 SINGLE VALUE
 ID at-responder-p1-mode}

P1Mode ::= ENUMERATED {
 push-only(0),
 pull-only(1),
 tw(2) }

polledMTAs ATTRIBUTE ::= {
 WITH SYNTAX PolledMTAs 230
 ID at-polled-mtas}

PolledMTAs ::= SEQUENCE {
 mta DistinguishedName,
 poll-frequency INTEGER OPTIONAL --frequency in minutes
}

mTAsAllowedToPoll ATTRIBUTE ::= {
 SUBTYPE OF distinguishedName
 ID at-mtas-allowed-to-poll} 240

responderAuthenticationRequirements ATTRIBUTE ::= {
 WITH SYNTAX AuthenticationRequirements
 SINGLE VALUE

ID at-responder-authentication-requirements}

initiatorAuthenticationRequirements ATTRIBUTE ::= {
 WITH SYNTAX AuthenticationRequirements
 SINGLE VALUE
 ID at-initiator-authentication-requirements} 250

repsonderPullingAuthenticationRequirements ATTRIBUTE ::= {
 WITH SYNTAX AuthenticationRequirements
 SINGLE VALUE
 ID at-responder-pulling-authentication-requirements}

initiatorPullingAuthenticationRequirements ATTRIBUTE ::= {
 WITH SYNTAX AuthenticationRequirements
 SINGLE VALUE
 ID at-initiator-pulling-authentication-requirements} 260

Kille Expires: December 1994 Page 67

INTERNET--DRAFT MHS Routing using Directory June 1994

AuthenticationRequirements ::= BITSTRING {
 mta-name-present(0),
 aet-present(1),
 aet-valid(2),
 network-address(3),
 simple-authentication(4),
 strong-authentication(5),
 bilateral-agreement-needed(6)} 270

respondingRTSCredentials ATTRIBUTE ::= {
 WITH SYNTAX RTSCredentials
 SINGLE VALUE
 ID at-responding-rts-credentials}

initiatingRTSCredentials ATTRIBUTE ::= {
 WITH SYNTAX RTSCredentials
 SINGLE VALUE
 ID at-initiating-rts-credentials} 280

RTSCredentials ::= SEQUENCE {
 request [0] MTAandPassword OPTIONAL,
 response [1] MTAandPassword OPTIONAL }

```
MTAandPassword ::= SEQUENCE {
    MTAName,
    Password }          -- MTAName and Password
                        -- from X.411
```

290

```
callingPresentationAddress ATTRIBUTE ::= {
    SUBTYPE OF presentationAddress
    MULTI VALUE
    ID at-calling-presentation-address}
```

```
callingSelectorValidity ATTRIBUTE ::= {
    WITH SYNTAX CallingSelectorValidity
    SINGLE VALUE
    ID at-calling-selector-validity}
```

300

```
CallingSelectorValidity ::= ENUMERATED {
```

Kille Expires: December 1994 Page 68

INTERNET--DRAFT MHS Routing using Directory June 1994

```
all-selectors-fixed(0),
tsel-may-vary(1),
all-selectors-may-vary(2) }
```

```
mTAWillRoute ATTRIBUTE ::= {
    WITH SYNTAX MTAWillRoute
    ID at-mta-will-route}
```

310

```
MTAWillRoute ::= SEQUENCE {
    from [0]          SET OF ORAddressPrefix OPTIONAL,
    to [1]            SET OF ORAddressPrefix OPTIONAL,
    from-excludes [2] SET OF ORAddressPrefix OPTIONAL,
    to-excludes [3]   SET OF ORAddressPrefix OPTIONAL }
```

```
ORAddressPrefix ::= DistinguishedName
```

320

```
mandatoryRedirect ATTRIBUTE ::= {
    SUBTYPE OF distinguishedName
    SINGLE VALUE
    ID at-mandatory-redirect}
```

```
filteredRedirect ATTRIBUTE ::= {
    WITH SYNTAX FilteredRedirect
    SINGLE VALUE
    ID at-filtered-redirect}
```

330

```

FilteredRedirect ::= SEQUENCE {
    redirect-to DistinguishedName,
    SEQUENCE {
        min-size [1] INTEGER,
        max-size [2] INTEGER,
        content [3] ContentType,
        eit [4] ExternalEncodedInformationType }
}

```

340

```

nonDeliveryInfo ATTRIBUTE ::= {
    WITH SYNTAX NonDeliveryReason
    ID at-non-delivery-info}

NonDeliveryReason ::= SEQUENCE {
    reason INTEGER (0..ub-reason-codes),

```

Kille Expires: December 1994 Page 69

INTERNET--DRAFT MHS Routing using Directory June 1994

```

    diagnostic INTEGER (0..ub-diagnostic-codes),
    supplementaryInfo PrintableString }

```

350

```

badAddressSearchPoint ATTRIBUTE ::= {
    SUBTYPE OF distinguishedName
    ID at-bad-address-search-point}

```

```

badAddressSearchAttributes ATTRIBUTE ::= {
    WITH SYNTAX AttributeType
    ID at-bad-address-search-attributes}

```

360

```

alternativeAddressInformation EXTENSION
    AlternativeAddressInformation
    ::= id-alternative-address-information
    -- X.400(92) continues to use MACRO notation

```

```

AlternativeAddressInformation ::= SET OF SEQUENCE {
    distinguished-name DistinguishedName OPTIONAL,
    or-address ORAddress OPTIONAL,
    other-useful-info SET OF Attribute }

```

370

```

localAccessUnit ATTRIBUTE ::= {
    WITH SYNTAX AccessUnitType
    ID at-local-access-unit}

```

```

AccessUnitType ::= ENUMERATED {
    fax (1),
    physical-delivery (2),
    teletex (3),
    telex (4) }
380

accessUnitsUsed ATTRIBUTE ::= {
    WITH SYNTAX SelectedAccessUnit
    ID at-access-units-used}

SelectedAccessUnit ::= SEQUENCE {
    type AccessUnitType,
    providing-MTA DistinguishedName,
    filter SET OF ORAddress OPTIONAL }
mhs-ds OBJECT-IDENTIFIER ::= {iso(1) org(3) dod(6) internet(1) private(4)
    enterprises(1) isode-consortium (453) mhs-ds (7)} 390

routing OBJECT IDENTIFIER ::= {mhs-ds 3}

```

Kille Expires: December 1994 Page 70

INTERNET--DRAFT MHS Routing using Directory June 1994

```

oc OBJECT IDENTIFIER ::= {routing 1}
at OBJECT IDENTIFIER ::= {routing 2}
id OBJECT IDENTIFIER ::= {routing 3}

oc-mta OBJECT IDENTIFIER ::= {oc 1}
oc-mta-bilateral-table-entry OBJECT IDENTIFIER ::= {oc 2} 400
oc-routing-information OBJECT IDENTIFIER ::= {oc 3}
oc-restricted-subtree OBJECT IDENTIFIER ::= {oc 4}
oc-routed-ua OBJECT IDENTIFIER ::= {oc 5}
oc-routing-tree-root OBJECT IDENTIFIER ::= {oc 6}
oc-mta-application-process OBJECT IDENTIFIER ::= {oc 7}

at-access-md OBJECT IDENTIFIER ::= {at 1}
at-access-units-used OBJECT IDENTIFIER ::= {at 2}
at-subtree-information OBJECT IDENTIFIER ::= {at 3}
at-bad-address-search-attributes OBJECT IDENTIFIER ::= {at 4} 410
at-bad-address-search-point OBJECT IDENTIFIER ::= {at 5}

at-calling-selector-validity OBJECT IDENTIFIER ::= {at 7}

at-filtered-redirect OBJECT IDENTIFIER ::= {at 38}
at-global-domain-id OBJECT IDENTIFIER ::= {at 10}
at-initiating-rtts-credentials OBJECT IDENTIFIER ::= {at 11}

```


at-initiator-authentication-requirements OBJECT IDENTIFIER ::= {at 12}
at-initiator-p1-mode OBJECT IDENTIFIER ::= {at 13}
at-initiator-pulling-authentication-requirements OBJECT IDENTIFIER ::=4{at214}0
at-local-access-unit OBJECT IDENTIFIER ::= {at 15}
at-mandatory-redirect OBJECT IDENTIFIER ::= {at 16}
at-mta-info OBJECT IDENTIFIER ::= {at 40}
at-mta-name OBJECT IDENTIFIER ::= {at 19}

at-mta-will-route OBJECT IDENTIFIER ::= {at 21}
at-calling-presentation-address OBJECT IDENTIFIER ::= {at 22}
at-responder-authentication-requirements OBJECT IDENTIFIER ::= {at 23}
at-responder-p1-mode OBJECT IDENTIFIER ::= {at 24}
at-responder-pulling-authentication-requirements OBJECT IDENTIFIER ::=4{at325}0
at-responding-rts-credentials OBJECT IDENTIFIER ::= {at 26}
at-routing-failure-action OBJECT IDENTIFIER ::= {at 27}
at-routing-filter OBJECT IDENTIFIER ::= {at 28}
at-routing-tree-list OBJECT IDENTIFIER ::= {at 29}
at-subtree-deliverable-content-length OBJECT IDENTIFIER ::= {at 30}
at-subtree-deliverable-content-types OBJECT IDENTIFIER ::= {at 31}
at-subtree-deliverable-eits OBJECT IDENTIFIER ::= {at 32}

Kille

Expires: December 1994 Page 71

INTERNET--DRAFT

MHS Routing using Directory

June 1994

at-supporting-mta OBJECT IDENTIFIER ::= {at 33}
at-transport-community OBJECT IDENTIFIER ::= {at 34}
at-user-name OBJECT IDENTIFIER ::= {at 35} 440
at-non-delivery-info OBJECT IDENTIFIER ::= {at 36}
at-polled-mtas OBJECT IDENTIFIER ::= {at 37}
at-bilateral-table OBJECT IDENTIFIER {at 41}
at-supported-extension OBJECT IDENTIFIER {at 42}
at-supported-mts-extension OBJECT IDENTIFIER {at 43}
at-mtas-allowed-to-poll OBJECT IDENTIFIER {at 44}

id-alternative-address-information OBJECT IDENTIFIER ::= {id 1}

ts-communities OBJECT-IDENTIFIER ::= {iso(1) org(3) dod(6)
internet(1)4private(4)50
enterprises(1) isode-consortium (453) ts-communities (4)}

tc-cons OBJECT IDENTIFIER ::= {ts-communities 1}	-- OSI CONS
tc-clns OBJECT IDENTIFIER ::= {ts-communities 2}	-- OSI CLNS
tc-internet OBJECT IDENTIFIER ::= {ts-communities 3}	-- Internet + RFC 1006
tc-int-x25 OBJECT IDENTIFIER ::= {ts-communities 4}	-- International X.25
	-- Without CONS
tc-ixi OBJECT IDENTIFIER ::= {ts-communities 5}	-- IXI (Europe)

```

tc-janet OBJECT IDENTIFIER ::= {ts-communities 6}          -- Janet (UK)460

mail-protocol OBJECT-IDENTIFIER ::= {iso(1) org(3) dod(6) internet(1)
private(4)
    enterprises(1) isode-consortium (453) mail-protocol (5)}

ac-p1-1984 OBJECT IDENTIFIER ::= {mail-protocol 1}         -- p1(1984)
ac-smtp  OBJECT IDENTIFIER ::= {mail-protocol 2}           -- SMTP
ac-uucp  OBJECT IDENTIFIER ::= {mail-protocol 3}           -- UUCP Mail
ac-jnt-mail OBJECT IDENTIFIER ::= {mail-protocol 4}        -- JNT Mail (UK)
ac-p1-1988-x410 OBJECT IDENTIFIER ::= {mail-protocol 5} -- p1(1988) in X.410
mode
ac-p3-1984 OBJECT IDENTIFIER ::= {mail-protocol 6}         -- p3(1984)470
END

```

Figure 22: ASN.1 Summary

Kille

Expires: December 1994 Page 72

INTERNET--DRAFT

MHS Routing using Directory

June 1994

E Regular Expression Syntax

This appendix defines a form of regular expression for pattern matching. This pattern matching is derived from the UNIX `ed(1)` command. The matching is modified to be case insensitive.

A regular expression (RE) specifies a set of character strings to match against - such as ``any string containing digits 5 through 9''. A member of this set of strings is said to be matched by the regular expression.

Where multiple matches are present in a line, a regular expression matches the longest of the leftmost matching strings.

Regular expressions can be built up from the following ``single-character'' RE's:

- c Any ordinary character not listed below. An ordinary character matches itself.
- \ Backslash. When followed by a special character, the RE matches the "quoted" character. A backslash followed by one of (or) represents an operator in a regular expression, as described below.
- .
- Dot. Matches any single character.
- ^ As the leftmost character, a caret (or circumflex) constrains the RE to match the leftmost portion of a string. A match of this type is called an "anchored match" because it is "anchored" to a specific place in the string. The ^ character loses its special meaning if it appears in any position other than the start of the RE.
- \$ As the rightmost character, a dollar sign constrains the RE to match the rightmost portion of a string. The \$ character loses its special meaning if it appears in any position other than at the end of the RE.

^RE\$ The construction ^RE\$ constrains the RE to match the

Kille

Expires: December 1994 Page 73

INTERNET--DRAFT

MHS Routing using Directory

June 1994

entire string.

[c...]

A nonempty string of characters, enclosed in square brackets matches any single character in the string. For example, [abcxyz] matches any single character from the set `abcxyz'. When the first character of the string is a caret (^), then the RE matches any character except those in the remainder of the string. For example, `[^45678]' matches any character except `45678'. A caret in any other position is interpreted as an ordinary character.

[]c...]

The right square bracket does not terminate the enclosed string if it is the first character (after an initial `^', if any), in the bracketed string. In this position it is treated as an ordinary character.

[l-r]

The minus sign, between two characters, indicates a range of consecutive ASCII characters to match. For example, the range `[0-9]' is equivalent to the string `[0123456789]'. Such a bracketed string of characters is known as a character class. The '-' is treated as an ordinary character if it occurs first (or first after an initial ^) or last in the string.

The following rules and special characters allow for constructing RE's from single-character RE's:

A concatenation of RE's matches a concatenation of text strings, each of which is a match for a successive RE in the search pattern.

- * A single-character RE, followed by an asterisk (*) matches zero or more occurrences of the single-character RE. Such a pattern is called a closure. For example, `[a-z][a-z]*` matches any string of one or more lower case letters.

\(...\)

An RE enclosed between the character sequences `\(` and `\)` matches whatever the unadorned RE matches, but saves

Kille

Expires: December 1994 Page 74

INTERNET--DRAFT

MHS Routing using Directory

June 1994

the string matched by the enclosed RE in a numbered substring register. There can be up to nine such substrings in an RE, and parenthesis operators can be nested.

- \n Match the contents of the nth substring register from the current RE. This provides a mechanism for extracting matched substrings. For example, the expression `^\(...)\1$` matches a string consisting entirely of two adjacent non-null appearances of the same string. When nested parenthesized substrings are present, n is determined by counting occurrences of `\(` starting from the left.

[E.1](#) Pseudo Code

This section describes the routing algorithm in pseudo-code and gives examples of its use.

To be supplied.

Kille

Expires: December 1994 Page 75

INTERNET--DRAFT

MHS Routing using Directory

June 1994

```
routedUA OBJECT-CLASS ::= {
  SUBCLASS OF {routingInformation}
  MAY CONTAIN {
    -- from X.402
    mhs-deliverable-content-length|
    mhs-deliverable-content-types|
    mhs-deliverable-eits|
    mhs-message-store|
    mhs-preferred-delivery-methods|
    -- defined here
    supportedExtensions|
    mandatoryRedirect|
    supportingMTA|
    filteredRedirect|
    userName|
    nonDeliveryInfo}
}
```

10

ID oc-routed-ua}

supportedExtensions ATTRIBUTE ::= {
 SUBTYPE OF objectIdentifier
 ID at-supported-extensions} 20

supportingMTA ATTRIBUTE ::= {
 SUBTYPE OF mtaInfo
 ID at-supporting-mta}

userName ATTRIBUTE ::= {
 SUBTYPE OF distinguishedName
 ID at-user-name} 30

Figure_5: __UA_Attributes__

mTABilateralTableEntry OBJECT-CLASS ::=
 SUBCLASS OF {mTA| distinguishedNameTableEntry}
 ID oc-mta-bilateral-table-entry}

Figure_7: __MTA_Bilateral_Table_Entry__

Kille

Expires: December 1994 Page 76

INTERNET--DRAFT

MHS Routing using Directory

June 1994

bilateralTable ATTRIBUTE ::= {
 WITH SYNTAX SEQUENCE OF DistinguishedName
 SINGLE VALUE
 ID at-bilateral-table}

Figure_8: __Bilateral_Table_Attribute__

supportedMTSExtensions ATTRIBUTE ::= {
 SUBTYPE OF objectIdentifier
 ID at-supported-mts-extensions}

Figure_9: __Supported_MTS_Extensions__

```

restrictedSubtree OBJECT-CLASS ::= {
    SUBCLASS OF {top}
    MAY CONTAIN {
        subtreeDeliverableContentLength|
        subtreeDeliverableContentTypes|
        subtreeDeliverableEITs}
    ID oc-restricted-subtree}

subtreeDeliverableContentLength ATTRIBUTE ::= {
    SUBTYPE OF mhs-deliverable-content-length
    ID at-subtree-deliverable-content-length}
10

subtreeDeliverableContentTypes ATTRIBUTE ::= {
    SUBTYPE OF mhs-deliverable-content-types
    ID at-subtree-deliverable-content-types}

subtreeDeliverableEITs ATTRIBUTE ::= {
    SUBTYPE OF mhs-deliverable-eits
    ID at-subtree-deliverable-eits}
20

```

Figure_10: Subtree_Capability_Restriction

Kille

Expires: December 1994 Page 77

INTERNET--DRAFT

MHS Routing using Directory

June 1994

```

initiatorP1Mode ATTRIBUTE ::= {
    WITH SYNTAX P1Mode
    SINGLE VALUE
    ID at-initiator-p1-mode}

responderP1Mode ATTRIBUTE ::= {
    WITH SYNTAX P1Mode
    SINGLE VALUE
    ID at-responder-p1-mode}
10

P1Mode ::= ENUMERATED {

```

```
push-only(0),
pull-only(1),
tw(2) }
```

```
polledMTAs ATTRIBUTE ::= {
  WITH SYNTAX PolledMTAs
  ID at-polled-mtas}
```

20

```
PolledMTAs ::= SEQUENCE {
  mta DistinguishedName,
  poll-frequency INTEGER OPTIONAL --frequency in minutes
}
```

```
mTAsAllowedToPoll ATTRIBUTE ::= {
  SUBTYPE OF distinguishedName
  ID at-mtas-allowed-to-poll}
```

Figure 11: Pulling Messages

Kille

Expires: December 1994 Page 78

INTERNET--DRAFT

MHS Routing using Directory

June 1994

```
responderAuthenticationRequirements ATTRIBUTE ::= {
  WITH SYNTAX AuthenticationRequirements
  SINGLE VALUE
  ID at-responder-authentication-requirements}
```

```
initiatorAuthenticationRequirements ATTRIBUTE ::= {
  WITH SYNTAX AuthenticationRequirements
  SINGLE VALUE
  ID at-initiator-authentication-requirements}
```

10

```
reponderPullingAuthenticationRequirements ATTRIBUTE ::= {
```



```
WITH SYNTAX AuthenticationRequirements
SINGLE VALUE
ID at-responder-pulling-authentication-requirements}
```

```
initiatorPullingAuthenticationRequirements ATTRIBUTE ::= {
  WITH SYNTAX AuthenticationRequirements
  SINGLE VALUE
  ID at-initiator-pulling-authentication-requirements}          20
```

```
AuthenticationRequirements ::= BITSTRING {
  mta-name-present(0),
  aet-present(1),
  aet-valid(2),
  network-address(3),
  simple-authentication(4),
  strong-authentication(5),
  bilateral-agreement-needed(6)}
```

Figure_12: Authentication_Requirements

Kille

Expires: December 1994 Page 79

INTERNET--DRAFT

MHS Routing using Directory

June 1994

```
respondingRTSCredentials ATTRIBUTE ::= {
  WITH SYNTAX RTSCredentials
  SINGLE VALUE
  ID at-responding-rts-credentials}
```

```
initiatingRTSCredentials ATTRIBUTE ::= {
  WITH SYNTAX RTSCredentials
  SINGLE VALUE
  ID at-initiating-rts-credentials}          10
```

```
RTSCredentials ::= SEQUENCE {
  request [0] MTAandPassword OPTIONAL,
  response [1] MTAandPassword OPTIONAL }
```

```
MTAandPassword ::= SEQUENCE {
    MTAName,
    Password }          -- MTAName and Password
                        -- from X.411
```

20

```
callingPresentationAddress ATTRIBUTE ::= {
    SUBTYPE OF presentationAddress
    MULTI VALUE
    ID at-calling-presentation-address}
```

```
callingSelectorValidity ATTRIBUTE ::= {
    WITH SYNTAX CallingSelectorValidity
    SINGLE VALUE
    ID at-calling-selector-validity}
```

30

```
CallingSelectorValidity ::= ENUMERATED {
    all-selectors-fixed(0),
    tsel-may-vary(1),
    all-selectors-may-vary(2) }
```

Figure_13: __MTA_Authentication_Parameters__

Kille

Expires: December 1994 Page 80

INTERNET--DRAFT

MHS Routing using Directory

June 1994

```
mTAWillRoute ATTRIBUTE ::= {
    WITH SYNTAX MTAWillRoute
    ID at-mta-will-route}
```

```
MTAWillRoute ::= SEQUENCE {
    from [0]          SET OF ORAddressPrefix OPTIONAL,
    to [1]            SET OF ORAddressPrefix OPTIONAL,
    from-excludes [2] SET OF ORAddressPrefix OPTIONAL,
    to-excludes [3]   SET OF ORAddressPrefix OPTIONAL } 10
```

```
ORAddressPrefix ::= DistinguishedName
```

Figure_14: __Simple_MTA_Policy_Specification__

```

mandatoryRedirect ATTRIBUTE ::= {
    SUBTYPE OF distinguishedName
    SINGLE VALUE
    ID at-mandatory-redirect}

filteredRedirect ATTRIBUTE ::= {
    WITH SYNTAX FilteredRedirect
    SINGLE VALUE
    ID at-filtered-redirect}
10

FilteredRedirect ::= SEQUENCE {
    redirect-to DistinguishedName,
    SEQUENCE {
        min-size [1] INTEGER,
        max-size [2] INTEGER,
        content [3] ContentType,
        eit [4] ExternalEncodedInformationType }
    }
20

```

Figure_15: Redirect_Definition

Kille

Expires: December 1994 Page 81

INTERNET--DRAFT

MHS Routing using Directory

June 1994

```

nonDeliveryInfo ATTRIBUTE ::= {
    WITH SYNTAX NonDeliveryReason
    ID at-non-delivery-info}

NonDeliveryReason ::= SEQUENCE {
    reason INTEGER (0..ub-reason-codes),
    diagnostic INTEGER (0..ub-diagnostic-codes),
    supplementaryInfo PrintableString }

```

Figure_16: Non_Delivery_Information

```

badAddressSearchPoint ATTRIBUTE ::= {
    SUBTYPE OF distinguishedName
    ID at-bad-address-search-point}

badAddressSearchAttributes ATTRIBUTE ::= {
    WITH SYNTAX AttributeType
    ID at-bad-address-search-attributes}

alternativeAddressInformation EXTENSION                                10
    AlternativeAddressInformation
    ::= id-alternative-address-information
        -- X.400(92) continues to use MACRO notation

AlternativeAddressInformation ::= SET OF SEQUENCE {
    distinguished-name DistinguishedName OPTIONAL,
    or-address ORAddress OPTIONAL,
    other-useful-info SET OF Attribute }

```

Figure_17: __Bad_Address_Pointers__

Kille Expires: December 1994 Page 82

INTERNET--DRAFT MHS Routing using Directory June 1994

```

localAccessUnit ATTRIBUTE ::= {
    WITH SYNTAX AccessUnitType
    ID at-local-access-unit}

AccessUnitType ::= ENUMERATED {
    fax (1),
    physical-delivery (2),
    teletex (3),
    telex (4) }
                                                                    10

accessUnitsUsed ATTRIBUTE ::= {
    WITH SYNTAX SelectedAccessUnit
    ID at-access-units-used}

SelectedAccessUnit ::= SEQUENCE {

```

```
type AccessUnitType,  
providing-MTA DistinguishedName,  
filter SET OF ORAddress OPTIONAL }
```

Figure_18: Access_Unit_Attributes

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

```
tc-cons OBJECT IDENTIFIER ::= {ts-communities 1}      -- OSI CONS  
tc-clns OBJECT IDENTIFIER ::= {ts-communities 2}      -- OSI CLNS  
tc-internet OBJECT IDENTIFIER ::= {ts-communities 3}  -- Internet + RFC 1006  
tc-int-x25 OBJECT IDENTIFIER ::= {ts-communities 4}  -- International X.25  
-- Without CONS10  
tc-ixi OBJECT IDENTIFIER ::= {ts-communities 5}      -- IXI (Europe)  
tc-janet OBJECT IDENTIFIER ::= {ts-communities 6}    -- Janet (UK)
```

Figure_20: Transport_Community_Object_Identifier_Assignments

Kille

Expires: December 1994 Page 83

INTERNET--DRAFT

MHS Routing using Directory

June 1994

```
mail-protocol OBJECT-IDENTIFIER ::= {iso(1) org(3) dod(6) internet(1)  
private(4)}
```

```

enterprises(1) isode-consortium (453) mail-protocol (5)}

ac-p1-1984 OBJECT IDENTIFIER ::= {mail-protocol 1}      -- p1(1984)
ac-smtp  OBJECT IDENTIFIER ::= {mail-protocol 2}      -- SMTP
ac-uucp  OBJECT IDENTIFIER ::= {mail-protocol 3}      -- UUCP Mail
ac-jnt-mail OBJECT IDENTIFIER ::= {mail-protocol 4}    -- JNT Mail (UK)
ac-p1-1988-x410 OBJECT IDENTIFIER ::= {mail-protocol 5} -- p1(1988) in X.410
mode
ac-p3-1984 OBJECT IDENTIFIER ::= {mail-protocol 6}      -- p3(1984) 10

```

_____Figure_21: __Protocol_Object_Identifier_Assignments_____