INTERNATIONAL STANDARD

ISO/IEC 10021-10

First edition
1998-09-01

Information technology — Message Handling Systems (MHS) —

Part 10: MHS routing

Technologies de l'information — Systèmes de messagerie (MHS) — Partie 10: Routage MHS

Reference number
ISO/IEC 10021-10:1998(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

International Standard ISO/IEC 10021-10 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 18, *Document processing and related communication*.

ISO/IEC 10021 consists of the following parts, under the general title *Information technology - Message Handling Systems (MHS):*

- *Part 1: System and service overview*
- *Part 2: Overall architecture*
- *Part 3: Abstract Service Definition Conventions*
- *Part 4: Message Transfer System: Abstract service definition and procedures*
- *Part 5: Message Store: Abstract service definition*
- *Part 6: Protocol specifications*
- *Part 7: Interpersonal messaging system*
- *Part 8: Electronic data interchange messaging service*
- *Part 9: Electronic data interchange messaging system*
- *Part 10: MHS routing*

Annexes A to D form an integral part of this part of ISO/IEC 10021. Annexes E to H are for information only.

Introduction

This part of ISO/IEC 10021 is one of a number of parts of ISO/IEC 10021 defining Message Handling in a distributed open systems environment.

Message Handling provides for the exchange of messages between users on a store-and-forward basis. A message submitted by one user (the originator) is transferred through the message-transfer-system (MTS) and delivered to one or more other users (the recipients).

This part of ISO/IEC 10021 defines a method for routing messages through the Message Handling System (MHS).

Information technology – Message Handling Systems (MHS) – Part 10: MHS routing

1 Scope

This part of ISO/IEC 10021 specifies the means by which messages are routed through the MHS, and supplements the procedures defined in 14.3 of ISO/IEC 10021-4.

Other parts of ISO/IEC 10021 define other aspects of the MHS. ITU-T Rec. F.400/X.400 | ISO/IEC 10021-1 defines the user-oriented services provided by the MHS. ITU-T Rec. X.402 | ISO/IEC 10021-2 provides an architectural overview of the MHS. ITU-T Rec. X.411 | ISO/IEC 10021-4 defines the abstract-service of the Message Transfer System.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 10021. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 10021 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

2.1 Presentation references

This part of ISO/IEC 10021 cites the following Presentation specifications:

— ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1: 1995, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

— ITU-T Recommendation X.681 (1994) | ISO/IEC 8824-2: 1995, *Information technology - Abstract Syntax Notation One (ASN.1): Information object specification.*

2.2 Directory references

This part of ISO/IEC 10021 cites the following Directory specifications:

— ITU-T Recommendation X.500 (1997) | ISO/IEC 9594-1: —[1)], *Information technology —Open Systems Interconnection —The Directory: Overview of Concepts, Models, and Services.*

— ITU-T Recommendation X.501 (1997) | ISO/IEC 9594-2: —[1)], *Information technology —Open Systems Interconnection —The Directory: Models.*

— ITU-T Recommendation X.520 (1997) | ISO/IEC 9594-6: —[1)], *Information technology —Open Systems Interconnection —The Directory: Selected attribute types.*

— ITU-T Recommendation X.521 (1997) | ISO/IEC 9594-7: —[1)], *Information technology —Open Systems Interconnection —The Directory: Selected object classes.*

2.3 Message Handling references

This part of ISO/IEC 10021 cites the following Message Handling System specifications:

— ITU-T Recommendation F.400/X.400 (1996), *Message handling services: Message handling system and service overview.*

ISO/IEC 10021-1: —[1)], *Information technology — Message Handling Systems (MHS) — Part 1: System and service overview.*

[1)] To be published

— ITU-T Recommendation X.402 (1995) | ISO/IEC 10021-2: 1996, *Information technology — Message Handling Systems (MHS): Overall architecture.*

— ITU-T Recommendation X.411 (1995) | ISO/IEC 10021-4: 1997, *Information technology — Message Handling Systems (MHS): Message transfer system: Abstract service definition and procedures.*

2.4 Country Code references

This part of ISO/IEC 10021 cites the following Country Code specifications:

— ISO 3166-1: 1997, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes.*

— CCITT Recommendation X.121 (1996), *International number plan for public data networks.*

2.5 Additional references

This part of ISO/IEC 10021 cites the following specification:

— ISO/IEC 9945-2: 1993, *Information technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities.*

3 Definitions

For the purposes of this part of ISO/IEC 10021, the following definitions apply.

3.1 MHS-routing definitions

The following terms are defined in clauses 6 and 7 of this part of ISO/IEC 10021:

— connection-group

— entry-connection-group

— enumerated connection-group

— indirect-exit-connection-group

— key-routing-collective

— local-exit-connection-group

— local-use-tables

— MHS-routing

— next-MTA

— OR-address-element

— OR-address-subtree

— routing-advice

— routing-collective

— routing-collective-subtree

— routing-MTA

— transit-exit-connection-group

— unenumerated connection-group

A glossary of these terms appears in Annex H.

3.2 MHS definitions

The following terms are defined in ITU-T Rec. X.402 | ISO/IEC 10021-2:

— Administration Management Domain (ADMD)

— Management Domain (MD)

— Message Handling System (MHS)

— Message Transfer Agent (MTA)

— Message Transfer System (MTS)

— Originator/recipient Address (OR-address)

— Private Management Domain (PRMD)

— Reliable Transfer Service Element (RTSE)

For the purposes of this part of ISO/IEC 10021, the term *message*, where used unqualified, refers generically to a *message*, *probe*, or *report*.

3.3 Directory definitions

The following terms are defined in ITU-T Rec. X.501 | ISO/IEC 9594-2:

— Directory Information Tree (DIT)

— Directory System Agent (DSA)

— Directory User Agent (DUA)

— Relative Distinguished Name (RDN)

4 Abbreviations

The abbreviations used in this part of ISO/IEC 10021 are defined in ITU-T Rec. X.402 | ISO/IEC 10021-2, and ITU-T Rec. X.501 | ISO/IEC 9594-2 (see 3.2 and 3.3), except for the following.

— ACSE Association Control Service Element (ITU-T Rec. X.217 | ISO/IEC 8649)

— APS Asynchronous Protocol Specification (ITU-T Rec.X.445)

— IP Internet Protocol

— LAN Local Area Network

— PSAP Presentation Service Access Point (ISO/IEC 7498-3)

— WAN Wide Area Network

— X.25 A packet-switched network conforming to ITU-T Rec. X.25

5 Conventions

This part of ISO/IEC 10021 uses the descriptive conventions listed below.

5.1 Conventions for routing model specification

This part of ISO/IEC 10021 uses the following ASN.1-based descriptive conventions for the indicated purposes:

a) To define the data types and values for MHS-routing, ASN.1 itself.

b) To define the Directory entries for MHS-routing, the OBJECT-CLASS, ATTRIBUTE, NAME-FORM, STRUCTURE-RULE, and MATCHING-RULE information object classes of ITU-T Rec. X.501 | ISO/IEC 9594-2.

Whenever this part of ISO/IEC 10021 describes a class of data structure having components, each component is categorized as one of the following **grades**:

a) **Mandatory** (M): A mandatory component shall be present in every instance of the class.

b) **Optional** (O): An optional component may be present in an instance of the class at the discretion of the object (e.g. user) supplying that instance.

c) **Conditional** (C): A conditional component shall be present in an instance of the class as specified by this part of ISO/IEC 10021.

5.2 General font conventions

Throughout this part of ISO/IEC 10021, terms are rendered in **bold** when defined. Terms that are proper nouns are capitalized, generic terms are not. Multi-word generic terms are hyphenated. Italic font is used for ASN.1 identifiers defined in other International Standards.

5.3 Font conventions for ASN.1 definitions

Throughout this part of ISO/IEC 10021, ASN.1 definitions appear in fixed-pitch font `(of this Type)` to highlight the difference between normal text and ASN.1 definitions. The font used for ASN.1 definitions is smaller than that used for normal text.

5.4 Rules for ASN.1 definitions

ASN.1 definitions appear both in the body of this part of ISO/IEC 10021 to aid the exposition, and again, formally, in annexes for reference. If a difference is found between the ASN.1 used in the exposition and that formally defined in the corresponding annex, a specification error is indicated.

6 MHS-routing Overview

The purpose of a Message Handling System (MHS) is to enable users to exchange messages on a store-and-forward basis. A message submitted on behalf of one user, the originator, is conveyed by the Message Transfer System (MTS) and subsequently delivered to the agents of one or more additional users, the recipients. The MTS comprises a collection of Message Transfer Agents (MTAs), which are highly distributed, and a message may traverse a number of MTAs on the journey from its originator to its recipient.

In MHS, the originator does not specify a path through the MTS to reach a recipient, but simply specifies a recipient OR-name (from which the OR-address is obtained). It is the responsibility of each MTA to determine the next MTA to which the message should be transferred to progress its journey to its recipient. Routing is thus the process of selecting, given an OR-address, the MTA to which the message should next be transferred.

The other parts of ISO/IEC 10021 specify the services provided by MHS, and the protocols used to transfer messages within the MTS. The means by which an MTA determines an appropriate route that will convey a message to its recipient is the subject of this part of ISO/IEC 10021. The various mechanisms defined herein which enable MTAs to make this determination constitute MHS-routing.

The path taken between an originator and recipient may vary on different occasions, since there will in general be a number of possible paths between them, and factors such as congestion and availability may influence route selection.

MTAs acquire routing information by accessing Directory entries whose maintenance is the responsibility of an MHS administrator. These entries model the various properties of the MHS that are relevant to routing. However, MHS-routing does not depend on the provision of a fully interconnected Directory.

6.1 Operational characteristics

MHS-routing has the following operational characteristics:

a) In MHS-routing, the OR-address name-space is decoupled from the constraints of hardware organization (e.g. the assignment of users to particular machines, or the ability of groups of machines to interconnect). This is in contrast with routing strategies that use OR-address pattern-matching methods to make routing decisions, which constrain an MHS administrator's choice in the allocation of OR-addresses to users, and require users to change their OR-addresses whenever their MTA changes.

   OR-addresses are intended to reflect the organizational hierarchy, and to use only as many levels of OR-address element as is required to achieve this. However, many organizations have staff distributed over multiple locations (where staff will necessarily use a local MTA), or have multiple messaging systems in use (e.g. mainframe-based, integrated office automation systems, and PC LAN email systems) where staff in any one department are arbitrarily allocated to different systems. Separating the OR-address hierarchy from the physical topology allows for the assignment to users of addresses which are compact and related to their organizational role, regardless of their physical location.

b) MHS-routing supports the range of connection densities possible among MTAs. One extreme is where all MTAs are connected to a common network and any MTA can connect to any other (e.g. a public wide-area network). At the other extreme an administrator specifies precisely which MTA pairs are able to communicate, as if the MTAs were connected by individual cables, regardless of the actual connection method.

   Some MHS installations restrict the topology (e.g. by performing all routing through a central switch MTA), because of the management cost in maintaining routing information, even when all the MTAs concerned are connected to a LAN or WAN and could, in principle, exchange messages directly. However, moving to the fully connected possibility may be undesirable for some organizations, due to concerns of security and cost optimization.

c) MHS-routing permits varying degrees of autonomy and segregation in the management of an MHS system. Many organizations will have the overall MHS managed by a central IT unit, with the management of individual system components devolved to local administrators. Often, the MHS management hierarchy will

be different from the organizational hierarchy. Typically, the MHS management reflects geographical locality, while the users are organized into geographically dispersed business units. In MHS-routing, this practice is accommodated by the use of two separate hierarchies for these two aspects of MHS organization.

d) MHS-routing accommodates a range of routing problem sizes, from an organization managing two or three MTAs, up to a PRMD operating hundreds or thousands of MTAs. An organization might organize only its internal connectivity, using ADMDs or specific bilateral agreements for all external connectivity, or it might join a more open community which publishes routing information in the Directory and allows for the open exchange of messages across a common network infrastructure.

e) MHS-routing does not constrain the choice of routing policies available to an organization. It provides a framework for the management of relevant routing information such that a range of routing strategies may be implemented.

f) MHS-routing provides a balanced trade-off in terms of cost/efficiency. Any scheme which relies on access to the Directory is likely to be slower to execute than one based purely on local tables held in the MTA. However, the Directory-based approach should provide the MTA with a better quality of information and so improve overall efficiency by the determination of optimal routes. For example, in a large PRMD it may often be infeasible to provide each MTA with customised tables listing every address in the domain, and consequently messages will take multiple hops towards their destination, passing through central MTAs that hold the tables. When Directory-based MHS-routing is used, the processing required at the originating MTA is greater, but is likely to identify a direct route and so enable transfer of the message in a single step.

   The use of MHS-routing also leads to a more scalable architecture, allowing for medium-capacity MTAs to be distributed throughout the network rather than depending on a small number of MTAs to perform central message switching. This scalable approach reduces the risks associated with a single point of failure on the network.

6.2 Components of the model

The different types of information stored in the Directory represent the modelling objects used to represent the MHS. Use of Directory access-control is not required for the operation of MHS-routing, but it may be employed where it is required to limit visibility to browsers of the Directory. Equally, information may be stored in a private DIT that has no connection to the outside world - there is no assumption that a fully interconnected Directory is available. However some routing policies, particularly those of an unrestricted nature, would benefit from a fully interconnected Directory.

6.2.1 Routing-collective

A **routing-collective** is a collection of one or more MTAs, under common management, which has collective responsibility for a portion of the OR-address name-space, and is capable of routing a message to any MTA managed within the collective. Therefore a routing-collective represents the management structure of some part of the MHS in the context of routing. By grouping together MTAs under common management control, the routing-collective provides abstraction (the internal structure of the routing-collective may be concealed from external MTAs) and is useful in limiting the scope of the information that MTAs need to hold.

The smallest instance of a routing-collective is a single MTA, and indeed each MTA is itself defined as a routing-collective. The largest instance of a routing-collective will typically be a Management Domain. While a routing-collective which comprises several PRMDs is not precluded, it is unusual for such PRMDs to be under genuinely common management control; the more typical case of loose confederations of PRMDs may be handled by other mechanisms.

Routing-collectives are structured in a hierarchical manner, with the number of levels chosen to suit the organization in question. Small MDs, or MDs of a very uniform nature under the complete control of a central IT management unit, will require only a two-level hierarchy. The MD is one routing-collective and contains all the MTAs (each of which is itself a routing-collective). More complex MDs will normally require a third level of routing-collective perhaps to group together all of the MTAs at a given location, or all those belonging to each department.

Each complete hierarchy of routing-collectives is represented by a DIT subtree, the **routing-collective-subtree**, which models their hierarchical relationships. All routing-collectives within a routing-collective subtree co-operate with one another in performing MHS-routing.

Each routing-collective has an entry in the Directory that indicates the connection-groups (see 6.2.3) by which a message may be transferred into the routing-collective, and those by which a message may be transferred out of the routing-collective.

6.2.2 Routing-MTA

A **routing-MTA** is an MTA that participates in MHS-routing as defined in this part of ISO/IEC 10021. By definition, a routing-MTA is the smallest instance of a routing-collective and occupies the lowest level in the routing-collective hierarchy, i.e. it appears as a leaf of the routing-collective subtree.

The routing-MTA is provided, by local configuration, with its routing-MTA Directory name (a routing-collective name) and credentials which enable it to read that Directory entry. All other information necessary for MHS-routing may be acquired from the Directory. As an implementation choice, this information may be retrieved at initialisation, or may be retrieved dynamically as required. For the purposes of exposition, the former approach is assumed. The caching of information acquired from the Directory may be necessary for efficient implementation, but is beyond the scope of this part of ISO/IEC 10021.

NOTE 1 – Implementors should be aware that out-of-date cached information can lead to routing loops and sub-optimal route choices. While procedures are specified to resolve routing loops, cache refreshing should also be performed at regular intervals.

A routing-MTA Directory entry indicates the one or more OR-address-subtrees which the routing-MTA's administrator has chosen to reflect the routing policy of this MTA, and also indicates the Directory name of the MHS Message Transfer Agent entry for this MTA (see A.1.3 of ITU-T Rec. X.402 | ISO/IEC 10021-2).

In order to fulfil its function, a routing-MTA must acquaint itself with knowledge of the other routing-collectives in its routing-collective-subtree. The minimum knowledge required for a routing-MTA to achieve this is discovered by collecting information from a specific subset of the routing-collectives in its routing-collective-subtree. These **key-routing-collectives** are defined as follows:

— the siblings of the routing-collective;

— the siblings of each of the routing-collective's superior routing-collectives.

NOTE 2 – While knowledge of a routing-collective's key-routing-collectives is required for the operation of MHS-routing, this information is not recorded explicitly in any Directory entry since it differs for every routing-collective, and may readily be discovered by inspection.

The reason for this definition of key-routing-collectives may be understood as follows. By the definition of routing-collective, a routing-MTA must be able to route a message to any of the other MTAs in each routing-collective of which it is a member; the practical effect is that the routing-MTA must be able to act on routing information that instructs it to transfer a message to one of the routing-collectives in the same routing-collective-subtree. Hence a simplistic definition of key-routing-collectives would include all of the routing-collectives in this tree. However, this gives more information than is strictly required. Plainly, the routing-MTA does not need to consider itself as a destination; similarly, an instruction to transfer the message to one of the routing-MTA's superior routing-collectives is meaningless (since the message has already arrived in that routing-collective), so these are not considered as key-routing-collectives. One of the purposes of having more than one level of routing-collective hierarchy is that a routing-MTA does not need to be aware of the internal structure of routing-collectives in distant parts of the routing-collective-subtree. Hence if one routing-collective is considered to be a key-routing-collective, its subordinates do not need to be. Excluding these unnecessary elements leaves the definition which has been adopted.

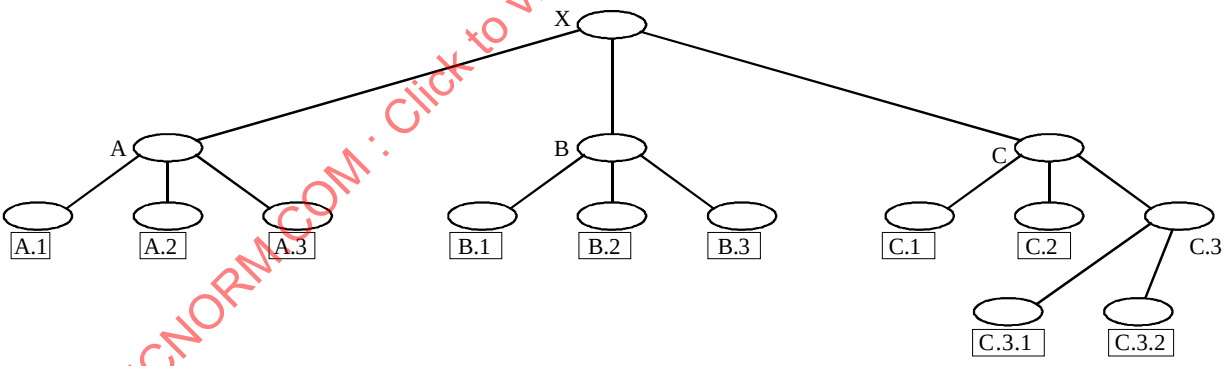

Figure 1 - Example of a routing-collective-subtree

Figure 1 illustrates the hierarchical relationships in a routing-collective-subtree. The routing-collective at the base of the subtree, X, has three immediate subordinate routing-collectives, A, B, and C. In turn, A, B, and C each contain three immediate subordinate routing-collectives. In the case of A and B, these routing-collectives are also routing-MTAs. In the case of C, C.1 and C.2 are routing-MTAs, while C.3 is a routing-collective containing two routing-MTAs. The names of the routing-MTAs are enclosed in boxes. For the routing-MTA B.3, the key-routing-collectives are B.1, B.2, A, and C.

6.2.3 Connection-group

A **connection-group** is a group of connections over which messages may be directly exchanged between members of a set of MTAs, using a specific MHS transfer protocol over a common network. It therefore represents the topology of the MHS, i.e., how the MTAs are physically interconnected. All MTAs in a connection-group have a network technology and a message transfer Application Context in common, and have administrative approval to interwork.

A connection-group is represented by specifying its member MTAs. A connection may be created between every pair of these MTAs. The Directory entry that represents a connection-group may also indicate the specific message transfer protocol used.

The simplest type of connection-group has just two members: this represents a conventional pair of MTAs that have been configured to communicate with one another. Larger sets may be constructed for convenience. If a number of MTAs are connected to a LAN or WAN, such that connections are permitted between any pair of them, they may all be assigned to a single connection-group, rather than to multiple groups that represent all the individual connections possible.

Connection-groups are of two types, enumerated and unenumerated. An **enumerated connection-group** is identified by Directory name, and has an associated list of member MTAs, as described above. An **unenumerated connection-group** is also identified by Directory name, and is typically associated with a network infrastructure such as a public or private wide-area network (e.g. X.25 or IP). Its membership is defined by self-declaration. The administrator who makes a local decision to configure an MTA into such a group is implicitly declaring that the MTA will accept connections from any other MTA on the same network, without prior agreement between MTA administrators.

Each routing-collective classifies the connection-groups available to it as follows:

a) an **entry-connection-group** is one that may be used to transfer messages into the routing-collective.

b) an **exit-connection-group** is one that may be used to transfer messages out of the routing-collective.

   Two types of exit-connection-groups are distinguished: a **transit-exit-connection-group** is one that may be used to transfer a message out of the routing-collective, regardless of origin; a **local-exit-connection-group** is one that may be used to transfer a message out of the routing-collective, but is available only for messages originated, or redirected, or DL-expanded within the routing-collective.

A given connection-group may be classified as both an entry- and exit-connection-group.

An **indirect-exit-connection-group** is an exit-connection-group which is not directly available to a routing-collective, but is available through one of its key-routing-collectives.

The connection-groups available to a routing-collective which is not a routing-MTA (i.e. does not occupy a leaf node in a routing-collective-subtree), correspond to connection-groups available to one or more of its subordinate routing-collectives. However, they are not necessarily the complete union of all such subordinate routing-collectives, since, as a matter of policy, a subordinate routing-collective may confine access to one of its connection-groups to itself and its subordinates.

Optionally, a security-context may be defined for a connection-group to govern the interactions permitted between members of that connection-group.

Figure 2 gives an alternative representation of the routing-collectives illustrated in Figure 1, and shows their connection-groups. The connection-groups CG1, 2, 4 and 5 are shown by indicating every individual connection within routing-collective X. Additional members of CG3 and CG6 exist within neighbouring routing-collectives.

6.2.4 OR-address-subtree

An **OR-address-subtree** is a Directory subtree which models a part of the OR-address name-space and associates routing-advice (see 6.2.5) with names within that name-space.

Directory attributes and object classes are defined corresponding to each element of the OR-address, such that a Directory name may be constructed by treating each OR-address element as an RDN, and assembling these RDNs in a prescribed order to form the Directory name. If entries were created in the Directory corresponding to all of the valid OR-addresses, this would create a subtree of the DIT corresponding to the whole OR-address name-space.

Since it is infeasible to place all routing information in a single Directory subtree (this would assume a fully interconnected Directory, a fully interconnected MHS, and that all users of Directory-based routing are prepared to reveal their complete addressing information to the world), it is usual to prefix the RDNs derived from OR-address elements with some arbitrary Directory name belonging to the administrator preparing the information. This allows multiple subtrees to be created in different parts of the global DIT, potentially containing information for every valid OR-address. A subtree may be incomplete, in that two or more subtrees are required to model a complete branch of the OR-address name-space.

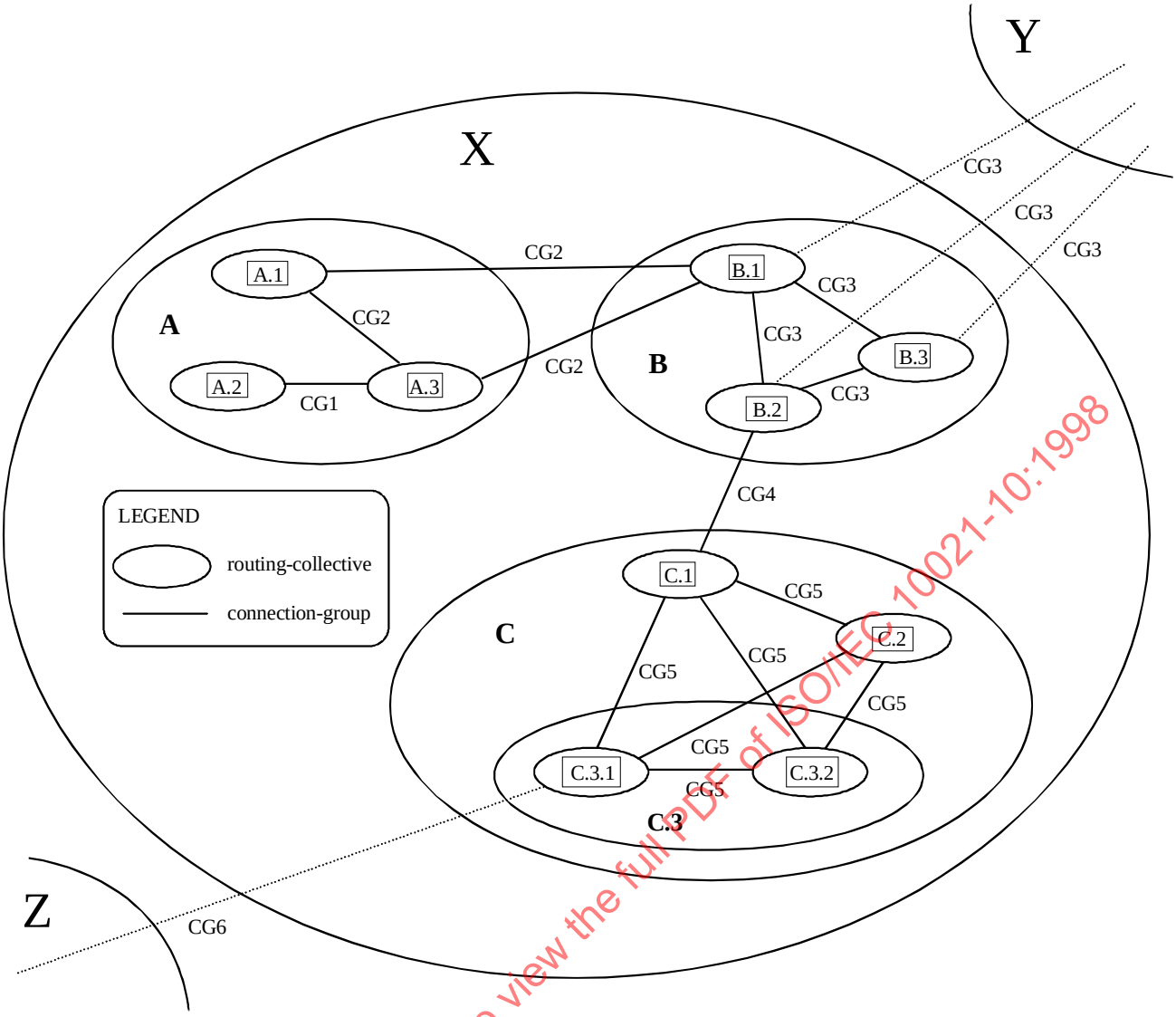

Figure 2 - Example of routing-collectives and their connection-groups

OR-address-subtrees containing different items of routing-advice will be required in various parts of the MHS for the same set of OR-addresses. This is due to the existence of firewalls, sparse connectivity, and the nature of devolved management. Different parts of the MHS will require different location-specific guidance to reach the same OR-address.

An administrator creates OR-address-subtrees to indicate the required routing behaviour for each possible OR-address. The entry corresponding to an OR-address will identify a routing-collective towards which the message should be transferred. In the simplest case, the routing-collective is the delivering-MTA for that recipient. Alternatively, the routing-collective might be larger, and contain the delivering-MTA as a subordinate routing-collective. In this case the message may be transferred to an arbitrary MTA in the routing-collective where more detailed information will be available to identify the delivering-MTA to which it should finally be transferred. Another possibility is that the indicated routing-collective is a relaying MTA that the administrator knows is able to handle all addresses in a particular domain.

Certain OR-addresses have an arbitrary internal structure, and cannot be fully represented in OR-address-subtrees (for example, one containing a domain defined attribute specifying an address in a foreign messaging environment). To represent a branch of the subtree that models OR-addresses of this type, an *expression-matches* attribute may be placed in the subtree to allow for algorithmic matching of these OR-addresses.

As it is impracticable to create entries in each OR-address-subtree for every possible OR-address, an OR-address-subtree may be incomplete in two ways: parts of the OR-address name-space may simply be absent from the tree (e.g. a tree which contains only local users), or whole branches of the tree may be truncated. Portions of the tree that are incomplete are marked as such, so that an MTA reading information from the tree is able to distinguish between an entry which is absent because the OR-address is invalid (causing the MTA to declare the message undeliverable), and an entry which is absent because the administrator has not provided the information (causing the MTA to look elsewhere for more information).

Portions of an OR-address-subtree may be truncated where the behaviour requested of the MTA is the same at all subordinate vertices of a vertex. For example, if all addresses within a particular Organizational Unit require transfer to the same MTA, it is not necessary to create entries for each OR-address in that Organizational Unit. Instead, the routing

information is placed at the vertex in the tree corresponding to that Organizational Unit. The procedures used by the MTA for reading information from OR-address-subtrees detect when the desired entry cannot be read because the tree has been truncated, and cause the MTA to consult the leaf node of the truncated tree for routing advice.

It is entirely practicable to have just one OR-address-subtree for a PRMD (or indeed for a number of co-operating PRMDs), even if the PRMD has a nesting of routing-collectives within it. However, there will often be a requirement for multiple OR-address-subtrees, for example, where administrators wish to conceal information about OR-addresses, or for devolved management of OR-address-subtrees, or for private subtrees that identify locally available relaying and ADMD services.

6.2.5 Routing-advice

The **routing-advice** present in an entry in an OR-address-subtree provides advice to the routing-MTA that assists it in arriving at a routing decision for the OR-address corresponding to the entry. It specifies one of the following actions:

a) Transfer the message to the target routing-collective indicated. If this identifies the routing-MTA itself, then local delivery is attempted.

b) Generate a non-delivery report with the reason and diagnostic codes indicated.

c) Redirect the message to a preferred address of the MTS-user. The new OR-address is indicated in the routing-advice.

d) Perform DL-expansion of the OR-address; if this is not possible, transfer the message to the target routing-collective indicated, which is known to be capable of performing the required DL-expansion.

e) Place the message in an inner-envelope content-type, using the information supplied to construct the outer-envelope (see 8.2.1.1.1.34 of ITU-T Rec. X.411 | ISO/IEC 10021-4).

6.2.6 Local use tables

The only configuration information required by a routing-MTA is its routing-collective Directory name, information needed to access the Directory (e.g., its Directory credentials and the address of a local DSA), and knowledge of how to map its exit-connection-groups to appropriate protocol stacks and physical interfaces. On initialization, the routing-MTA reads the entry identified by its routing-collective Directory name, and follows a procedure that provides it with knowledge of its complete routing-collective-subtree, and thereby enables it to participate in MHS-routing (see 9.1.5). This information is recorded by the routing-MTA in its **local-use-tables**, as follows:

a) The Directory name of this MHS Message Transfer Agent.

b) The names of one or more OR-address-subtrees configured for this routing-MTA.

c) A list of the key-routing-collectives of this routing-MTA, each accompanied by the names of one or more next-MTAs for that key-routing-collective.

d) The names of the routing-MTA's transit- and local-exit-connection-groups.

e) The names of the routing-MTA's indirect-exit-connection-groups (i.e. those exit-connection-groups available through its key-routing-collectives), each accompanied by the names of one or more next-MTAs for that connection-group.

f) For each of the routing-MTA's entry-connection-groups that is of type enumerated, authentication information required to identify each possible calling MTA.

6.3 Routing decision overview

When routing a message, the routing-MTA performs the following actions to arrive at a routing decision for each recipient OR-address in the message:

a) The routing-MTA transforms the OR-address into a Directory name, and attempts to read the corresponding entry from the first configured OR-address-subtree. If the read is successful, the routing-advice associated with the entry is obeyed. If the read fails with a Name Error, and the lowest entry that was matched contains routing-advice then that advice is followed. If the lowest matching entry does not contain routing-advice and the next level of the subtree is marked as incomplete, then there may be no information available in this subtree, and the routing-MTA repeats the procedure using the next OR-address-subtree (typically, a routing-MTA is configured with several). However, if the next level of the subtree is marked as complete, then the OR-address is invalid and a non-delivery report is generated.

b) If routing-advice is obtained, this may identify a target routing-collective; alternatively, it may indicate that the OR-address is invalid, or that redirection of the message is required, or that DL-expansion is required.

c) If a target routing-collective is identified, the routing-MTA checks its local-use-tables (generated when the routing-MTA was initialized) to determine whether this is a key-routing-collective. If so, the next-MTA for that routing-collective is known, and the message is transferred to it.

d) The target routing-collective's Directory entry is read and a list of its entry-connection-groups is retrieved. If any of this routing-MTA's exit-connection-groups matches one of these entry-connection-groups then the message is transferred by direct connection. Alternatively, if the routing-MTA's local-use-tables identify the target connection-group as belonging to an indirect-exit-connection-group (i.e. an exit-connection-group offered by a key-routing-collective) then the next-MTA for that connection-group is known, and the message is transferred to it.

e) If the actions described above have not identified an MTA to which the message is to be transferred, this may be because the OR-address-subtree contains unnecessary detail, and has identified an inner routing-collective of which this MTA has limited knowledge. The routing-MTA discards the last RDN of the target routing-collective name, thus obtaining the name of the target routing-collective's parent entry.

f) If the base of the routing-collective-subtree has not yet been reached, the procedure resumes at step *c* using the truncated name as the new target routing-collective.

NOTE – Truncation of the target routing-collective name is particularly effective in the case where the first target routing-collective corresponds to the delivering MTA (for which this routing-MTA knows no route), and a superior routing-collective is more publicly visible.

g) Otherwise, the procedure is repeated from step *a* using the next configured OR-address-subtree.

6.4 Directory organization

The various information required to support MHS-routing is stored in the Directory according to:

— the access requirements of the routing-MTAs that are users of the information;

— the management requirements of the various administrators of the information.

For performance reasons, it may be convenient to store each routing-collective-subtree and OR-address-subtree in a single DSA, using access controls where devolved management is required. However, all the normal capabilities of the Directory for distributed operation are available. Each routing-collective-subtree is stored in a part of the global DIT (or private DIT) chosen by the administrator of the subtree.

NOTE 1 – In general, access controls should not be applied selectively to the attributes present in routing-collective or OR-address entries since these can cause erroneous routing decisions. Where required, access control should be applied to complete entries.

Access to the constituent routing-collective entries of a routing-collective-subtree will be granted to the one or more administrators who manage these constituent routing-collectives. Every routing-MTA in a routing-collective-subtree requires read access to all routing-collective entries in that subtree. Other routing-MTAs that are permitted to communicate with routing-MTAs in the routing-collective will also require read access to these entries. Routing-collectives change relatively infrequently.

NOTE 2 – In the absence of a globally interconnected Directory, it may be necessary to hold duplicate entries for distant routing-collectives within a private (unconnected) DIT.

The various OR-address-subtrees used by a routing-MTA may be maintained under different management and stored in different locations. Only those routing-MTAs that have been configured to use a particular OR-address-subtree require access to it.

Connection-groups are not interrelated. Hence the Directory entry representing a connection-group may be stored in any location at the discretion of the connection-group's administrator. Every routing-MTA in a routing-collective-subtree requires access to the entries for each entry- and exit-connection-group for which it is configured.

An MHS Message Transfer Agent entry, which stores protocol information for an MTA, may be stored in any convenient location. Access to such an entry is required by every other routing-MTA which shares a connection-group with the MTA.

6.5 Authentication principles

Associations between MTAs are established by means of the MTA-bind abstract-operation where three authentication methods are possible: no authentication, simple password, and strong authentication. Each routing-MTA may support one or more of these methods. The actual method used in any instance of connection is governed by the associated connection-group.

For the purposes of simple password each routing-MTA has just one password value, which is used in the initiator-credentials of all MTA-bind operations to other routing-MTAs, and is also used in the responder-credentials when responding to an MTA-bind from another routing-MTA. This password is stored in the MTA's MHS-Message Transfer Agent entry in the Directory. The MTA itself can read the value of the password to use in MTA-bind operations; access controls may be applied to limit the access of other MTAs to the use of the Compare operation.

NOTE – The denial of read access prevents other MTAs from reading the password into local tables, and so implies a Directory transaction for each MTA-bind attempt.

Exceptionally, to allow interworking with existing MTAs which are not routing-MTAs, a separate password may be specified for each connection between a routing-MTA and an MTA without MHS-routing capability. Typically, a non-routing MTA will not have a Directory entry associated with it, and so the administrator responsible for the routing-MTA will create a 'proxy' Directory entry corresponding to the non-routing MTA; any necessary additional passwords are stored in that entry.

A routing-MTA may supply its Directory name (i.e. the name of its MHS Message Transfer Agent entry) in the A-ASSOCIATE service of ACSE used when invoking MTA-bind. Under most circumstances, this is optional and serves only to improve the efficiency of the authentication process at the called MTA. However, the administrator may specify that for a particular connection-group the use of Directory name in A-ASSOCIATE is mandatory, either to inform MTAs that they can rely on the presence of the Directory name (and so implement optimisations such as reducing the size of local-use-tables), or in the case of unenumerated connection-groups to give a slightly improved level of authentication.

7 Routing-collective-subtree

The **routing-collective-subtree** models the high level management framework of an MHS. It provides a mechanism for storing information that may be accessed by an MTA to assist in making a routing decision.

7.1 Object classes

The following Directory object classes relevant to the routing-collective-subtree are defined in 7.1.1 - 7.1.4:

— the *Routing Collective* object class;

— the *Routing MTA* object class;

— the *Connection Group* object class;

— the *MTA Information* object class.

7.1.1 Routing Collective object class

The **Routing Collective** object class is a structural object class used to represent a collection of one or more MTAs, under common management, that has collective responsibility for a portion of the OR-address name-space. The attributes in its entry identify its routing-collective-name, and, to the extent that the relevant attributes are present, its description, its entry-connection-group-names, its transit-exit-connection-group-names, and its local-exit-connection-group-names.

```
routingCollective      OBJECT-CLASS ::= {
    SUBCLASS OF        {top}
    MUST CONTAIN       {routingCollectiveName}
    MAY CONTAIN        {description | entryConnectionGroupName | localExitConnectionGroupName |
                       transitExitConnectionGroupName}
                       --at least one entry-CG and exit-CG should be present--
    ID                 id-oc-routing-collective }
```

At least one transit-exit-connection-group-name or local-exit-connection-group-name, and one entry-connection-group-name attribute should be present in entries of the Routing Collective object-class.

NOTE – Normally, a routing-collective will have access to one or more entry- and exit-connection-groups. Exceptionally, a top-level routing-collective in a closed network may have access to no entry- or exit-connection-groups.

7.1.2 Routing MTA object class

The **Routing MTA** object class, a subclass of Routing Collective, is a structural object class used to represent the smallest instance of a routing-collective, i.e. an MTA that participates in MHS-routing. The attributes in its entry, in addition to those defined for a Routing Collective, identify the OR-address-subtrees it is to use, and name its MHS Message Transfer Agent entry.

```
routingMTA             OBJECT-CLASS ::= {
    SUBCLASS OF        {routingCollective}
    MUST CONTAIN       {oRAddressSubtrees | mHSMessageTransferAgentName}
    ID                 id-oc-routing-mta }
```

7.1.3 Connection Group object class

The **Connection Group** object class is a structural object class used to represent a group of MTAs that possess the mutual capability of directly exchanging messages with one another. It therefore represents the physical topology of the MHS, i.e., how the MTAs are physically interconnected. The attributes in its entry identify its common name, indicate whether it is of type enumerated or unenumerated, and, to the extent that the relevant attributes are present, describe the connection-

group, indicate its connection-type, indicate a password for the connection-group, and enumerate its member-MTAs (if of type enumerated), and identify its security-context.

```
connectionGroup         OBJECT-CLASS ::= {
    SUBCLASS OF         {top}
    MUST CONTAIN        {commonName | enumeratedFlag}
    MAY CONTAIN         {description | connectionType | groupMTAPassword | memberMTA |
                        securityContext}
    ID                  id-oc-connection-group }
```

7.1.4 MTA Information object class

The **MTA Information** object class is an auxiliary object class used to represent the additional information required to perform MHS-routing. It is intended to be used in objects of class MHS Message Transfer Agent (see A.1.3 of ITU-T Rec. X.402 | ISO/IEC 10021-2). The attributes in its entry identify its MTA name, its global domain identifier, and, to the extent that the relevant attributes are present, indicate the MTA password, other specific passwords, and the calling presentation addresses.

```
mTAInformation          OBJECT-CLASS ::= {
    KIND                auxiliary
    MUST CONTAIN        {mTAName | globalDomainIdentifier}
    MAY CONTAIN         {mTAPassword | specificPasswords | callingPSAPs}
    ID                  id-oc-mta-information }
```

7.2 Attribute types

The attribute types specific to entries of the Routing Collective, Connection Group, Routing MTA, and MTA Information object classes are defined in 7.2.1 - 7.2.4.

7.2.1 Routing Collective attribute types

The attribute types defined below are specific to entries of the Routing Collective object class.

7.2.1.1 Routing Collective Name attribute type

The **Routing Collective Name** attribute type specifies an identifier for Routing Collective entries.

```
routingCollectiveName   ATTRIBUTE ::= {
    SUBTYPE OF          commonName -- see ITU-T X.520 | ISO/IEC 9594-6 --
    SINGLE VALUE        TRUE
    ID                  id-at-routing-collective-name }
```

NOTE - This attribute is used, rather than Common Name, because it makes possible an optimization in the Routing-decision procedure.

7.2.1.2 Connection Group Name attribute type

The **Connection Group Name** attribute type identifies entries of the Connection Group object class which represent connection-groups that may be used to transfer a message into, or out of a routing-collective.

```
connectionGroupName     ATTRIBUTE ::= {
    WITH SYNTAX         DistinguishedName
    SINGLE VALUE        FALSE
    ID                  id-at-connection-group-name }
```

The following attribute types classify Connection Groups according to the distinctions made by each routing-collective:

a) The **Entry Connection Group Name** attribute type identifies the connection-groups which may be used to transfer a message into a routing-collective.

```
entryConnectionGroupName ATTRIBUTE ::= {
    SUBTYPE OF          connectionGroupName
    ID                  id-at-entry-connection-group-name }
```

b) The **Transit Exit Connection Group** attribute type identifies the connection-groups which may be used to transfer a message out of a routing-collective.

```
transitExitConnectionGroupName ATTRIBUTE ::= {
    SUBTYPE OF          connectionGroupName
    ID                  id-at-transit-exit-connection-group-name }
```

c) The **Local Exit Connection Group Name** type identifies the connection-groups which may be used to transfer a message out of a routing-collective, if, and only if, that message originated, or was redirected, or was DL-expanded within the routing-collective.

```
localExitConnectionGroupName ATTRIBUTE ::= {
    SUBTYPE OF          connectionGroupName
    ID                  id-at-local-exit-connection-group-name }
```

7.2.2 Routing MTA attribute types

The attribute types defined below are specific to entries of the Routing MTA object class.

7.2.2.1 OR-address Subtrees attribute type

The **OR-address Subtrees** attribute type identifies the OR-address-subtrees that have been configured for a routing-MTA.

```
oRAddressSubtrees       ATTRIBUTE ::= {
    WITH SYNTAX         ORAddressSubtreeNames
    SINGLE VALUE        TRUE
    ID                  id-at-oraddress-subtrees }

ORAddressSubtreeNames ::= SEQUENCE OF DistinguishedName
```

7.2.2.2 MHS Message Transfer Agent Name attribute type

The **MHS Message Transfer Agent Name** attribute type identifies the MHS Message Transfer Agent entry for a routing-MTA.

```
mHSMessageTransferAgentName ATTRIBUTE ::= {
    SUBTYPE OF          distinguishedName
    SINGLE VALUE        TRUE
    ID                  id-at-mhs-message-transfer-agent }
```

7.2.3 Connection Group attribute types

The attribute types defined below are specific to entries of the Connection Group object class.

7.2.3.1 Enumerated Flag attribute type

The **Enumerated Flag** attribute type indicates whether a connection-group is of type enumerated or unenumerated.

```
enumeratedFlag          ATTRIBUTE ::= {
    WITH SYNTAX         BOOLEAN -- True=enumerated, False=unenumerated --
    SINGLE VALUE        TRUE
    ID                  id-at-enumerated-flag }
```

7.2.3.2 Connection Type attribute type

The **Connection Type** attribute type indicates details of connection information specific to a connection-group: the application-context, profiles, use of A-ASSOCIATE, use of network address, and the authentication method.

```
connectionType          ATTRIBUTE ::= {
    WITH SYNTAX         ConnectionInformation
    SINGLE VALUE        TRUE
    ID                  id-at-connection-type }

ConnectionInformation ::= SET {
    application-context      [0] OBJECT IDENTIFIER DEFAULT id-ac-mts-transfer,
    profiles                 [1] SET OF OBJECT IDENTIFIER OPTIONAL,
    dn-used-in-a-associate   [2] BOOLEAN DEFAULT TRUE,
    network-address-reliable [3] BOOLEAN DEFAULT TRUE,
    authentication-method    [4] AuthenticationMethod DEFAULT simple-password}

AuthenticationMethod ::= INTEGER {
    no-authentication        (0),
    simple-password          (1),
    strong-authentication    (2) }
```

The components of **connection-information** are defined as follows:

a) **Application-context** (D *id-ac-mts-transfer*): The MHS application-context used in this connection-group. This value should be present in the Supported Application Context attribute of each entry of the MHS Message Transfer Agent object class which represents the member MTAs of the connection-group. By default, this component identifies the *mts-transfer* application-context.

b) **Profiles** (O): The one or more profiles that characterize the Connection Type (see Annex G). At least one of the values of profiles should be present in the Protocol Information attribute of each entry of the MHS Message Transfer Agent object class which represents the member MTAs of the connection-group. The profiles component is used to select between the various network addresses present in this entry. The absence of the profiles component indicates that the profiles are unspecified, and hence that any of the MTA's addresses may be used.

c) **DN-used-in-A-ASSOCIATE** (D *true*): This indicates whether all member MTAs of the connection-group supply their Directory names (of the MHS Message Transfer Agent entry) in the calling application-entity-title of the A-ASSOCIATE service (of ACSE) used to invoke MTA-bind. If *true*, then each of the member MTAs of the connection-group shall supply its Directory name when using that connection-group. If *true*, then application-context shall have the value *mts-transfer* (since *mts-transfer-protocol* and *mts-transfer-protocol-1984* do not use ACSE).

If DN is not present in A-ASSOCIATE for an unenumerated connection-group where the authentication-method is *simple-password*, then the Group MTA Password attribute must be present in the entry. If DN is not present in A-ASSOCIATE for an unenumerated connection-group where the authentication-method is *strong-authentication*, then certificate must be present in the initiator-credentials argument of MTA-bind.

NOTES

1 If set *true* for an enumerated connection-group, the size of the local-use-tables may be considerably reduced since only the DNs of other MTAs need be stored, not their complete connection information.

2 For unenumerated connection-groups, masquerade is simple and undetectable unless the Directory name is present in A-ASSOCIATE, and this name used to validate the existing Trace and the calling PSAP. While this check makes masquerade harder, it does not make it impossible, since the Directory name may identify a bogus entry in the Directory.

d) **Network-address-reliable** (D *true*): This specifies whether the calling network address is required to match for MTA-bind authentication. The component is *false* for those network types where the calling network address is not available or is unpredictable (e.g., the PSTN). If the component is *true*, and the Enumerated Flag of the connection-group is *false*, then the value of DN-used-in-A-ASSOCIATE shall be *true*.

NOTE 3 – In an unenumerated connection-group, unless DN-used-in-A-ASSOCIATE is *true*, then a routing-MTA will have no means to discover the required calling network address of a calling MTA and will be unable to test the address actually used.

e) **Authentication-method** (D *simple-password*): This indicates the authentication-method used by all MTAs in the connection-group. Its possible values are: *no-authentication*, *simple-password*, and *strong-authentication*.

NOTE 4 – In an unenumerated connection-group, the authentication-method *strong-authentication* requires either that DN-used-in-A-ASSOCIATE is *true*, or that certificate is present in MTA-bind.

7.2.3.3 Group MTA Password attribute type

The **Group MTA Password** attribute type contains a password used by members of an unenumerated connection-group. This attribute shall be absent from the connection-group object class if the Enumerated Flag attribute has the value *true*. This attribute shall be present if the Enumerated Flag attribute has the value *false*, and the value of DN-used-in-A-ASSOCIATE is *false*. When using a connection-group for which a Group MTA Password attribute is defined, an MTA's own MTA Password attribute is disregarded.

NOTE – The presence of this attribute indicates the administrator's view that not all of the MTAs in the connection-group are capable of presenting their Directory names in A-ASSOCIATE, or capable of obtaining access to the Directory entries pointed to by their peers.

```
groupMTAPassword        ATTRIBUTE ::= {
    WITH SYNTAX         Password
    SINGLE VALUE        TRUE
    ID                  id-at-group-mta-password }
```

7.2.3.4 Member MTA attribute type

The **Member MTA** attribute type identifies the MTAs that are members of an enumerated connection-group. This attribute shall be present in entries of the Connection Group object class if the Enumerated Flag attribute has the value *true*, and shall be absent otherwise.

```
memberMTA               ATTRIBUTE ::= {
    WITH SYNTAX         RoutingMTAName
    SINGLE VALUE        FALSE
    ID                  id-at-member-mta }

RoutingMTAName ::= RoutingCollectiveName

RoutingCollectiveName ::= DistinguishedName
```

7.2.3.5 Security Context attribute type

The **Security Context** attribute type identifies the security-context within which members of a connection-group interact (see 12.1.1.1.1.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4).

```
securityContext         ATTRIBUTE ::= {
    WITH SYNTAX         SecurityContext
    SINGLE VALUE        TRUE
    ID                  id-at-security-context }
```

7.2.4 MTA Information attribute types

The attribute types defined below are specific to entries of the MTA Information object class.

7.2.4.1 MTA Name attribute type

The **MTA Name** attribute type identifies the MTA-name used by an MTA when generating trace information, and which may be used in the arguments and results of MTA-bind.

```
mTAName                 ATTRIBUTE ::= {
    WITH SYNTAX         MTAName
    SINGLE VALUE        TRUE
    ID                  id-at-mta-name }
```

7.2.4.2 Global Domain Identifier attribute type

The **Global Domain Identifier** attribute type identifies the global-domain-identifier used by an MTA when generating trace information, and, where strong-authentication is in use, in the arguments and results of MTA-bind.

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

7.2.4.3 MTA Password attribute type

The **MTA Password** attribute type identifies the password optionally presented by an MTA when invoking MTA-bind and returned in response to MTA-bind. This value is used on every occasion when communicating with another routing-MTA if the associated connection-group's authentication-method is *simple-password*, and the connection-group's Group MTA Password attribute is absent.

NOTE – Typically, an unenumerated connection-group will possess a Group MTA Password attribute which takes precedence over the individual MTA Passwords of its members.

```
mTAPassword             ATTRIBUTE ::= {
    WITH SYNTAX         Password
    SINGLE VALUE        TRUE
    ID                  id-at-mta-password }
```

7.2.4.4 Specific Passwords attribute type

The **Specific Passwords** attribute type identifies passwords required when communicating with MTAs that do not support MHS-routing. It may be present in proxy entries created to represent these non-routing MTAs.

```
specificPasswords ATTRIBUTE ::= {
    WITH SYNTAX         SpecificPassword
    SINGLE VALUE        FALSE
    ID                  id-at-specific-passwords }

SpecificPassword ::= SET {
    routing-collective-name RoutingCollectiveName,
    this-mta-password       [0] Password,
    calling-mta-password    [1] Password }
```

The component of specific-password are defined as follows:

a) **Routing-collective-name** (M): The Directory name of the routing-collective for whose use this attribute value is provided.

b) **This-MTA-password** (M): The password used by this MTA when it initiates an association with a member of the routing-collective indicated. This value takes precedence over the value of the MTA Password attribute associated with this entry.

c) **Calling-MTA-password** (M): The password to be used by a calling MTA in the routing-collective indicated when it initiates an association with this MTA. The value overrides the value of the MTA Password attribute present in the calling routing-MTA's entry.

When a routing-MTA communicates with an MTA for which a Specific Passwords attribute is defined, which identifies the routing-MTA's routing-collective, then the calling-MTA-password and this-MTA-password take precedence over the MTA Password of the routing-MTA and MTA Password of this non-routing MTA. These values also take precedence over any Group MTA Password defined for the connection-group.

7.2.4.5 Calling PSAPs attribute type

The **Calling PSAPs** attribute type identifies the presentation address values used by this MTA when it invokes MTA-bind (where these differ from those given in the Presentation Address attribute of the MHS Message Transfer Agent entry). It may be used by a called MTA to authenticate the identity of the calling MTA.

```
callingPSAPs ATTRIBUTE ::= {
    WITH SYNTAX         PresentationAddress
    SINGLE VALUE        FALSE
    ID                  id-at-calling-psaps }
```

7.3 Name forms

The **Routing Collective name form**, which specifies how entries of object class Routing Collective may be named, and the **Connection Group name form**, which specifies how entries of object class Connection Group may be named, are defined as follows:

```
routingCollectiveNameForm NAME-FORM ::= {
    NAMES               routingCollective
    WITH ATTRIBUTES     {routingCollectiveName}
    ID                  id-nf-routing-collective }

connectionGroupNameForm NAME-FORM ::= {
    NAMES               connectionGroup
    WITH ATTRIBUTES     {commonName}
    ID                  id-nf-connection-group}
```

8 OR-address-subtree

The **OR-address subtree** is a DIT subtree which models part of the OR-address name-space and contains routing-advice for OR-addresses present within that name-space.

8.1 OR-address Element object class

The **OR-address Element** object class is an abstract object class, and is the direct superclass of every entry in an OR-address-subtree. It associates routing-advice with OR-addresses. The attributes in its entry, to the extent that the relevant attributes are present, indicate an item of routing-advice, an expression-match, a recipient-MD-assigned-alternate-recipient, and indicate whether all potential immediate subordinate entries of this entry are actually present.

```
oRAddressElement        OBJECT-CLASS ::= {
    SUBCLASS OF         {top}
    KIND                abstract
    MAY CONTAIN         {routingAdvice | expressionMatches | nextLevelComplete |
                        recipientMDAssignedAlternateRecipient}
    ID                  id-oc-mhs-or-address-element }
```

Each OR-address Element represents a primitive component of the OR-address.

NOTE – For example, the standard attribute *personal-name* comprises the four OR-address-elements MHS Surname, MHS Given Name, MHS Initials, and MHS Generation Qualifier.

Some OR-address attributes (e.g. the domain-defined-attributes) have an arbitrary internal structure, and cannot be treated directly as naming elements, or be represented productively by corresponding OR-address Elements. The Expression Matches attribute type is provided to enable algorithmic matching of these OR-address attributes.

8.2 OR-address Element attribute types

The attribute types defined below are specific to entries of the OR-address Element object class.

8.2.1 Routing Advice

The **Routing Advice** attribute type specifies an item of routing-advice associated with the OR-address that corresponds to an entry. This indicates one of the following:

- a route for the message;
- an indication that it cannot be delivered to the recipient;
- the OR-address to which the message is to be redirected;
- information that will enable the MTA to DL-expand the message or transfer it to its DL-expansion point;
- an indication that the message is to be placed in an inner-envelope content-type.

```
routingAdvice          ATTRIBUTE ::= {
    WITH SYNTAX        RoutingAdvice
    SINGLE VALUE       TRUE
    ID                 id-at-mhs-routing-advice }

RoutingAdvice ::= CHOICE {
    target-routing-collective   [0] TargetRoutingCollective,
    non-delivery-information    [1] NonDeliveryInformation,
    alias-redirection           [2] AliasRedirection,
    dl-expansion-information    [3] DLExpansionInformation,
    double-envelope-information [4] DoubleEnvelopeInformation,
    ...  }
```

An item of routing-advice contains one of the following:

a) **Target-routing-collective** (C): The name of a **target routing-collective** to which, it is advised, the message should be transferred. The **local-user-identifier** is intended for use by the delivering MTA (i.e. is relevant only if the routing-MTA is also the delivering MTA). The values of local-user-identifier are implementation-specific.

```
TargetRoutingCollective ::= SEQUENCE {
    target-routing-collective [0] RoutingCollectiveName,
    local-user-identifier [1] UniversalOrBMPString {ub-local-user-identifier}
                                    OPTIONAL }

ub-local-user-identifier INTEGER ::= 128
```

b) **Non-delivery-information** (C): Information used to construct the non-delivery report. It consists of a *non-delivery-reason-code*, and, optionally, a *non-delivery-diagnostic-code*, and *supplementary-information*, to be placed in the report generation instruction.

```
NonDeliveryInformation ::= SEQUENCE {
    reason             [0] NonDeliveryReasonCode,
    diagnostic         [1] NonDeliveryDiagnosticCode OPTIONAL,
    supplementary-information [2] SupplementaryInformation OPTIONAL }
```

NOTE – In practice, the Error-processing procedure of ITU-T Rec. X.411 | ISO/IEC 10021-4 may cause redirection rather than the generation of a non-delivery report.

c) **Alias-redirection** (C): An OR-address to which the message is to be redirected. If **edit** is *false*, then the **redirection-address** replaces the OR-address present in the message. If edit is *true*, the new OR-address is constructed as follows. That part of the OR-address that was used to locate the entry in which this item of routing-advice was found is discarded. The new OR-address is formed by concatenating the redirection-address with the remaining subordinate elements of the original OR-address.

```
AliasRedirection ::= SEQUENCE {
    redirection-address [0] ORAddress,
    edit                [1] BOOLEAN DEFAULT TRUE }
```

NOTE – If edit is *false*, it will often be useful to set the entry's Next Level Complete attribute *true* to prevent the unintended redirection of addresses which contain OR-address elements in addition to those intended for redirection.

d) **DL-expansion-information** (C): Information that enables the routing-MTA to expand the distribution list identified by the OR-address or to transfer it towards one of its expansion points. **DL-expansion-routing-collectives** identifies the routing-collectives capable of performing the DL expansion. If this identifies this routing-MTA, or one of its superior routing-collectives, the distribution list may be expanded locally. **DL-name**, if present, identifies the Directory entry which holds details of the DL members. If **any-MTA-may-expand** is *true*, then again, the distribution list may be expanded locally. Otherwise, the message should be routed to one of the DL-expanding-routing-collectives identified.

```
DLExpansionInformation ::= SEQUENCE {
    dl-expansion-routing-collectives [0] SET OF TargetRoutingCollective,
    dl-name             [1] MHSDistributionListName OPTIONAL,
    any-mta-may-expand [2] BOOLEAN DEFAULT FALSE }

MHSDistributionListName ::= DistinguishedName
```

e) **Double-envelope-information** (C): Advises the routing-MTA to place the message in an inner-envelope content-type (see 8.2.1.1.1.34 of ITU-T Rec. X.411 | ISO/IEC 10021-4), and supplies information to construct the outer-envelope in which the inner-envelope is to be conveyed. This information comprises the **envelope-opener** (the Directory name and OR-address of an entity which will extract the original message from the inner-envelope), and the preference order for security algorithms to be used for content-confidentiality, message-token-encrypted-data, message-origin-authentication-check, and the signature of the token. The routing-MTA uses the first algorithm in the preference order which is supported by both the routing-MTA and by the envelope-opener. The algorithm-information contains an algorithm-identifier, and optionally information to select an appropriate Certificate for that algorithm for the originator or recipient or

both (depending on the requirements of the algorithm). Certificate-selector information is required only if the Directory entry may contain more than one Certificate for the identified algorithm.

```
DoubleEnvelopeInformation ::= SEQUENCE {
   envelope-opener  [0] ORAddressAndDirectoryName,
   content-confidentiality-algorithm-preference [1] SEQUENCE OF AlgorithmInformation,
   key-encryption-algorithm-preference  [2] SEQUENCE OF AlgorithmInformation OPTIONAL,
   message-origin-algorithm-preference  [3] SEQUENCE OF AlgorithmInformation OPTIONAL,
   token-signature-algorithm-preference [4] SEQUENCE OF AlgorithmInformation OPTIONAL,
   ... }

ORAddressAndDirectoryName ::= ORName -- with both Directory name and OR-address present

AlgorithmInformation ::= SEQUENCE {
   algorithm-identifier             [0] AlgorithmIdentifier,
   originator-certificate-selector  [1] CertificateAssertion OPTIONAL,
   recipient-certificate-selector   [2] CertificateAssertion OPTIONAL }
```

NOTE – The content-confidentiality-algorithm-preference may identify symmetric or asymmetric encryption algorithms; the key-encryption-algorithm-preference identifies asymmetric encryption algorithms, and is only used where the selected content-confidentiality-algorithm-preference identifies a symmetric encryption algorithm; the message-origin-algorithm-preference identifies signature algorithms.

8.2.2 Expression Matches

The **Expression Matches** attribute type, which is single-valued, contains information which enables a routing-MTA to relate routing-advice to an OR-address that satisfies an OR-address pattern match expression.

```
expressionMatches ATTRIBUTE ::= {
    WITH SYNTAX        ExpressionMatches
    SINGLE VALUE       TRUE
    ID                 id-at-mhs-expression-match }

ExpressionMatches ::= SEQUENCE OF ExpressionMatch
```

Each expression-match is ordered in sequence, such that the first of these that an OR-address satisfies provides the most pertinent routing-advice for that OR-address.

```
ExpressionMatch ::= SEQUENCE {
    filter-set         SET OF ORAddressFilter,
    routing-advice     RoutingAdvice }

ORAddressFilter ::= SEQUENCE {
    attribute-type     CHOICE {
           standard-attribute        INTEGER,
           domain-defined-attribute  UniversalOrBMPString {
                                     ub-domain-defined-attribute-type-length} },
    pattern            ExtendedRegularExpression }

ExtendedRegularExpression ::= UniversalOrBMPString {ub-extended-regular-expression}

ub-extended-regular-expression INTEGER ::= 1024
```

The components of an expression-match are defined as follows:

a) **Filter-set** (M): A set of OR-address-filters. If an OR-address satisfies all the OR-address-filters in a filter-set, then the corresponding routing-advice shall apply.

   Each OR-address-filter identifies an **attribute-type** (standard or domain-defined) and a corresponding **pattern**. Each **standard-attribute** is identified by the Integers used in Figure 2 of ITU-T Rec. X.411 | ISO/IEC 10021-4 to identify the *extension standard attributes*. The pattern is constructed as an extended regular expression, as defined in ISO/IEC 9945-2.

b) **Routing-advice** (M): An item of routing-advice which applies to OR-addresses that satisfy the filter-set. If the alias-redirection alternative of routing-advice is specified, the value of its edit component will normally be *false*.

   NOTE – If *true*, that part of the OR-address that was used to locate the entry in which this Expression Matches attribute was found is discarded, and the new OR-address is formed by concatenating the redirection-address with the remaining subordinate elements of the original OR-address. The value of the matching filter-set is not used in forming the new OR-address.

8.2.3 Next Level Complete

The **Next Level Complete** attribute type, which is single-valued, indicates by its presence or absence whether the set of immediate subordinate entries of the present entry is complete, i.e. whether a subordinate entry is actually present for every OR-address-element allocated at this point in the OR-address name-space.

```
nextLevelComplete       ATTRIBUTE ::= {
    WITH SYNTAX         NULL
    SINGLE VALUE        TRUE
    ID                  id-at-mhs-next-level-complete }
```

8.2.4 Recipient MD Assigned Alternate Recipient

The **Recipient MD Assigned Alternate Recipient** attribute type identifies an alternate-recipient, assigned by the administrator of the OR-address-subtree, to receive messages for unknown or ambiguously addressed recipients with this OR-address prefix. If redirection to an MD-specified alternate-recipient is required for an OR-address, the Error-processing procedure, defined in 14.3.6 of ITU-T Rec. X.411 | ISO/IEC 10021-4, makes use of the value of this attribute, taken from the last OR-address subtree entry read (in the Routing-decision procedure - see 9.1.2).

```
recipientMDAssignedAlternateRecipient ATTRIBUTE ::= {
    WITH SYNTAX         ORName
    SINGLE VALUE        FALSE
    COLLECTIVE          TRUE
    ID                  id-at-mhs-recipient-md-assigned-alternate-recipient }
```

NOTES

1 Different values of the attribute may apply within different regions of a single MD.

2 The attribute is multi-valued to conform to the rules governing the definition of collective attributes but should normally contain a single value. Where a different value appears in a superior part of the OR-address-subtree, appropriate scope must be given to the collective attribute so that only a single value appears in any one entry. The effect of multiple values is undefined and therefore unpredictable.

8.3 OR-address Element subclasses

The OR-address Element object class is an abstract object class that models the common properties of entries in the OR-address-subtree. The structural object classes derived from this, used to construct entries of each of the OR-address forms, are defined in 8.3.1 - 8.3.6.

8.3.1 OR-address Subtree Base object class

The **OR-address Subtree Base** object class is a structural object class used to represent the base of an OR-address-subtree.

```
oRAddressSubtreeBase    OBJECT-CLASS ::= {
    SUBCLASS OF         {oRAddressElement}
    KIND                structural
    MUST CONTAIN        {commonName}
    ID                  id-oc-oraddress-subtree-base }
```

8.3.2 Common OR-address object classes

The MHS Country, MHS ADMD, and MHS PRMD object classes are common to all OR-address forms:

```
mHSCountry              OBJECT-CLASS ::= {
    SUBCLASS OF         {oRAddressElement}
    KIND                structural
    MUST CONTAIN        {mHSCountryName}
    ID                  id-oc-mhs-country }

mHSADMD                 OBJECT-CLASS ::= {
    SUBCLASS OF         {oRAddressElement}
    KIND                structural
    MUST CONTAIN        {mHSADMDName}
    ID                  id-oc-mhs-admd }

mHSPRMD                 OBJECT-CLASS ::= {
    SUBCLASS OF         {oRAddressElement}
    KIND                structural
    MUST CONTAIN        {mHSPRMDName}
    ID                  id-oc-mhs-prmd }
```

8.3.3 Mnemonic OR-address object classes

The MHS Organization, MHS Organizational Unit, MHS Common Name, MHS Surname, MHS Given Name, MHS Initials, and MHS Generation Qualifier object classes are specific to the Mnemonic OR-address form:

```
mHSOrganization         OBJECT-CLASS ::= {
    SUBCLASS OF         {oRAddressElement}
    KIND                structural
    MUST CONTAIN        {mHSOrganizationName}
    ID                  id-oc-mhs-organization }
```

```
mHSOrganizationalUnit  OBJECT-CLASS ::= {
    SUBCLASS OF        {oRAddressElement}
    KIND               structural
    MUST CONTAIN       {mHSOrganizationalUnitName}
    ID                 id-oc-mhs-organizational-unit }

mHSCommonName          OBJECT-CLASS ::= {
    SUBCLASS OF        {oRAddressElement}
    KIND               structural
    MUST CONTAIN       {mHSCommonNameAttribute}
    ID                 id-oc-mhs-common-name }

mHSSurname             OBJECT-CLASS ::= {
    SUBCLASS OF        {oRAddressElement}
    KIND               structural
    MUST CONTAIN       {mHSSurnameAttribute}
    ID                 id-oc-mhs-surname }

mHSGivenName           OBJECT-CLASS ::= {
    SUBCLASS OF        {oRAddressElement}
    KIND               structural
    MUST CONTAIN       {mHSGivenNameAttribute}
    ID                 id-oc-mhs-given-name }

mHSInitials            OBJECT-CLASS ::= {
    SUBCLASS OF        {oRAddressElement}
    KIND               structural
    MUST CONTAIN       {mHSInitialsAttribute}
    ID                 id-oc-mhs-initials }

mHSGenerationQualifier OBJECT-CLASS ::= {
    SUBCLASS OF        {oRAddressElement}
    KIND               structural
    MUST CONTAIN       {mHSGenerationQualifierAttribute}
    ID                 id-oc-mhs-generation-qualifier }
```

8.3.4 Terminal OR-address object classes

The MHS Network Address, MHS Extended Network Address, MHS Terminal Identifier, and MHS Terminal Type object classes are specific to the Terminal OR-address form:

```
mHSNetworkAddress      OBJECT-CLASS ::= {
    SUBCLASS OF        {oRAddressElement}
    KIND               structural
    MUST CONTAIN       {mHSNetworkAddressAttribute}
    ID                 id-oc-mhs-network-address }

mHSExtendedNetworkAddress OBJECT-CLASS ::= {
    SUBCLASS OF        {oRAddressElement}
    KIND               structural
    MUST CONTAIN       {mHSExtendedNetworkAddressAttribute}
    ID                 id-oc-mhs-extended-network-address }

mHSTerminalIdentifier  OBJECT-CLASS ::= {
    SUBCLASS OF        {oRAddressElement}
    KIND               structural
    MUST CONTAIN       {mHSTerminalIdentifierAttribute}
    ID                 id-oc-mhs-terminal-identifier }

mHSTerminalType        OBJECT-CLASS ::= {
    SUBCLASS OF        {oRAddressElement}
    KIND               structural
    MUST CONTAIN       {mHSTerminalTypeAttribute}
    ID                 id-oc-mhs-terminal-type }
```

8.3.5 Numeric OR-address object classes

The MHS Numeric User Identifier object class is specific to the Numeric OR-address form:

```
mHSNumericUserIdentifier OBJECT-CLASS ::= {
    SUBCLASS OF        {oRAddressElement}
    KIND               structural
    MUST CONTAIN       {mHSNumericUserIdentifierAttribute}
    ID                 id-oc-mhs-numeric-user-identifier }
```

8.3.6 Postal OR-address object classes

The MHS PDS Name, MHS Physical Delivery Country, and MHS Postal Code object classes are specific to the Postal OR-address form:

```
mHSPDSName              OBJECT-CLASS ::= {
    SUBCLASS OF         {oRAddressElement}
    KIND                structural
    MUST CONTAIN        {mHSPDSNameAttribute}
    ID                  id-oc-mhs-pds-name }

mHSPhysicalDeliveryCountry OBJECT-CLASS ::= {
    SUBCLASS OF         {mHSCountry}
    KIND                structural
    ID                  id-oc-mhs-physical-delivery-country-name }

mHSPostalCode           OBJECT-CLASS ::= {
    SUBCLASS OF         {oRAddressElement}
    KIND                structural
    MUST CONTAIN        {mHSPostalCodeAttribute}
    ID                  id-oc-mhs-postal-code }
```

8.4 OR-address Element Names

The attributes used to name the various OR-address Elements are defined below. Each is derived from the OR-address Element Name attribute type.

The **OR-address Element Name** attribute type, which is single-valued, is the direct supertype of every attribute used to form the RDN of an entry in an OR-address-subtree.

```
oRAddressElementName    ATTRIBUTE ::= {
    SUBTYPE OF          name -- see ITU-T Rec. X.520 | ISO/IEC 9594-6 --
    SINGLE VALUE        TRUE
    ID                  id-at-oraddress-element-name }
```

8.4.1 Common OR-address Element Names

The MHS Country Name, MHS ADMD Name, and MHS PRMD Name attribute types are common to all OR-address forms:

```
mHSCountryName          ATTRIBUTE ::= {
    SUBTYPE OF          oRAddressElementName -- contains ISO 3166 and X.121 codes only --
    WITH SYNTAX         DirectoryString {ub-country-name-numeric-length}
    ID                  id-at-mhs-country-name }

mHSADMDName             ATTRIBUTE ::= {
    SUBTYPE OF          oRAddressElementName
    WITH SYNTAX         DirectoryString {ub-domain-name-length}
    ID                  id-at-mhs-admd-name }

mHSPRMDName             ATTRIBUTE ::= {
    SUBTYPE OF          oRAddressElementName
    WITH SYNTAX         DirectoryString {ub-domain-name-length}
    ID                  id-at-mhs-prmd-name }
```

8.4.2 Mnemonic OR-address Element Names

The MHS Organization Name, MHS Organizational Unit Name, MHS Common Name Attribute, MHS Surname Attribute, MHS Given Name Attribute, MHS Initials Attribute, and MHS Generation Qualifier Attribute attribute types are specific to the Mnemonic OR-address form:

```
mHSOrganizationName     ATTRIBUTE ::= {
    SUBTYPE OF          oRAddressElementName
    WITH SYNTAX         DirectoryString {ub-organization-name-length}
    ID                  id-at-mhs-organization-name }

mHSOrganizationalUnitName ATTRIBUTE ::= {
    SUBTYPE OF          oRAddressElementName
    WITH SYNTAX         DirectoryString {ub-organizational-unit-name-length}
    ID                  id-at-mhs-organizational-unit-name }

mHSCommonNameAttribute  ATTRIBUTE ::= {
    SUBTYPE OF          oRAddressElementName
    WITH SYNTAX         DirectoryString {ub-common-name-length}
    ID                  id-at-mhs-common-name }

mHSSurnameAttribute     ATTRIBUTE ::= {
    SUBTYPE OF          oRAddressElementName
    WITH SYNTAX         DirectoryString {ub-surname-length}
    ID                  id-at-mhs-surname }

mHSGivenNameAttribute   ATTRIBUTE ::= {
    SUBTYPE OF          oRAddressElementName
    WITH SYNTAX         DirectoryString {ub-given-name-length}
    ID                  id-at-mhs-given-name }
```

```
mHSInitialsAttribute   ATTRIBUTE ::= {
    SUBTYPE OF         oRAddressElementName
    WITH SYNTAX        DirectoryString {ub-initials-length}
    ID                 id-at-mhs-initials }

mHSGenerationQualifierAttribute ATTRIBUTE ::= {
    SUBTYPE OF         oRAddressElementName
    WITH SYNTAX        DirectoryString {ub-generation-qualifier-length}
    ID                 id-at-mhs-generation-qualifier }
```

8.4.3 Terminal OR-address Element Names

The MHS Network Address Attribute, MHS Extended Network Address Attribute, MHS Terminal Identifier Attribute, and MHS Terminal Type Attribute attribute types are specific to the Terminal OR-address form:

```
mHSNetworkAddressAttribute ATTRIBUTE ::= {
    SUBTYPE OF         oRAddressElementName
    WITH SYNTAX        DirectoryString {ub-x121-address-length}
    ID                 id-at-mhs-network-address }

mHSExtendedNetworkAddressAttribute ATTRIBUTE ::= {
    SUBTYPE OF         oRAddressElementName
    WITH SYNTAX        DirectoryString {ub-extended-network-address-length}
    ID                 id-at-mhs-extended-network-address }

ub-extended-network-address-length INTEGER ::= 256

mHSTerminalIdentifierAttribute ATTRIBUTE ::= {
    SUBTYPE OF         oRAddressElementName
    WITH SYNTAX        DirectoryString {ub-terminal-id-length}
    ID                 id-at-mhs-terminal-identifier }

mHSTerminalTypeAttribute ATTRIBUTE ::= {
    SUBTYPE OF         oRAddressElementName
    WITH SYNTAX        DirectoryString {ub-terminal-type-length}
    ID                 id-at-mhs-terminal-type }

ub-terminal-type-length INTEGER ::= 5
```

8.4.4 Numeric OR-address Element Names

The MHS Numeric User Identifier Attribute attribute type is specific to the Numeric OR-address form:

```
mHSNumericUserIdentifierAttribute ATTRIBUTE ::= {
    SUBTYPE OF         oRAddressElementName
    WITH SYNTAX        DirectoryString {ub-numeric-user-id-length}
    ID                 id-at-mhs-numeric-user-identifier }
```

8.4.5 Postal OR-address Element Names

The MHS PDS Name Attribute and MHS Postal Code Attribute attributes types are specific to the Postal OR-address form:

```
mHSPDSNameAttribute ATTRIBUTE ::= {
    SUBTYPE OF         oRAddressElementName
    WITH SYNTAX        DirectoryString {ub-pds-name-length}
    ID                 id-at-mhs-pds-name-attribute }

mHSPostalCodeAttribute ATTRIBUTE ::= {
    SUBTYPE OF         oRAddressElementName
    WITH SYNTAX        DirectoryString {ub-postal-code-length}
    ID                 id-at-mhs-postal-code }
```

8.5 Generation of OR-address-element attributes

Each attribute is generated from the corresponding element of an OR-address. Every attribute is a subtype of the OR-address-element-name attribute type.

Additional rules apply for generation of the following attribute types:

a) *MHS Country Name:* This attribute holds either a two-character ISO 3166 alphabetic code or a three character X.121 numeric code. When constructing an OR-address subtree, where an OR-address has equivalent ISO 3166 and X.121 values, the former shall be present in the object entry for MHS Country, and the latter shall be present as an alias entry.

b) *MHS Organizational Unit Name:* When constructing an OR-address-subtree, the sequential order of organizational-unit-names in the OR-address shall be reflected in the hierarchy of MHS Organizational Unit objects, with the first of the organizational-unit-names used to form the superior entry and successive organizational-unit-names used to form successive subordinate entries.

c) *MHS Terminal Type Attribute:* To create a value of the MHS Terminal Type attribute, the Integer value of terminal-type is mapped to one of the strings defined in F.3.2.2 of ITU-T Rec. X.402 | ISO/IEC 10021-2.

d) *MHS Extended Network Address Attribute:* To create a value of this attribute, the value of Extended Network Address is mapped to a string using the keyword notation defined in F.3.2 of ITU-T Rec. X.402 | ISO/IEC 10021-2, as if it were an OR-address containing only the Extended Network Address.

NOTE – As ITU-T Rec. X.402 (1995) | ISO/IEC 10021-2: 1996 defines notation for only a subset of the Extended Network Address, the mapping of other variants will be a local matter.

8.6 OR-address-subtree name forms

The name forms defined below specify how the entries in an OR-address-subtree may be named.

```
mHSCountryNameForm      NAME-FORM ::= {
    NAMES               mHSCountry
    WITH ATTRIBUTES     {mHSCountryName}
    ID                  id-nf-mhs-country }

mHSADMDNameForm         NAME-FORM ::= {
    NAMES               mHSADMD
    WITH ATTRIBUTES     {mHSADMDName}
    ID                  id-nf-mhs-admd }

mHSPRMDNameForm         NAME-FORM ::= {
    NAMES               mHSPRMD
    WITH ATTRIBUTES     {mHSPRMDName}
    ID                  id-nf-mhs-prmd }

mHSOrganizationNameForm NAME-FORM ::= {
    NAMES               mHSOrganization
    WITH ATTRIBUTES     {mHSOrganizationName}
    ID                  id-nf-mhs-organization }

mHSOrganizationalUnitNameForm NAME-FORM ::= {
    NAMES               mHSOrganizationalUnit
    WITH ATTRIBUTES     {mHSOrganizationalUnitName}
    ID                  id-nf-mhs-organizational-unit }

mHSCommonNameForm       NAME-FORM ::= {
    NAMES               mHSCommonName
    WITH ATTRIBUTES     {mHSCommonNameAttribute}
    ID                  id-nf-mhs-common-name }

mHSSurnameNameForm      NAME-FORM ::= {
    NAMES               mHSSurname
    WITH ATTRIBUTES     {mHSSurnameAttribute}
    ID                  id-nf-mhs-surname }

mHSGivenNameNameForm    NAME-FORM ::= {
    NAMES               mHSGivenName
    WITH ATTRIBUTES     {mHSGivenNameAttribute}
    ID                  id-nf-mhs-given-name }

mHSInitialsNameForm     NAME-FORM ::= {
    NAMES               mHSInitials
    WITH ATTRIBUTES     {mHSInitialsAttribute}
    ID                  id-nf-mhs-initials }

mHSGenerationQualifierNameForm NAME-FORM ::= {
    NAMES               mHSGenerationQualifier
    WITH ATTRIBUTES     {mHSGenerationQualifierAttribute}
    ID                  id-nf-mhs-generation-qualifier }

mHSNetworkAddressNameForm NAME-FORM ::= {
    NAMES               mHSNetworkAddress
    WITH ATTRIBUTES     {mHSNetworkAddressAttribute}
    ID                  id-nf-mhs-network-address }

mHSExtendedNetworkAddressNameForm NAME-FORM ::= {
    NAMES               mHSExtendedNetworkAddress
    WITH ATTRIBUTES     {mHSExtendedNetworkAddressAttribute}
    ID                  id-nf-mhs-extended-network-address }

mHSTerminalIdentifierNameForm NAME-FORM ::= {
    NAMES               mHSTerminalIdentifier
    WITH ATTRIBUTES     {mHSTerminalIdentifierAttribute}
    ID                  id-nf-mhs-terminal-identifier }

mHSTerminalTypeNameForm NAME-FORM ::= {
    NAMES               mHSTerminalType
    WITH ATTRIBUTES     {mHSTerminalTypeAttribute}
    ID                  id-nf-mhs-terminal-type }
```

```
mHSNumericUserIdentifierNameForm NAME-FORM ::= {
    NAMES               mHSNumericUserIdentifier
    WITH ATTRIBUTES     {mHSNumericUserIdentifierAttribute}
    ID                  id-nf-mhs-numeric-user-identifier }

mHSPDSNameNameForm NAME-FORM ::= {
    NAMES               mHSPDSName
    WITH ATTRIBUTES     {mHSPDSNameAttribute}
    ID                  id-nf-mhs-pds-name }

mHSPhysicalDeliveryCountryNameForm NAME-FORM ::= {
    NAMES               mHSPhysicalDeliveryCountry
    WITH ATTRIBUTES     {mHSCountryName}
    ID                  id-nf-mhs-physical-delivery-country }

mHSPostalCodeNameForm NAME-FORM ::= {
    NAMES               mHSPostalCode
    WITH ATTRIBUTES     {mHSPostalCodeAttribute}
    ID                  id-nf-mhs-postal-code }
```

9 Procedures

Two categories of procedure are required to support MHS-routing:

a) Procedures followed by the routing-MTA when performing MHS-routing as specified in this part of ISO/IEC 10021.

b) Procedures followed by an MHS administrator to maintain the OR-address-subtrees for which the administrator has responsibility, and to maintain the routing-collectives and connection-groups that are locally managed.

The procedures defined herein consider routing in terms of an individual recipient. Where a message has more than one recipient, the procedures in ITU-T Rec. X.411 | ISO/IEC 10021-4 specify that where the routing instructions for different recipients (considered separately) identify the same next MTA then a single copy of the message will be transferred to that MTA. However, it is possible to consider the routing decisions for multiple recipients collectively to attempt a greater degree of copy optimisation than is achieved by considering the recipients separately. For example, if the (separate) routing-collectives identified for different recipients share a common parent routing-collective, then it may be considered desirable to transfer a single copy of the message to that parent routing-collective instead of transferring multiple separate copies to the individual (child) routing-collectives. However, the desirability of introducing additional hops to reduce the number of copies transferred depends critically on network bandwidth and cost. This process, often called Late Fan-Out, is beyond the scope of this standard but it is not precluded.

9.1 Routing-MTA procedures

An MTA that conforms to this part of ISO/IEC 10021 shall be capable of performing the following procedures:

a) The Front-end procedure (9.1.1). This amends the Front-end procedure definition in 14.3.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

b) The Routing-decision procedure (9.1.2). This replaces the Routing-decision procedure definition in 14.3.4 of ITU-T Rec. X.411 | ISO/IEC 10021-4. It contains two additional procedures required for MHS-routing (see Figure 3). The procedure also affects the Report-routing procedure defined in 14.4.4.4 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

c) The OR-address-subtree-read procedure (9.1.3).

d) The Local-delivery-evaluation procedure (9.1.4).

e) The Routing-knowledge-acquisition procedure (9.1.5).

f) The MTA-bind-in procedure (9.1.6). This replaces the MTA-bind-in procedure definition in 14.9.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

g) The MTA-bind-out procedure (9.1.7). This replaces the MTA-bind-out procedure definition in 14.9.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

h) The Message-in, Probe-in, and Report-in procedures defined in 14.10.1, 14.10.2, and 14.10.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4 shall be performed as modified by clause 9.1.8 of this part of ISO/IEC 10021.

The routing-decision procedure normally calls the OR-address-subtree-read procedure at least once, and may call the Local-delivery-evaluation procedure.

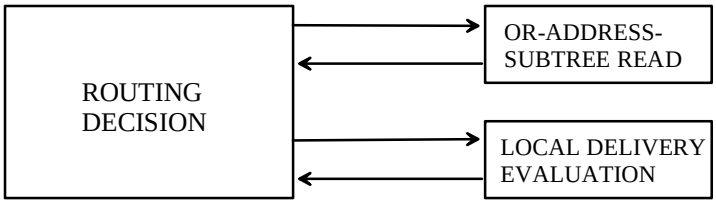

Figure 3 - Organization of procedures within the Routing-decision procedure

9.1.1 Amendment to the Front-end procedure

The Front-end procedure defined in 14.3.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4 is amended as follows for an MTA conforming to this part of ISO/IEC 10021. Step 5 of 14.3.2.4 (Loop detection) is replaced with the following:

a) The trace-information and internal-trace-information is examined to determine whether loop detection will be performed. If the internal-trace-information is complete (i.e. fully represents the information present in trace-information), then loop detection will be handled by the Routing-decision procedure, and the Front-end procedure resumes at step 6.

b) Otherwise, external trace-information is identified by deleting from the end of trace-information those elements whose global-domain-identifier corresponds to that of this routing-MTA. In addition, elements of trace-information generated prior to the most recent DL-expansion, redirection, conversion, or re-routed action are not considered. Loop detection is undertaken by examining the remaining elements of this external trace-information. If the global-domain-identifier of any of these trace-information elements identifies the domain of this routing-MTA, then the message is declared to be looping. The *non-delivery-reason-code* is set to *transfer-failure*, and the *non-delivery-diagnostic-code* is set to *loop-detected*. The procedure then terminates.

c) Otherwise, the Front-end procedure resumes at step 6.

9.1.2 Routing-decision procedure

The procedure generates a routing instruction for a single message recipient. This definition replaces the Routing-decision procedure definition in 14.3.4 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

9.1.2.1 Arguments

a) A message recipient plus the per-recipient instruction, if any, applicable to this recipient.

b) The per-message instruction, if any, applicable to this message. Other message fields are also accessible to the procedure as required.

9.1.2.2 Results

A new or possibly revised routing instruction applicable to this recipient. Possible instructions are:

a) deliver to a local recipient;

b) relay to another MTA;

c) generate a report indicating delivery failure. The *non-delivery-reason-code* and, optionally, *non-delivery-diagnostic-code* and *supplementary-information* are included in the instruction;

d) expand the distribution list represented by this recipient;

e) redirect to a preferred address or to a recipient specified alternate recipient.

9.1.2.3 Errors

None. Error conditions are recorded in the routing instruction.

9.1.2.4 Procedure description

NOTES

1 To ensure the security-policy is not violated during routing, the *message-security-label* should be checked as appropriate against the *security-context*, and connection-groups whose *security-context* is incompatible with that of the message should be disregarded.

2 This clause describes cases where specific values of supplementary-information are required. Elsewhere, it may be given other values as a local matter.

The Routing-decision procedure is described in the following steps:

a) If the per-message instruction indicates a relay failure, then the procedure attempts to identify an alternative next-MTA destination for this recipient. The procedure defined below is followed (from step *d*), except that when a next-MTA is selected which matches that indicated in the relay failure, the procedure discards the selection and continues until it locates an alternative next-MTA destination. If an alternative destination is found, the message's *internal-trace-information* is updated with a *re-routed* routing-action to reflect the fact that the message has been re-routed (see 12.3.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4). If the message would have crossed a domain boundary (if it had been transferred successfully to the next-MTA) then the *trace-information* is similarly updated. The procedure returns a relay instruction to the alternative destination and terminates.

   If the procedure defined below (from step *d*) identifies a candidate next-MTA, the routing-MTA verifies that this route has not already been attempted, as follows:

   1) The routing-MTA performs a Directory Read on the next-MTA's MHS Message Transfer Agent entry, and fetches the global-domain-identifier and MTA-name of the next-MTA.

   2) The routing-MTA examines relevant elements of internal-trace-information (if any), i.e. those elements that were generated subsequent to the last DL-expansion, redirection, conversion, or re-routed action.

   3) For each element in turn, the global-domain-identifier and MTA-name components are compared which those of the candidate next-MTA. If a match is found, the preceding internal trace element is examined. If this identifies this routing-MTA, then that route has already been attempted and found ineffective, and the procedure will continue its attempt to locate an effective next-MTA.

      If the preceding internal trace element identifies some other routing-MTA then the route may be, but is not certain to be unproductive. This next-MTA is noted as a non-optimal route, and the procedure continues in an attempt to find a better route.

b) If the per-recipient instruction indicates a delivery failure, then the procedure returns a report generation instruction for this recipient. The *non-delivery-reason-code* and *non-delivery-diagnostic-code* are those supplied by the Message-delivery or Probe-delivery-test procedures. The procedure then terminates.

c) If the recipient is specified by an OR-name which contains only a *directory-name* (which may happen following distribution list expansion, if a DL member is specified only by *directory-name*), the routing-MTA attempts to acquire the OR-address from the Directory. If the OR-address cannot be determined, the procedure returns a report generation instruction for this recipient. The *non-delivery-reason-code* is set to *directory-operation-unsuccessful* and the *non-delivery-diagnostic-code* may be set according to the problem encountered. The procedure then terminates.

d) The routing-MTA invokes the OR-address-subtree-read procedure, as defined in 9.1.3, initially using the first configured OR-address subtree. If this read does not yield routing-advice, the procedure continues at step *j*.

e) One of the following steps is taken according to the routing-advice found:

   1) *Target-routing-collective*: The routing-advice identifies a target routing-collective for the message. The routing-MTA compares its own routing-collective name with that of the target routing-collective, and, if these match, the Local-delivery-evaluation procedure is invoked (see 9.1.4), and the routing instruction returned by it is inspected. If this contains a relay instruction which specifies a next-MTA that has been the subject of a previous relay failure for this message, or represents a non-optimal route, then the procedure continues at step *j*. Otherwise the routing instruction returned by the Local-delivery-evaluation procedure is returned and the procedure terminates.

      If the target routing-collective is a superior of the routing-MTA (i.e. the name of the routing-MTA contains that of the target routing-collective), the routing-MTA abandons the present OR-address-subtree and continues at step *j*.

      Otherwise, the procedure continues at step *f*.

   2) *Non-delivery-information*: The routing-advice indicates that delivery to this OR-address is not possible. The procedure returns a report generation instruction for this recipient. The *non-delivery-reason-code*, *non-delivery-diagnostic-code*, and *supplementary-information* are set to the corresponding values present in the routing-advice. The procedure then terminates.

   3) *Alias-redirection*: The routing-advice specifies a preferred address for the recipient to which the message is to be redirected. The replacement OR-address is constructed as defined in 8.2.1. A

redirection instruction is generated containing the replacement OR-address, with redirection reason set to *alias*. The procedure then terminates.

4) *DL-expansion-information*: The routing-advice indicates that the recipient is a distribution list. The message's *DL-expansion-prohibited* argument is examined. If the value is *DL-expansion-prohibited*, or the security-policy prohibits the use of a distribution list, then the procedure returns a report generation instruction for this recipient. The *non-delivery-reason-code* is set to *unable-to-transfer*, and the *non-delivery-diagnostic-code* is set to *DL-expansion-prohibited*. The procedure then terminates.

   Otherwise, if one of the DL-expansion-routing-collectives identifies this routing-MTA then the procedure returns a routing instruction to expand the distribution list, and terminates.

   If one of the DL-expansion-routing-collectives identifies a superior of the routing-MTA, or if any-MTA-may-expand is *true*, and the routing-MTA is capable of performing the DL expansion and is so permitted by policy, then the procedure returns a routing instruction to expand the distribution list, and terminates.

   Otherwise, one of the DL-expansion-routing-collectives that is not a superior of the routing-MTA is selected as the target routing-collective and the procedure continues at step *f*.

   If all the DL-expansion-routing-collectives are superiors of this routing-MTA, and local policy does not permit expansion of the list then a configuration error is indicated. The procedure returns a report generation instruction for this recipient with the *non-delivery-reason-code* set to *unable-to-transfer*, and the *non-delivery-diagnostic-code* set to *DL-expansion-failure*. The procedure then terminates.

5) *Double-envelope-information*: The routing-advice indicates that the message is to be placed in an inner-envelope content-type. The Supported Algorithms and User Certificate attributes are read from the Directory entry identified by the Directory name for the envelope-opener. The MTS-APDU is encrypted using the first algorithm in the content-confidentiality-algorithm-preference order which is supported by both the routing-MTA and by the envelope-opener; this may be an asymmetric algorithm, or if this is a symmetric algorithm then a random content-confidentiality-key is generated and used to encrypt the content, and a message-token created with this key encrypted using the first algorithm in the key-encryption-algorithm-preference order which is supported by both the routing-MTA and by the envelope-opener (which must be an asymmetric algorithm). The public key which is used with the asymmetric algorithm is found by using the algorithm-identifier and recipient-certificate-selector to select an appropriate Certificate of the recipient.

   A new message is created containing the encrypted MTS-APDU in its content, the envelope-opener as its recipient, and, if message-origin-algorithm-preference is specified, a message-origin-authentication-check containing a signature of the encrypted content using the first algorithm in the preference order which is supported by both the routing-MTA and by the envelope-opener together with the private key of the routing-MTA corresponding to its Certificate identified by originator-certificate-selector. The procedure then continues at step *d* using the first configured OR-address-subtree.

f) The routing-MTA examines its local-use-tables to determine whether the target routing-collective is a key-routing-collective. If so, a next-MTA to which the message may be transferred is known. If the next-MTA has been the subject of a previous relay failure for this message, or represents a non-optimal route, then the procedure continues at step *j*. Otherwise, the procedure returns a relay instruction requesting transfer to that MTA, and terminates.

g) If the target routing-collective is not a key-routing-collective, its Directory entry is read and a list of the names of its entry-connection-groups retrieved. Any of the routing-MTA's exit-connection-groups that matches the target-routing-collective's entry-connection-groups may be used to transfer the message.

   If the target routing-collective is a routing-MTA, then this is the next-MTA. Otherwise, the routing-MTA performs a subtree Search below the target routing-collective for entries of object class routing-MTA, which contain an entry-connection-group corresponding to any of those matched above. If all candidate next-MTAs so identified have been the subject of a previous relay failure for this message, or represent non-optimal routes, then the procedure continues at step *h*. Otherwise, one of these routing-MTAs is selected as the next-MTA. The procedure returns a relay instruction to that MTA and terminates.

   NOTE – If this step has selected a route that uses a local-exit-connection-group, but the message in question did not originate, redirect, or DL-expand at this MTA, then an unauthorized relay is being requested. As a matter of local policy, the MTA may relay the message regardless, or return it, or seek another route, or generate a non-delivery report.

h) If no pair of matching connection-groups names is found, the routing-MTA examines its local-use-tables to discover whether any of the target's entry-connection-groups are exit-connection-groups available to one of the routing-MTA's key-routing-collectives, i.e. are indirect-exit-connection-groups of this routing-collective. If so, a next-MTA to which the message may be transferred is known. If the next-MTA has been the subject of a previous relay failure for this message, or represents a non-optimal route, then the procedure continues at step *i*. Otherwise, the procedure returns a relay instruction to that MTA and terminates.

i) The routing-MTA abandons the attempt to find a route to the current target routing-collective and selects a new target routing-collective, as follows. It discards the last RDN of the target routing-collective name, and so obtains the name of that entry's parent entry. If the last RDN of this parent entry has an attribute type of Routing Collective Name, then the truncated name is adopted as the new target routing-collective, and the procedure continues at step *f*.

j) If not all of the configured OR-address-subtrees have been examined, the next subtree is selected and the procedure resumes at step *d*.

k) If all OR-address-subtrees have been examined, and no next-MTA relay has been identified, but one or more non-optimal next-MTAs have been identified, then the procedure returns a relay instruction requesting transfer to the first of these, and terminates.

m) The routing-MTA considers returning the message to the MTA from which it was first received. If this is not prohibited by policy, then the procedure returns a relay instruction requesting transfer to that MTA, and terminates.

n) Otherwise, the procedure returns a report generation instruction for this recipient, with the *non-delivery-reason-code* set to *unable-to-transfer*. If a previous relay failure occurred for this recipient, the *non-delivery-diagnostic-code* is set according to the relay failure encountered; otherwise, the *non-delivery-diagnostic-code* is set to *unrecognized-OR-name*. The procedure then terminates.

9.1.3 OR-address-subtree-read procedure

This procedure retrieves the routing-advice associated with an OR-address from an entry which represents that OR-address in an OR-address-subtree.

9.1.3.1 Arguments

a) The OR-address of a message recipient.

b) The Directory name of the base vertex of an OR-address-subtree.

9.1.3.2 Results

An item of routing-advice associated with the message recipient's OR-address, or an indication that no routing-advice was found in this OR-address-subtree for the OR-address presented.

9.1.3.3 Errors

Any of the errors defined for the Directory Read operation.

9.1.3.4 Procedure description

The OR-address-subtree-read procedure is described in the following steps.

a) The routing-MTA transforms the OR-address into a purported Directory name. This is done by mapping elements of the OR-address into corresponding OR-address attributes (see 8.5), and assembling in a prescribed order the RDNs thus produced, prefixed with the name of the OR-address-subtree, to form the purported name (see Annex D). Certain OR-address elements are never used in constructing the purported name; other elements are used in some OR-address forms but not in others. Only those attributes listed below shall be used.

   According to the OR-address form (as defined in 18.5.5 of ITU-T Rec. X.402 | ISO/IEC 10021-2) taken by a given OR-address, OR-address elements are assembled in the following order:

   — *All OR-address forms:* MHS Country, MHS ADMD, MHS PRMD.

   — *Mnemonic OR-address form:* MHS Organization, MHS Organizational Unit (1), MHS Organizational Unit (2), MHS Organizational Unit (3), MHS Organizational Unit (4), MHS Common Name, MHS Surname, MHS Given Name, MHS Initials, MHS Generation Qualifier.

   — *Terminal OR-address form:* MHS Network Address, MHS Terminal Identifier, MHS Terminal Type.

— *Numeric OR-address form:* MHS Numeric User Identifier.

— *Postal OR-address form:* MHS PDS Name, MHS Physical Delivery Country, MHS Postal Code.

b) The routing-MTA invokes a Directory Read operation, with the purported name as the *object* argument, and the following attributes as the *selection* argument:

— Routing Advice;

— Expression Matches;

— Next Level Complete.

c) If the Read is successful, and the identified entry contains an Expression Matches attribute, then the OR-address given as an argument of the procedure is compared against the OR-address-filters of each expression-match. If a match is found, the routing-advice associated with it is returned and the procedure terminates.

d) If the Read is successful, and the identified entry contains a Routing Advice attribute, that attribute-value is returned and the procedure terminates.

e) If the Read is successful and the identified entry does not contain a Routing Advice attribute, the Next Level Complete attribute is inspected, and, if *true*, a Directory Search operation (one level) is invoked with its *selection* argument specifying OR-address attributes. In the result of this Search, only distinct objects (after all aliases have been dereferenced) are considered.

NOTE 1 – A simple implementation may specify all the OR-address attributes in this selection, while a more sophisticated implementation might limit the selection to only those OR-address attributes which are permitted to be subordinate to the identified entry. This choice simply affects the size of the arguments of the Search operation: either choice will yield the same result, provided that the OR-address-subtree conforms to the structure rules of Annex D.

If the result of this Search shows that the identified entry has precisely one subordinate entry (discounting aliases which produce additional references to the same entry), then the purported name is replaced by the Name of this subordinate entry, and the procedure resumes at step *b*.

NOTE 2 – The above implements the requirement contained in 8.2 of ISO/IEC ISP 10611-1:1996, that an OR-address which is an unambiguous underspecification of an actual OR-address should be regarded as identifying that OR-address.

Otherwise the procedure generates an item of routing-advice as follows:

1) The non-delivery-information alternative of routing-advice is chosen.

2) Its reason component is set to *unable-to-transfer*.

3) If the identified entry has no subordinate entry, then the diagnostic component is set to *unrecognized-OR-name*. If the identified entry has two or more subordinate entries, then the diagnostic component is set to *ambiguous-OR-name* and the supplementary-information component may be used to indicate a list of potential values of an additional address element which could be added to disambiguate the address. For example: “Two addresses have surname ‘Jones’; choose given name ‘James’ or ‘John’”.

NOTE 3 – An appropriate sizeLimit should be used in the Search operation to avoid the Search result returning more information than could conceivably be accommodated within supplementary-information.

The procedure returns this routing-instruction and terminates.

f) As in the case above, but where Next Level Complete is *false*, then no further information can be obtained from this OR-address-subtree and the procedure returns an indication that no routing-advice was found for the specified OR-address, and terminates.

g) If the Read fails with a Name Error, indicating *no-such-object*, then the routing-MTA invokes a Directory Read operation with the name returned as the *matched* component of the Name Error (i.e. the longest part of the original name that could be matched) as the *object* argument, and the following attributes as the *selection* argument:

— Routing Advice;

— Expression Matches;

— Next Level Complete.

If the Next Level Complete attribute is *false*, the procedure resumes at step *c*. Otherwise, an item of routing-advice is generated as follows:

1) The non-delivery-information alternative of routing-advice is chosen.

2) Its reason component is set to *unable-to-transfer*.

3) Its diagnostic component is set to *unrecognized-OR-name*, and its *supplementary-information* component may be used to indicate those OR-address elements that were successfully matched in this overspecified address.

The procedure returns this routing-instruction and terminates.

h) If the Read fails with any other error, that error is returned and the procedure terminates.

9.1.4 Local-delivery-evaluation procedure

This procedure generates a routing-instruction for a single message recipient who is purportedly a registered MTS-user of this routing-MTA.

9.1.4.1 Arguments

a) A message recipient plus the per-recipient instruction, if any, applicable to this recipient.

b) The per-message instruction, if any, applicable to this message. Other message fields are also accessible to the procedure as required.

9.1.4.2 Results

A new or possibly revised routing instruction applicable to this recipient. Possible instructions are enumerated in 9.1.2.2.

9.1.4.3 Errors

None. Error conditions are recorded in the routing instruction.

9.1.4.4 Procedure description

The Local-delivery-evaluation procedure is described in step 6 of 14.3.4.4 in ITU-T Rec. X.411 | ISO/IEC 10021-4.

NOTE – References to the Routing-decision procedure in step 6 of 14.3.4.4 in ITU-T Rec. X.411 | ISO/IEC 10021-4 should be read as references to the Local-delivery-evaluation procedure.

9.1.5 Routing-knowledge-acquisition procedure

This procedure acquires the routing knowledge required by a routing-MTA for the performance of MHS-routing. It is invoked when the routing-MTA is initialized.

The procedure has no formal interaction with any other procedure.

NOTE – Local-use-tables should be recomputed at intervals, and whenever the volume of loop detection incidents exceeds some threshold (which may suggest that the routing-MTA is using out-of-date cached information).

9.1.5.1 Arguments

The Directory name that identifies this routing-collective.

9.1.5.2 Results

The following information, that comprises the routing-MTA's **local-use-tables**:

a) The Directory name of this MHS Message Transfer Agent.

b) An ordered list of the names of one or more OR-address-subtrees configured for this routing-MTA.

c) A list of the key-routing-collectives of this routing-MTA, each accompanied by the names of one or more next-MTAs for the routing-collective.

d) A list of the Directory names of the routing-MTA's transit- and local-exit-connection-groups.

e) An ordered list of the indirect-exit-connection-groups of this routing-MTA, each accompanied by the names of one or more next-MTAs for the connection-group.

f) For each of the routing-MTA's entry-connection-groups that is of type enumerated, authentication information required to identify each possible calling MTA.

NOTE – The syntax of local-use-tables used to record MHS MTA name, OR-address-subtrees, key-routing-collectives, connection-groups, and authentication information is a local matter for each MTA and is not prescribed. However, its intended structure may be illustrated in the following example.

EXAMPLE –

```
LocalUseTablesExample ::= SEQUENCE {
    this-mta                        MHSMessageTransferAgentName,
    or-address-subtrees             ORAddressSubtreeNames,
    key-routing-collectives         KeyRoutingCollectives,
    transit-exit-connection-groups  SET OF ConnectionGroupName,
    local-exit-connection-groups    SET OF ConnectionGroupName,
    indirect-exit-connection-groups SEQUENCE OF Connection,
    entry-connection-groups         SET OF AuthenticationInformation }

KeyRoutingCollectives ::= SET OF Route

Route ::= SET {
    routing-collective      RoutingCollectiveName,
    next-mtas               SET OF NextMTA }

NextMTA ::= SET {
    mhs-mta-name            MHSMessageTransferAgentName,
    connection-groups       SET OF ConnectionGroupName }

MHSMessageTransferAgentName ::= DistinguishedName

ConnectionGroupName ::= DistinguishedName

Connection ::= SET {
    connection-group        ConnectionGroupName,
    authentication-method   AuthenticationMethod,
    profiles                SET OF OBJECT IDENTIFIER OPTIONAL,
    next-mtas               SEQUENCE OF MHSMessageTransferAgentName }

AuthenticationInformation ::= SET {
    entry-connection-group  ConnectionGroupName,
    authentication-method   AuthenticationMethod,
    profiles                SET OF OBJECT IDENTIFIER OPTIONAL,
    network-address-reliable BOOLEAN DEFAULT TRUE,
    connection-group-type   CHOICE {
        enumerated-authentication   SET OF CallingMTAAuthentication,
        unenumerated-authentication CHOICE {
            group-mta-password          Password
            no-group-password           NULL } } }

CallingMTAAuthentication ::= SET {
    mhs-mta-name                    MHSMessageTransferAgentName,
    calling-mta-name                MTAName,
    calling-mta-password            Password OPTIONAL,
    calling-presentation-address    SET OF PresentationAddress OPTIONAL }
```

9.1.5.3 Errors

Any Directory Read errors, or inconsistencies in the information retrieved, are reported as errors..

9.1.5.4 Procedure description

For the purposes of exposition, the Routing-knowledge-acquisition procedure is regarded as possessing the following variables:

— current-routing-collective;

— parent-routing-collective.

The procedure is described in the following steps:

a) A Directory Read operation is invoked with the name of this routing-collective as the *object* argument, and the following attributes as the *selection* argument:

— MHS Message Transfer Agent Name;

— OR-address Subtrees;

— Transit Exit Connection Group Name;

— Local Exit Connection Group Name;

— Entry Connection Group Name.

The content of these attributes is recorded in the local-use-tables.

b) The routing-MTA identifies its key-routing-collectives and indirect-exit-connection-groups as follows:

1) The routing-MTA initializes the current-routing-collective variable with the Directory name specified as the argument of the procedure.

2) The routing-MTA discards the last routing-collective-name component of the current-routing-collective to form the name of the parent-routing-collective. If the terminal RDN of the parent-routing-collective is not of attribute type routing-collective-name, then the complete routing-collective-subtree has been examined and the procedure continues at step *c*.

3) The routing-MTA invokes a Directory Search operation (one level) relative to the parent entry, which retrieves the entry-connection-group-names, the routing-collective names, the transit-exit-connection-group-names, and, if present, the MHS Message Transfer Agent Name of each of the parent entry's immediate subordinates. Of these immediate subordinate entries, the siblings of the current-routing-collective are recorded in the routing-MTA's local-use-tables as key-routing-collectives (i.e. the current-routing-collective is itself omitted). The next-MTAs associated with each of these will be discovered in steps *c* and *d*.

4) For each of the key-routing-collectives found, the routing-MTA identifies each transit-exit-connection-group of the key-routing-collective, that is not an exit-connection-group of this routing-MTA, and records this as an indirect-exit-connection-group in the routing-MTA's local-use-tables (unless already recorded). The next-MTAs associated with each of these will be discovered in step *d*.

5) The current-routing-collective variable is assigned the value of the parent-routing-collective variable and the procedure continues at step *b-2*.

NOTE 1 – In this way, the procedure identifies the minimal set of routing-collectives that gives it knowledge of the complete routing-collective, i.e. its immediate peers, and those of each of its superior routing-collectives up to the base vertex of its routing-collective subtree.

c) For each of these key-routing-collectives, one or more next-MTAs are identified, i.e. MTAs reachable from this routing-MTA that provides a route to the key-routing-collective. This proceeds in stages. Firstly, the key-routing-collectives to which a message may be transferred directly are identified.

The following steps are taken for each key-routing-collective:

1) The key-routing-collective's entry-connection-group-names retrieved in step *b-3* are considered. The routing-MTA's exit-connection-group names are compared with the key-routing-collective's entry-connection-group names. If a common connection-group is found, and the key-routing-collective is a routing-MTA then a next-MTA has been found. The key-routing-collective's MHS Message Transfer Agent Name attribute (found in *b-3*) is recorded in the local-use-tables as a next-MTA for the key-routing-collective.

2) If a common connection-group is found, but the key-routing-collective is not a routing-MTA, the routing-MTA invokes a Directory Search (subtree) to discover a subordinate entry of the key-routing-collective with object class of Routing MTA, and with the common connection-group as an entry-connection-group. The MHS Message Transfer Agent Name attributes of any routing-MTAs discovered are retrieved and recorded in the local-use-tables as next-MTAs for the key-routing-collective.

NOTE 2 – Where multiple next-MTAs are found for a key-routing-collective, the routing-MTA may, as a local matter, retain all of these in its local-use-tables or retain only a subset.

3) The procedure continues at step *c-1* for the next key-routing-collective until all have been inspected.

d) Next-MTAs are now identified for any remaining key-routing-collectives. Since no direct connection exists for these key-routing-collectives, they are reachable only via an indirect-exit-connection-group. Next-MTAs are also identified for the indirect-exit-connection-groups identified in step *b-4*.

1) The routing-MTA considers each key-routing-collective for which a next-MTA has just been identified:

— For each transit-exit-connection-group available to the key-routing-collective that is an indirect-exit-connection-group of the routing-MTA, the next-MTAs associated with the key-routing-collective are recorded as the next-MTAs for the indirect-exit-connection-group.

2) For each key-routing-collective for which a next-MTA has not yet been identified, the following step is performed:

— Those indirect-exit-connection-groups of the routing-MTA for which a next-MTA has been identified are matched with the key-routing-collective's entry-connection-groups. For each match, the next-MTAs of the key-routing-collective are recorded by copying the next-MTAs of the indirect-exit-connection-group.

If a next-MTA has now been found for every key-routing-collective and indirect-exit-connection-group, the procedure resumes at step *e*.

If this step leaves the local-use-tables unchanged, then a configuration error is indicated and the procedure terminates.

3) The procedure continues at step *d-1*, except that only those key-routing-collectives for which a next-MTA has just been identified in step *d-2* are considered.

e) The routing-MTA considers each entry-connection-group (identified in step *a*), and acquires the authentication information necessary to identify possible calling MTAs of the connection-group.

For each entry-connection-group a Directory Read operation is invoked with the name of the connection-group as the *object* argument, and the following attributes as the *selection* argument:

— Enumerated Flag;

— Connection Type;

— Group MTA Password;

— Member MTA.

The Enumerated Flag is examined:

1) If the connection-group is of type enumerated, and its connection-type indicates that DN Used in A-ASSOCIATE has the value *false*, a Directory Read operation is invoked for each member MTA as the *object* argument, (a routing-MTA entry), and the MHS Message Transfer Agent Name attribute as the *selection* argument.

A further Directory Read is invoked with each MHS Message Transfer Agent Name as the *object* argument, and with the following attributes as the *selection* argument:

— MTA Name;

— MTA Password;

— Specific Passwords;

— Calling PSAPs;

— Presentation Address;

— Protocol Information.

The routing-MTA compiles a table containing the MHS Message Transfer Agent Name, MTA name, MTA Password, and calling presentation addresses for each member MTA of each of its (enumerated) entry-connection-groups.

NOTE 3 – As an implementation choice, the routing-MTA may also compile the same information for members of connection-groups where DN Used in A-ASSOCIATE is *true*. Alternatively, it may acquire authentication information for these MTAs dynamically, when MTA-bind-in is invoked.

If the member MTA's entry contains a Specific Passwords attribute, and the routing-MTA's routing-collective-name (or any of its superiors in the routing-collective-subtree) matches that present in a value of Specific Passwords, then the associated password takes precedence over the value present in the MTA Password attribute.

The presentation addresses are obtained from the Calling PSAPs attribute, if present. Otherwise they are taken from the Presentation Address attribute. In either case, any addresses whose network type is incompatible with that of the connection-group (as indicated in the profiles component of the Connection Information attribute) are eliminated.

NOTE 4 – In the simple case of symmetric addresses (where calling and called addresses are identical), only a single attribute is needed. In the asymmetric case, a second attribute is required.

2) If the connection-group is of type unenumerated, its Group MTA Password (if defined), is recorded in the local-use-tables.

9.1.6 MTA-bind-in procedure

This clause describes the behaviour of the routing-MTA when an MTA-bind is invoked by another MTA. This definition replaces the MTA-bind-in procedure definition in 14.9.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

9.1.6.1 Arguments

The MTA-bind arguments are defined in 12.1.1.1.1 and listed in Table 28 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

9.1.6.2 Results

The MTA-bind results are defined in 12.1.1.1.2 and listed in Table 29 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

9.1.6.3 Errors

The bind-errors are defined in 12.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

9.1.6.4 Procedure description

The MTA-bind-in procedure is described in the following steps:

a) If the routing-MTA's resources cannot currently support the establishment of a new association, the procedure returns a Busy bind-error and terminates.

b) The called MTA is presented with the following authentication information:

— an MTA-bind argument of Null, indicating no authentication, *or*,

— the calling MTA's MTA name and simple password, *or*,

— the calling MTA's MTA name, and strong authentication arguments (which include the called MTA's MTA name and, optionally, global domain identifier.

It may also be presented with additional connection information:

— the calling presentation address, possibly incomplete (for example, on an APS connection the network address may be absent);

— possibly, a security-context;

— possibly, a Directory name contained in the calling application-entity-title of the A-ASSOCIATE service (of the ACSE) that was used to invoke the MTA-bind;

— possibly, a Directory name from the certificate in the MTA-bind argument, if strong authentication is used.

The routing-MTA is required to verify the identity of the calling MTA (if authentication is in use), verify that the calling MTA is a member of one of its entry-connection-groups, verify that the rules of the entry-connection-group are observed (e.g., the security-context), and determine the correct MTS-bind response.

c) The routing-MTA considers each of its entry-connection-groups to determine which of them may have been selected by the calling MTA.

1) The routing-MTA first eliminates those entry-connection-groups whose Connection Type is incompatible with the type of connection made by the calling MTA (i.e. wrong application-context, wrong authentication-method, wrong network type, Directory name in A-ASSOCIATE required but not present).

2) Any remaining enumerated entry-connection-groups are considered. If the Directory name of the calling MTA is available, the routing-MTA attempts to match this name with those listed in the local-use-tables (in enumerated-authentication; see 9.1.5.2). Otherwise, it attempts to match both the MTA name and calling presentation address with those recorded in the local-use-tables and retrieves the Directory names associated with these table entries (i.e. the names of the MHS Message Transfer Agent entries from which the addressing information was originally obtained). Any enumerated entry-connect-groups which fail to match are eliminated.

3) Any remaining unenumerated entry-connection-groups are considered. If the simple-password authentication-method is in use, and the supplied credentials do not match the connection-group's Group MTA Password (if present), recorded in unenumerated-authentication, then that connection-group is eliminated and the next unenumerated entry-connection-group is considered. If strong-authentication is in use and no Certificate or Directory name is available, the connection-group is eliminated (unless local procedures allow the strong credentials to be authenticated in this

case). If an unenumerated entry-connection-group satisfies these tests, but no Directory name is available, then routing-MTA verifies that the security-context presented in the MTA-bind argument is compatible with that associated with the connection-group. If so, the procedure resumes at step *e*.

4) If all entry-connection-groups have been eliminated then the proposed association cannot be accepted. The routing-MTA returns an authentication-error and the procedure terminates.

Otherwise, one or more pairs of Directory name and connection-group have now been identified. The Directory name is derived either from A-ASSOCIATE, or by considering known adjacent MTAs in enumerated connection-groups and retrieving the Directory name from local-use tables.

d) Each pair of Directory name and connection-group is considered in turn. The routing-MTA verifies that the Directory entry corresponds to the calling MTA (much of the required information is already available in the local-use-tables, and some of the checks will have already been made). Alternatively, the routing-MTA may acquire this information dynamically by invoking a Directory Read with the MHS Message Transfer Name as the *object* argument.

1) The routing-MTA compares the calling presentation address of the connection with those defined for the MTA (in the Calling PSAPs attribute, if present; otherwise in the Presentation Address attribute), taking account of the network-address-reliable indicator associated with the connection-group to determine the required degree of match. If the match fails, the procedure resumes at step *d-4*.

2) If no-authentication is specified for this connection-group, or a simple-password is presented which matches the connection-group's Group MTA Password, then authentication requirements are satisfied. Otherwise, the credentials presented in the MTA-bind argument are validated according to the authentication-method in use.

If strong-authentication is in use, the signature of the initiator-bind-token is verified using the public key from the Certificate, if present; otherwise, from the appropriate value of the calling MTA's User Certificate attribute. The Directory name from the subject field of that Certificate is verified to be that of the calling MTA; the mta-name in the subject-alternate-name field of that Certificate is verified to correspond to the calling MTA's MTA Name and Global Domain Identifier attributes and to the mta-name present in the initiator-name field of MTA-bind; and the mta-name within the initiator-bind-token is verified as being the name of this routing-MTA.

If the calling MTA is not a routing-MTA its entry may contain a Specific Passwords attribute; if the routing-MTA's routing-collective-name (or any of its superiors in the routing-collective-subtree) matches that present in a value of Specific Passwords then the corresponding value of this-MTA-password is matched. If all matches fail, the procedure resumes at step *d-4*.

3) The routing-MTA verifies that the security-context presented in the MTA-bind argument is compatible with that associated with the connection-group. If so, the security-context is satisfied and the procedure resumes at step *e*.

4) If all pairs of Directory name and connection-group have been considered, the routing-MTA returns an authentication-error and the procedure terminates. Otherwise the next pair is selected and the procedure repeats step *d*.

NOTE 1 – If there is at least one case where the calling MTA satisfies all authentication requirements except for the security-context, then an unacceptable-security-context error is returned rather than authentication-error.

e) The routing-MTA establishes the connection and returns an MTA-bind result. If no-authentication was specified, the same value is returned in the MTA-bind result. For simple authentication, the routing-MTA's MTA Password is returned (if the calling MTA is not a routing-MTA then its MHS Message Transfer Agent may contain a value of Specific Passwords which takes precedence over the routing-MTA's MTA Password). If strong authentication is in use, the same signature algorithm is used to generate a responder-bind-token for the calling MTA.

f) If the calling MTA's MHS Message Transfer Agent entry is known, a Directory Read is invoked with MTA Name and Global Domain Identifier as the *selection* argument. These attributes are returned as results. The procedure then terminates.

NOTE 2 – The calling-MTA's retrieved MTA Name and Global Domain Identifier values may be used subsequently to verify the accuracy of the Trace-information and Internal-trace-information arguments of the Message-in, Probe-in, and Report-in procedures.

9.1.7 MTA-bind-out procedure

This clause describes the steps taken by a routing-MTA when it attempts to establish an association with another MTA. This definition replaces the MTA-bind-out procedure definition in 14.9.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

9.1.7.1 Arguments

A relay instruction generated by the Routing-decision procedure, which comprises:

a) The distinguished name of the MHS Message Transfer Agent with which an association is to be established.

b) The distinguished names of the one or more Connection Group entries that may be used to establish it.

c) Optionally, the security-context for the association.

9.1.7.2 Results

An internal identifier for the association established.

9.1.7.3 Errors

The procedure returns a failure indication in the event that an association could not be established.

9.1.7.4 Procedure description

The MTA-bind-out procedure is described in the following steps:

a) The following attributes are retrieved from the MHS Message Transfer Agent entry of the target MTA:

— MHS Maximum Message Length;

— Presentation Address;

— MTA Name;

— MTA Password;

— Specific Passwords;

— Global Domain Identifier;

— Protocol Information

— Supported Algorithms (see 6.16 of ITU-T Rec. X.521 | ISO/IEC 9594-7);

— User Certificate (see 6.17 of ITU-T Rec. X.521 | ISO/IEC 9594-7.

The following attributes are retrieved from the Connection Group entries:

— Connection Type:

— Group MTA Password;

— Security Context.

NOTE 1 – In practice, much of this information is likely to be already in hand in the local-use-tables.

b) The routing-MTA verifies that the message is no larger than the maximum message length defined for the target MTA. If the message is too large, a relay failure is returned and the procedure terminates.

NOTE 2 – If, subsequently, the Routing-decision procedure is unable to identify an alternative next-MTA destination for the recipient concerned, a report generation instruction is issued with the *non-delivery-reason-code* set to *unable-to-transfer*, and the *non-delivery-diagnostic-code* set to *content-too-long*.

c) The routing-MTA selects a suitable connection-group for the association, according to the specified security-context. The parameters for establishing the connection are determined by matching the information stored in the MHS Message Transfer Agent and Connection Group entries. The destination network address is determined by matching the profiles component of connection-information with the protocol information stored in the MTA's entry.

The initiator-credentials are set according to the authentication-method specified for the connection-group:

— if no-authentication, a Null;

— if simple-authentication, then MTA Password for an enumerated connection-group, or Group MTA Password (if present) for an unenumerated-connection-group, or calling-MTA-password if an applicable value of Specific Passwords is present;

— if strong-authentication, a signature algorithm is selected (from those identified in the target MTA's Supported Algorithms attribute which are supported by the routing-MTA, and for which a Certificate is present in the target MTA's User Certificate attribute) to generate an initiator-bind-token for the target MTA, and the appropriate Certificate of the routing-MTA for that algorithm is included.

NOTE 3 – The Certificate may only be omitted if the routing-MTA has only one Certificate for the identified signature algorithm, and the routing-MTA's Directory name is included in the calling name parameter of A-ASSOCIATE.

The routing-MTA attempts to establish the association using the information retrieved. If the attempt is unsuccessful, a failure indication is returned and the procedure terminates.

d) The result returned by the called MTA is examined. The responder-credentials must be of the same type (no authentication, simple, or strong) as that of the initiator credentials. If strong-authentication is in use, the following checks are made:

— the signature of the responder-bind-token is verified using the public key from the appropriate Certificate from the target MTA's User Certificate attribute;

— the Directory name from the subject field of that Certificate is verified to be that of the target MTA;

— the mta-name in the subject-alternate-name field of that Certificate is verified to correspond to the target MTA's MTA Name and Global Domain Identifier attributes and to the mta-name present in the responder-name field of Bind Result;

— the mta-name within the responder-bind-token is verified as being the name of this routing-MTA.

If the MTA-name and responder-credentials do not match those supplied in the MHS Message Transfer Agent entry then the procedure returns a relay failure to the caller, terminates the association, and terminates. Otherwise, the procedure returns an association identifier, and terminates.

NOTE 4 – If the two-way-alternate dialogue-mode is in use, the routing-MTA also returns the called-MTA's authenticated MTA-name and global-domain-identifier. These may be used subsequently for inward transfers over the same association to verify the accuracy of the Trace-information and Internal-trace-information arguments of the Message-in, Probe-in, and Report-in procedures.

9.1.8 Trace verification step

The **Trace verification step** is performed as an additional step in the existing Procedure Descriptions for the Message-in, Probe-in, and Report-in procedures defined in 14.10.1, 14.10.2, and 14.10.3, respectively, of ITU-T Rec. X.411 | ISO/IEC 10021-4. A routing-MTA shall perform this step when Message-in, Probe-in, or Report-in is invoked:

a) The routing-MTA compares the last element of Trace (and Internal-trace, if present), with the values of MTA name and global domain identifier obtained in the performance of the MTA-bind which caused the establishment of the present association. If matching fails, then as a local matter the routing-MTA may perform one of the following:

1) Instruct the RTSE to reject the transfer (if still in progress).

2) If currently performing the Message-in or Probe-in procedure, return a report generation instruction for all recipients for which *responsibility* is *responsible*. The *non-delivery-reason-code* is set to *unable-to-transfer*, and the *non-delivery-diagnostic-code* is set to *invalid-arguments*. The procedure then terminates.

3) Using the authenticated values of MTA-name and global-domain-identifier discovered in MTA-bind, construct additional elements of Trace and Internal-trace to correctly identify the MTA which transferred the message (or probe, or report) to this routing-MTA.

NOTE – In all cases, the trace verification failure should be reported to the MTA manager for investigation, as it is caused either by configuration error or by attempted masquerade.

b) If matching succeeds, or step *a-3* is followed, or the MTA name and global domain identifier were not discovered in the MTA-bind, or local policy permits transfer even where matching fails, then the Message-in, Probe-in, and Report-in procedures resume at the first step defined in their Procedure Descriptions.

9.2 Administrative procedures

An MHS administrator follows procedures described in 9.2.1 and 9.2.2 in order to configure an MTA for MHS-routing, and to construct an OR-address-subtree used to represent a portion of the OR-address name-space.

9.2.1 Routing-MTA configuration

The MHS administrator performs the follow steps to configure an MTA for MHS-routing:

a) A Directory entry of object class MHS Message Transfer Agent is created (see A.1.3. of ITU-T Rec. X.402 | ISO/IEC 10021-2). The following attributes associated with the MTA Information auxiliary object class are assigned:

— MTA Name;

— Global Domain Identifier;

— MTA Credentials.

b) A Directory entry of object class Routing MTA is created in the routing-collective-subtree for this routing-MTA and assigned the following attributes:

— OR-address Subtrees;

— MHS Message Transfer Agent Name.

c) The following attributes associated with the object's direct superclass, Routing Collective, are assigned:

— Routing Collective Name;

— Entry Connection Group Name;

— Description;

— Transit Exit Connection Group Name;

— Local Exit Connection Group Name.

d) If suitable entries do not already exist, an entry is created for each connection-group referenced. It is verified that the routing-MTA has been granted access permissions so that all connection-group entries can be read.

e) If any of the connection-groups referenced is of type enumerated, its Member MTA attribute is updated to indicate the addition of this routing-MTA to the connection-group.

e) Depending on policy, any of the connection-groups referenced above that are not represented in those associated with the routing-collective's immediate superior, may be added to the superior's entry-connection-groups, and transit- or local-exit-connection-groups. This procedure may be repeated recursively up to the base of the routing-collective-subtree.

f) If required, a new OR-address-subtree is created for the OR-addresses for which the routing-MTA is the delivering MTA. Alternatively, the OR-addresses are added to an existing OR-address-subtree. The procedure defined in 9.2.2 is followed (omitting step *a* of that procedure if an existing subtree is used).

9.2.2 OR-address-subtree construction

To form an OR-address-subtree that represents a collection of OR-addresses, the MHS administrator performs the following steps:

a) The administrator creates an arbitrary entry in a part of the Directory to act as the base vertex of the OR-address subtree. Access-controls may be applied to this entry such that it is accessible only to those MTAs that will be configured to use the OR-address-subtree.

b) Each OR-address in the collection is transformed into a Directory name as follows:

For each element of OR-address relevant to routing, there is a corresponding Directory object class definition (see 8.3). The value of the element is transformed into a value of the attribute defined for the element (a subtype of the OR-address-element-name attribute type). This is treated as the RDN of an entry whose immediate superior corresponds either to the base vertex (in the case of MHS Country) or to the vertex derived from the preceding element of the OR-address. Hence, a series of entries is created, one for each relevant element present in the OR-address. The Directory name of the final entry represents a complete mapping of the OR-address (see 9.1.3.4).

c) A set of OR-address aliases may be presented. These consist of OR-address pairs, the first corresponding to a preferred name, and the second to an alternative name. An alias entry is created for the second name, which references the first name.

NOTE – A Directory alias is not intended to cause an alias Redirection, but is used to specify an alternative OR-address which is to be routed equivalently.

d) Message originators may use different combinations of Personal Name OR-address elements at different times to identify personal recipients. To deal with these vagaries, a range of alias requirements exists for the following OR-address elements: MHS Common Name, MHS Surname, MHS Given Name, MHS Initials, MHS Generation Qualifier. To accommodate these requirements, the MHS administrator of the local OR-

address name-space may configure the subtree such that entries exist for every combination of Personal Name elements. This may be accomplished as follows: each Personal Name branch of the tree is examined, and the highest level entry that unambiguously denotes the user remains an object entry. All subordinate entries become alias entries that refer to this object entry. Some examples of modelling Personal Name OR-address elements are given in Figure 4.

e) A set of additional OR-address element attributes is presented. Each presentation has three parts:

1) the name of the entry in which the attribute is to be placed;

2) the attribute type: Routing Advice, Expression Matches, or Next Level Complete;

3) the attribute-value.

Each attribute is created in the specified entry.

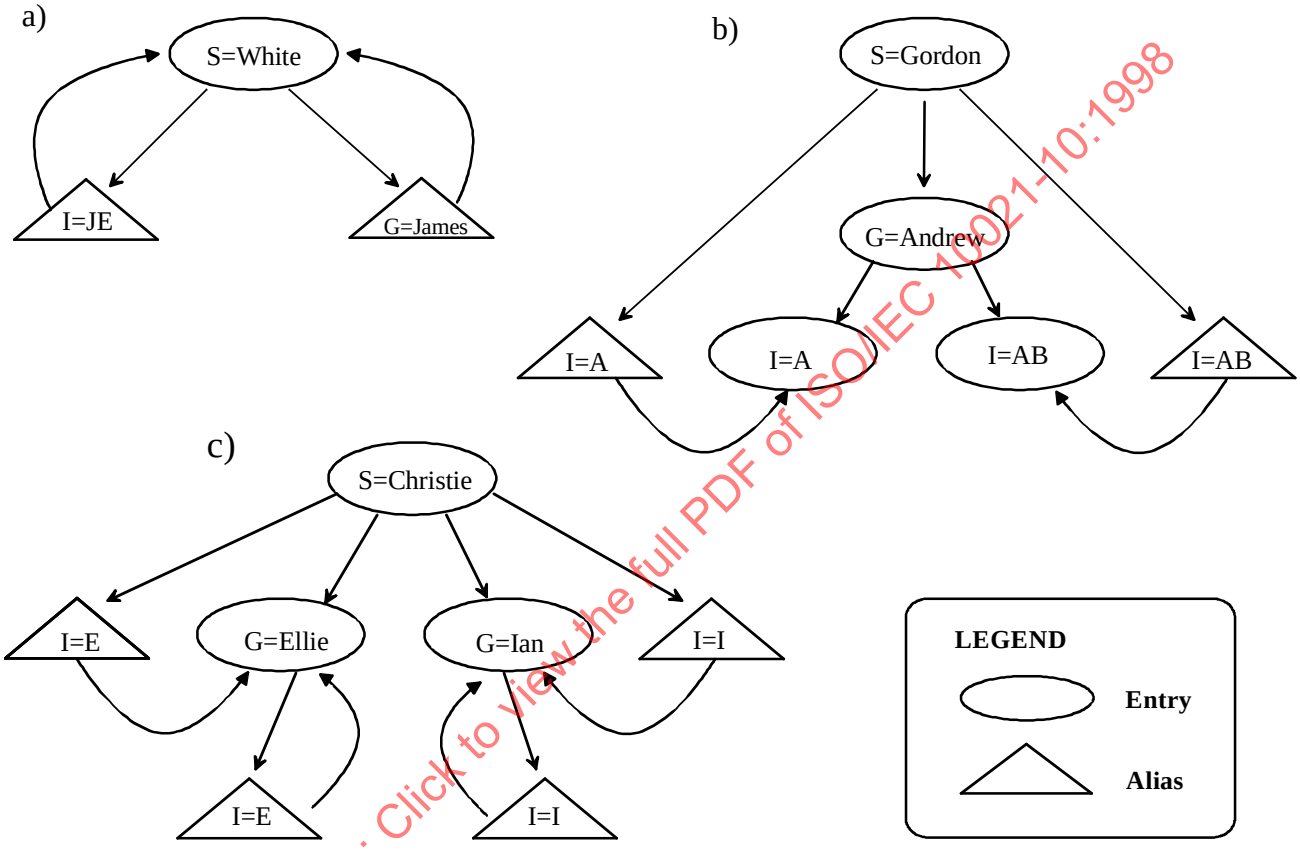

Figure 4 - Modelling examples for Personal Name elements

NOTES

1 Example *a* illustrates the case where a particular Surname is unique in this OR-address-subtree. Alias entries are created corresponding to the optional Initials and Given Name elements.

2 In example *b*, entries exist for two users who share the same Surname and Given Name. The users are distinguished by their differing Initials.

3 In example *c*, entries exist for two users who share the same surname. The users are distinguished either by their differing Given Names or by their differing Initials (or both).

10 Conformance

The requirements an MTA, a DUA, and a DSA shall satisfy when a claim of conformance to this part of ISO/IEC 10021 is made are identified in 10.1 - 10.3.

10.1 Routing-MTA conformance

A routing-MTA implementation shall satisfy the following static requirements:

a) Given its routing-collective Directory name, a routing-MTA shall be capable of acquiring from the Directory all the information necessary to perform MHS-routing.

b) A DUA incorporated with a routing-MTA shall be capable of supporting the object classes, attribute types, and matching-rules defined for MHS-routing.

A routing-MTA implementation shall satisfy the following dynamic requirement:

c) A routing-MTA shall conform to the Front-end procedure defined in 14.3.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4, as modified by 9.1.1 of this part of ISO/IEC 10021.

d) A routing-MTA shall conform to the Routing-decision procedure defined in 9.1.2 of this part of ISO/IEC 10021, which replaces the definition in 14.3.4 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

e) A routing-MTA shall conform to the OR-address-subtree-read and Local-delivery-evaluation procedures defined in 9.1.3 and 9.1.4, respectively, of this part of ISO/IEC 10021.

f) A routing-MTA shall conform to the MTA-bind-in procedure defined in 9.1.6 of this part of ISO/IEC 10021, which replaces the definition in 14.9.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

g) A routing-MTA shall conform to the MTA-bind-out procedure defined in 9.1.7 of this part of ISO/IEC 10021, which replaces the definition in 14.9.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

h) A routing-MTA shall conform to the Trace verification step defined in 9.1.8 of this part of ISO/IEC 10021, which supplements the Message-in, Probe-in, and Report-in procedures defined in 14.10.1, 14.10.2, and 14.10.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

10.2 Administrative DUA conformance

A DUA implementation employed to maintain the Directory information used for MHS-routing shall satisfy the following static requirement:

— An administrative DUA shall be capable of supporting the object classes, attribute types, and matching-rules defined for MHS-routing, and shall apply consistency checks to ensure the internal consistency of MHS-routing Directory information.

10.3 DSA conformance

A DSA implementation employed to hold information used for MHS-routing shall satisfy the following static requirement:

— A DSA shall be capable of supporting the object classes, attribute types, and matching-rules defined for MHS-routing.

Annex A
(normative)
Reference Definition of Object Identifiers

This annex defines for reference purposes various Object Identifiers cited in the ASN.1 modules of subsequent annexes. All Object Identifiers assigned in this part of ISO/IEC 10021 are assigned in this annex.

```
MHSRoutingObjectIdentifiers {joint-iso-itu-t mhs(6) routing(10) modules(0) object-identifiers(0)}
DEFINITIONS ::=

BEGIN

-- Prologue

-- Exports everything

IMPORTS -- nothing -- ;

ID ::= OBJECT IDENTIFIER

-- MHS-routing

id-mhs-routing              ID ::= {joint-iso-itu-t mhs(6) routing(10)} -- not definitive

-- Categories

id-mod                                  ID ::= {id-mhs-routing 0} -- modules; not definitive
id-oc                                   ID ::= {id-mhs-routing 1} -- object-classes
id-nf                                   ID ::= {id-mhs-routing 2} -- name-forms
id-at                                   ID ::= {id-mhs-routing 3} -- attributes
id-pro                                  ID ::= {id-mhs-routing 4} -- profiles
-- Modules

id-mod-object-identifiers               ID ::= {id-mod 0} -- not definitive
id-mod-directory-objects                ID ::= {id-mod 1} -- not definitive
id-mod-oraddress-subtree                ID ::= {id-mod 2} -- not definitive

-- Object classes

id-oc-connection-group                  ID ::= {id-oc 0}
id-oc-mhs-admd                          ID ::= {id-oc 1}
id-oc-mhs-common-name                   ID ::= {id-oc 2}
id-oc-mhs-country                       ID ::= {id-oc 3}
id-oc-mhs-extended-network-address      ID ::= {id-oc 4}
id-oc-mhs-generation-qualifier          ID ::= {id-oc 5}
id-oc-mhs-given-name                    ID ::= {id-oc 6}
id-oc-mhs-initials                      ID ::= {id-oc 7}
id-oc-mhs-network-address               ID ::= {id-oc 8}
id-oc-mhs-numeric-user-identifier       ID ::= {id-oc 9}
id-oc-mhs-or-address-element            ID ::= {id-oc 10}
id-oc-mhs-organization                  ID ::= {id-oc 11}
id-oc-mhs-organizational-unit           ID ::= {id-oc 12}
id-oc-mhs-pds-name                      ID ::= {id-oc 13}
id-oc-mhs-physical-delivery-country-name ID ::= {id-oc 14}
id-oc-mhs-postal-code                   ID ::= {id-oc 15}
id-oc-mhs-prmd                          ID ::= {id-oc 16}
id-oc-mhs-surname                       ID ::= {id-oc 17}
id-oc-mhs-terminal-identifier           ID ::= {id-oc 18}
id-oc-mhs-terminal-type                 ID ::= {id-oc 19}
id-oc-mta-information                   ID ::= {id-oc 20}
id-oc-oraddress-subtree-base            ID ::= {id-oc 21}
id-oc-routing-collective                ID ::= {id-oc 22}
id-oc-routing-mta                       ID ::= {id-oc 23}

-- Name forms

id-nf-connection-group                  ID ::= {id-nf 0}
id-nf-mhs-admd                          ID ::= {id-nf 1}
id-nf-mhs-common-name                   ID ::= {id-nf 2}
id-nf-mhs-country                       ID ::= {id-nf 3}
id-nf-mhs-extended-network-address      ID ::= {id-nf 4}
id-nf-mhs-generation-qualifier          ID ::= {id-nf 5}
id-nf-mhs-given-name                    ID ::= {id-nf 6}
id-nf-mhs-initials                      ID ::= {id-nf 7}
id-nf-mhs-network-address               ID ::= {id-nf 8}
id-nf-mhs-numeric-user-identifier       ID ::= {id-nf 9}
id-nf-mhs-organization                  ID ::= {id-nf 10}
id-nf-mhs-organizational-unit           ID ::= {id-nf 11}
```

```
id-nf-mhs-pds-name                          ID ::= {id-nf 12}
id-nf-mhs-physical-delivery-country         ID ::= {id-nf 13}
id-nf-mhs-postal-code                       ID ::= {id-nf 14}
id-nf-mhs-prmd                              ID ::= {id-nf 15}
id-nf-mhs-surname                           ID ::= {id-nf 16}
id-nf-mhs-terminal-identifier               ID ::= {id-nf 17}
id-nf-mhs-terminal-type                     ID ::= {id-nf 18}
id-nf-routing-collective                    ID ::= {id-nf 19}

-- Attributes

id-at-calling-psaps                         ID ::= {id-at 0}
id-at-connection-group-name                 ID ::= {id-at 1}
id-at-connection-type                       ID ::= {id-at 2}
id-at-entry-connection-group-name           ID ::= {id-at 3}
id-at-enumerated-flag                       ID ::= {id-at 4}
id-at-global-domain-identifier              ID ::= {id-at 5}
id-at-group-mta-password                    ID ::= {id-at 6}
id-at-local-exit-connection-group-name      ID ::= {id-at 7}
id-at-member-mta                            ID ::= {id-at 8}
id-at-mhs-admd-name                         ID ::= {id-at 9}
id-at-mhs-common-name                       ID ::= {id-at 10}
id-at-mhs-country-name                      ID ::= {id-at 11}
id-at-mhs-expression-matches                ID ::= {id-at 12}
id-at-mhs-extended-network-address          ID ::= {id-at 13}
id-at-mhs-generation-qualifier              ID ::= {id-at 14}
id-at-mhs-given-name                        ID ::= {id-at 15}
id-at-mhs-initials                          ID ::= {id-at 16}
id-at-mhs-message-transfer-agent            ID ::= {id-at 17}
id-at-mhs-network-address                   ID ::= {id-at 18}
id-at-mhs-next-level-complete               ID ::= {id-at 19}
id-at-mhs-numeric-user-identifier           ID ::= {id-at 20}
id-at-mhs-organization-name                 ID ::= {id-at 21}
id-at-mhs-organizational-unit-name          ID ::= {id-at 22}
id-at-mhs-pds-name-attribute                ID ::= {id-at 23}
id-at-mhs-postal-code                       ID ::= {id-at 24}
id-at-mhs-prmd-name                         ID ::= {id-at 25}
id-at-mhs-routing-advice                    ID ::= {id-at 26}
id-at-mhs-surname                           ID ::= {id-at 27}
id-at-mhs-terminal-identifier               ID ::= {id-at 28}
id-at-mhs-terminal-type                     ID ::= {id-at 29}
id-at-mta-name                              ID ::= {id-at 30}
id-at-mta-password                          ID ::= {id-at 31}
id-at-oraddress-element-name                ID ::= {id-at 32}
id-at-oraddress-subtrees                    ID ::= {id-at 33}
id-at-recipient-md-assigned-alternate-recipient ID ::= {id-at 34}
id-at-routing-collective-name               ID ::= {id-at 35}
id-at-security-context                      ID ::= {id-at 36}
id-at-specific-passwords                    ID ::= {id-at 37}
id-at-transit-exit-connection-group-name    ID ::= {id-at 38}

-- Profiles

id-pro-x25                                  ID ::= {id-pro 0}
id-pro-rfc1006                              ID ::= {id-pro 1}
id-pro-x445                                 ID ::= {id-pro 2}

END -- of MHS-Routing Object Identifiers --
```

Annex B
(normative)
Reference Definition of MHS-routing Directory Objects

This annex defines for reference purposes the object classes, attributes, and name forms specific to MHS-routing. It uses the OBJECT-CLASS, ATTRIBUTE, and NAME-FORM information object classes of ITU-T Rec. X.501 | ISO/IEC 9594-2.

```
MHSRoutingDirectoryObjects {joint-iso-itu-t mhs(6) routing(10) modules(0) directory-objects(1)}
DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- Prologue

-- Exports everything

IMPORTS

-- Information framework

      ATTRIBUTE, DistinguishedName, NAME-FORM, OBJECT-CLASS, top
            ----
            FROM InformationFramework {joint-iso-itu-t ds(5) module(1) informationFramework(1) 3}

-- Selected attribute types

      commonName, description, distinguishedName, PresentationAddress
            ----
            FROM SelectedAttributeTypes {joint-iso-itu-t ds(5) module(1) selectedAttributeTypes(5) 2}

-- MTS abstract service

      GlobalDomainIdentifier, InitiatorCredentials, MTAName, ORName, Password, ResponderCredentials,
      SecurityContext
            ----
            FROM MTSAbstractService {joint-iso-itu-t mhs(6) mts(3) modules(0) mts-abstract-service(1)
                                     version-1994(0)}

-- MHS Directory objects and attributes

      mhs-message-transfer-agent
            ----
            FROM MHSDirectoryObjectsAndAttributes {joint-iso-itu-t mhs(6) arch(5) modules(0)
                                                   directory(1) version-1997(1)}

-- MHS protocol object identifier

      id-ac-mts-transfer
            ----
            FROM MHSProtocolObjectIdentifiers {joint-iso-itu-t mhs(6) protocols(0) modules(0)
                                               object-identifiers(0) version-1994(0)}

-- MHS-routing object identifiers

      id-at-calling-psaps, id-at-connection-group-name, id-at-connection-type,
      id-at-entry-connection-group-name, id-at-enumerated-flag, id-at-global-domain-identifier,
      id-at-group-mta-password, id-at-local-exit-connection-group-name, id-at-member-mta,
      id-at-mhs-message-transfer-agent, id-at-mta-name, id-at-mta-password, id-at-oraddress-subtrees,
      id-at-recipient-md-assigned-alternate-recipient, id-at-routing-collective-name,
      id-at-security-context, id-at-specific-passwords, id-at-transit-exit-connection-group-name,
      id-nf-connection-group, id-nf-routing-collective, id-oc-connection-group, id-oc-mta-information,
      id-oc-routing-collective, id-oc-routing-mta
            ----
            FROM MHSRoutingObjectIdentifiers {joint-iso-itu-t mhs(6) routing(10) modules(0)
                                              object-identifiers (0)};

-- OBJECT-CLASSES

routingCollective      OBJECT-CLASS ::= {
      SUBCLASS OF      {top}
      MUST CONTAIN     {routingCollectiveName}
      MAY CONTAIN      {description | entryConnectionGroupName | localExitConnectionGroupName |
                        transitExitConnectionGroupName}
                        --at least one entry-CG and exit-CG should be present--
      ID               id-oc-routing-collective }
```

```
routingMTA              OBJECT-CLASS ::= {
      SUBCLASS OF       {routingCollective}
      MUST CONTAIN      {oRAddressSubtrees | mHSMessageTransferAgentName}
      ID                id-oc-routing-mta }

connectionGroup         OBJECT-CLASS ::= {
      SUBCLASS OF       {top}
      MUST CONTAIN      {commonName | enumeratedFlag}
      MAY CONTAIN       {description | connectionType | groupMTAPassword | memberMTA | securityContext}
      ID                id-oc-connection-group }

mTAInformation          OBJECT-CLASS ::= {
      KIND              auxiliary
      MUST CONTAIN      {mTAName | globalDomainIdentifier }
      MAY CONTAIN       {mTAPassword | specificPasswords | callingPSAPs}
      ID                id-oc-mta-information }

-- ATTRIBUTE-TYPES

-- Routing-collective attribute types

routingCollectiveName   ATTRIBUTE ::= {
      SUBTYPE OF        commonName -- see ITU-T X.520 | ISO/IEC 9594-6 --
      SINGLE VALUE      TRUE
      ID                id-at-routing-collective-name }

connectionGroupName     ATTRIBUTE ::= {
      WITH SYNTAX       DistinguishedName
      SINGLE VALUE      FALSE
      ID                id-at-connection-group-name }

entryConnectionGroupName ATTRIBUTE ::= {
      SUBTYPE OF        connectionGroupName
      ID                id-at-entry-connection-group-name }

transitExitConnectionGroupName ATTRIBUTE ::= {
      SUBTYPE OF        connectionGroupName
      ID                id-at-transit-exit-connection-group-name }

localExitConnectionGroupName ATTRIBUTE ::= {
      SUBTYPE OF        connectionGroupName
      ID                id-at-local-exit-connection-group-name }

-- Routing-MTA attribute types

oRAddressSubtrees       ATTRIBUTE ::= {
      WITH SYNTAX       ORAddressSubtreeNames
      SINGLE VALUE      TRUE
      ID                id-at-oraddress-subtrees }

ORAddressSubtreeNames ::= SEQUENCE OF DistinguishedName

mHSMessageTransferAgentName ATTRIBUTE ::= {
      SUBTYPE OF        distinguishedName
      SINGLE VALUE      TRUE
      ID                id-at-mhs-message-transfer-agent }

-- Connection-group attribute types

enumeratedFlag          ATTRIBUTE ::= {
      WITH SYNTAX       BOOLEAN -- True=enumerated, False=unenumerated --
      SINGLE VALUE      TRUE
      ID                id-at-enumerated-flag }

connectionType          ATTRIBUTE ::= {
      WITH SYNTAX       ConnectionInformation
      SINGLE VALUE      TRUE
      ID                id-at-connection-type }

ConnectionInformation ::= SET {
      application-context       [0] OBJECT IDENTIFIER DEFAULT id-ac-mts-transfer,
      profiles                  [1] SET OF OBJECT IDENTIFIER OPTIONAL,
      dn-used-in-a-associate    [2] BOOLEAN DEFAULT TRUE,
      network-address-reliable  [3] BOOLEAN DEFAULT TRUE,
      authentication-method     [4] AuthenticationMethod DEFAULT simple-password}

AuthenticationMethod ::= INTEGER {
      no-authentication         (0),
      simple-password           (1),
      strong-authentication     (2) }

groupMTAPassword        ATTRIBUTE ::= {
      WITH SYNTAX       Password
      SINGLE VALUE      TRUE
      ID                id-at-group-mta-password }
```

```
memberMTA               ATTRIBUTE ::= {
      WITH SYNTAX       RoutingMTAName
      SINGLE VALUE      FALSE
      ID                id-at-member-mta }

RoutingMTAName ::= RoutingCollectiveName

RoutingCollectiveName ::= DistinguishedName

securityContext         ATTRIBUTE ::= {
      WITH SYNTAX       SecurityContext
      SINGLE VALUE      TRUE
      ID                id-at-security-context }

-- MTA-information attribute types

mTAName                 ATTRIBUTE ::= {
      WITH SYNTAX       MTAName
      SINGLE VALUE      TRUE
      ID                id-at-mta-name }

globalDomainIdentifier  ATTRIBUTE ::= {
      WITH SYNTAX       GlobalDomainIdentifier
      SINGLE VALUE      TRUE
      ID                id-at-global-domain-identifier }

mTAPassword             ATTRIBUTE ::= {
      WITH SYNTAX       Password
      SINGLE VALUE      TRUE
      ID                id-at-mta-password }

specificPasswords ATTRIBUTE ::= {
      WITH SYNTAX       SpecificPassword
      SINGLE VALUE      FALSE
      ID                id-at-specific-passwords }

SpecificPassword ::= SET {
      routing-collective-name RoutingCollectiveName,
      this-mta-password       [0] Password,
      calling-mta-password    [1] Password }

callingPSAPs ATTRIBUTE ::= {
      WITH SYNTAX       PresentationAddress
      SINGLE VALUE      FALSE
      ID                id-at-calling-psaps }

-- NAME-FORMS

routingCollectiveNameForm NAME-FORM ::= {
      NAMES             routingCollective
      WITH ATTRIBUTES   {routingCollectiveName}
      ID                id-nf-routing-collective }

connectionGroupNameForm NAME-FORM ::= {
      NAMES             connectionGroup
      WITH ATTRIBUTES   {commonName}
      ID                id-nf-connection-group}

END -- of MHS-routing Directory Objects
```

Annex C
(normative)
Reference Definition of MHS-routing OR-address-subtree

This annex defines for reference purposes the object classes, attributes, name forms, and matching rules that define the OR-address-subtree. It uses the OBJECT-CLASS, ATTRIBUTE, NAME-FORM and MATCHING-RULE information object classes of ITU-T Rec. X.501 | ISO/IEC 9594-2.

```
MHSRoutingORAddressSubtree {joint-iso-itu-t mhs(6) routing(10) modules(0) oraddress-subtree(2)}
DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- Prologue

-- Exports everything

IMPORTS

-- MHS-routing Directory objects

     RoutingCollectiveName
           ----
           FROM MHSRoutingDirectoryObjects {joint-iso-itu-t mhs(6) routing(10) modules(0)
                                            directory-objects(1)}

-- MTS abstract service

     NonDeliveryDiagnosticCode, NonDeliveryReasonCode, ORAddress, ORName, RecipientName,
     SupplementaryInformation, UniversalOrBMPString{}
           ----
           FROM MTSAbstractService {joint-iso-itu-t mhs(6) mts(3) modules(0) mts-abstract-service(1)
                                    version-1994(0)}

-- MTS upper bounds

     ub-common-name-length, ub-country-name-numeric-length, ub-domain-defined-attribute-type-length,
     ub-domain-defined-attribute-value-length, ub-domain-name-length, ub-generation-qualifier-length,
     ub-given-name-length, ub-initials-length, ub-numeric-user-id-length,
     ub-organization-name-length, ub-organizational-unit-name-length, ub-pds-name-length,
     ub-postal-code-length, ub-surname-length, ub-terminal-id-length, ub-x121-address-length
           ----
           FROM MTSUpperBounds {joint-iso-itu-t mhs(6) mts(3) modules(0) upper-bounds(3)}

-- MHS-routing object identifiers

     id-at-mhs-admd-name, id-at-mhs-common-name, id-at-mhs-country-name,
     id-at-mhs-expression-matches, id-at-mhs-extended-network-address,
     id-at-mhs-generation-qualifier, id-at-mhs-given-name, id-at-mhs-initials,
     id-at-mhs-network-address, id-at-mhs-numeric-user-identifier, id-at-mhs-organization-name,
     id-at-mhs-organizational-unit-name, id-at-mhs-pds-name-attribute, id-at-mhs-postal-code,
     id-at-mhs-prmd-name, id-at-mhs-routing-advice, id-at-mhs-next-level-complete, id-at-mhs-surname,
     id-at-mhs-terminal-identifier, id-at-mhs-terminal-type, id-at-oraddress-element-name,
     id-at-recipient-md-assigned-alternate-recipient, id-nf-mhs-admd, id-nf-mhs-common-name,
     id-nf-mhs-country, id-nf-mhs-extended-network-address, id-nf-mhs-generation-qualifier,
     id-nf-mhs-given-name, id-nf-mhs-initials, id-nf-mhs-network-address,
     id-nf-mhs-numeric-user-identifier, id-nf-mhs-organization, id-nf-mhs-organizational-unit,
     id-nf-mhs-pds-name, id-nf-mhs-physical-delivery-country, id-nf-mhs-postal-code, id-nf-mhs-prmd,
     id-nf-mhs-surname, id-nf-mhs-terminal-identifier, id-nf-mhs-terminal-type, id-oc-mhs-admd,
     id-oc-mhs-common-name, id-oc-mhs-country, id-oc-mhs-extended-network-address,
     id-oc-mhs-generation-qualifier, id-oc-mhs-given-name, id-oc-mhs-initials,
     id-oc-mhs-network-address, id-oc-mhs-numeric-user-identifier, id-oc-mhs-or-address-element,
     id-oc-mhs-organization, id-oc-mhs-organizational-unit, id-oc-mhs-pds-name,
     id-oc-mhs-physical-delivery-country-name, id-oc-mhs-postal-code, id-oc-mhs-prmd,
     id-oc-mhs-surname, id-oc-mhs-terminal-identifier, id-oc-mhs-terminal-type,
     id-oc-oraddress-subtree-base
     ----
           FROM MHSRoutingObjectIdentifiers {joint-iso-itu-t mhs(6) routing(10) modules(0)
                                             object-identifiers(0)}

-- Information framework

     ATTRIBUTE, DistinguishedName, MATCHING-RULE, NAME-FORM, OBJECT-CLASS, top
           ----
           FROM InformationFramework {joint-iso-itu-t ds(5) module(1) informationFramework(1) 3}
```

```
-- Directory authentication framework

      AlgorithmIdentifier
            ----
            FROM AuthenticationFramework {joint-iso-itu-t ds(5) module(1) auhenticationFramework(7) 3}

-- Directory certificate extensions

      CertificateAssertion
            ----
            FROM CertificateExtensions {joint-iso-itu-t ds(5) module(1) certificateExtensions(26) 0}

-- Selected attribute types

      commonName, DirectoryString{}, name
            ----
            FROM SelectedAttributeTypes {joint-iso-itu-t ds(5) module(1) selectedAttributeTypes(5) 2};


-- OR-ADDRESS-SUBTREE


-- OR-address element

oRAddressElement      OBJECT-CLASS ::= {
      SUBCLASS OF     {top}
      KIND            abstract
      MAY CONTAIN     {routingAdvice | expressionMatches | nextLevelComplete |
                      recipientMDAssignedAlternateRecipient }
      ID              id-oc-mhs-or-address-element }


--

routingAdvice         ATTRIBUTE ::= {
      WITH SYNTAX     RoutingAdvice
      SINGLE VALUE    TRUE
      ID              id-at-mhs-routing-advice }

RoutingAdvice ::= CHOICE {
      target-routing-collective   [0] TargetRoutingCollective,
      non-delivery-information    [1] NonDeliveryInformation,
      alias-redirection           [2] AliasRedirection,
      dl-expansion-information    [3] DLExpansionInformation,
      double-envelope-information [4] DoubleEnvelopeInformation,
      ...   }


TargetRoutingCollective ::= SEQUENCE {
      target-routing-collective   [0] RoutingCollectiveName,
      local-user-identifier       [1] UniversalOrBMPString {ub-local-user-identifier} OPTIONAL }

ub-local-user-identifier INTEGER ::= 128

NonDeliveryInformation ::= SEQUENCE {
      reason                      [0] NonDeliveryReasonCode,
      diagnostic                  [1] NonDeliveryDiagnosticCode OPTIONAL,
      supplementary-information   [2] SupplementaryInformation OPTIONAL }

AliasRedirection ::= SEQUENCE {
      redirection-address         [0] ORAddress,
      edit                        [1] BOOLEAN DEFAULT TRUE }

DLExpansionInformation ::= SEQUENCE {
      dl-expansion-routing-collectives [0] SET OF TargetRoutingCollective,
      dl-name                     [1] MHSDistributionListName OPTIONAL,
      any-mta-may-expand          [2] BOOLEAN DEFAULT FALSE }

MHSDistributionListName ::= DistinguishedName

DoubleEnvelopeInformation ::= SEQUENCE {
      envelope-opener             [0] ORAddressAndDirectoryName,
      content-confidentiality-algorithm-preference [1] SEQUENCE OF AlgorithmInformation,
      key-encryption-algorithm-preference  [2] SEQUENCE OF AlgorithmInformation OPTIONAL,
      message-origin-algorithm-preference  [3] SEQUENCE OF AlgorithmInformation OPTIONAL,
      token-signature-algorithm-preference [4] SEQUENCE OF AlgorithmInformation OPTIONAL,
      ... }

ORAddressAndDirectoryName ::= ORName -- with both Directory name and OR-address present --

AlgorithmInformation ::= SEQUENCE {
      algorithm-identifier             [0] AlgorithmIdentifier,
      originator-certificate-selector  [1] CertificateAssertion OPTIONAL,
      recipient-certificate-selector   [2] CertificateAssertion OPTIONAL }
```

```
--

expressionMatches ATTRIBUTE ::= {
      WITH SYNTAX      ExpressionMatches
      SINGLE VALUE     TRUE
      ID               id-at-mhs-expression-matches }

ExpressionMatches ::= SEQUENCE OF ExpressionMatch

ExpressionMatch ::= SEQUENCE {
      filter-set       SET OF ORAddressFilter,
      routing-advice   RoutingAdvice }

ORAddressFilter ::= SEQUENCE {
      attribute-type   CHOICE {
            standard-attribute        INTEGER,
            domain-defined-attribute  UniversalOrBMPString {
                              ub-domain-defined-attribute-type-length} },
      pattern          ExtendedRegularExpression }

ExtendedRegularExpression ::= UniversalOrBMPString {ub-extended-regular-expression}

ub-extended-regular-expression INTEGER ::= 1024


--

nextLevelComplete      ATTRIBUTE ::= {
      WITH SYNTAX      NULL
      SINGLE VALUE     TRUE
      ID               id-at-mhs-next-level-complete }

recipientMDAssignedAlternateRecipient ATTRIBUTE ::= {
      WITH SYNTAX      ORName
      SINGLE VALUE     FALSE
      COLLECTIVE       TRUE
      ID               id-at-recipient-md-assigned-alternate-recipient }


-- OR-address element subclasses

oRAddressSubtreeBase   OBJECT-CLASS ::= {
      SUBCLASS OF      {oRAddressElement}
      KIND             structural
      MUST CONTAIN     {commonName}
      ID               id-oc-oraddress-subtree-base }


-- Common OR-address object classes

mHSCountry             OBJECT-CLASS ::= {
      SUBCLASS OF      {oRAddressElement}
      KIND             structural
      MUST CONTAIN     {mHSCountryName}
      ID               id-oc-mhs-country }

mHSADMD                OBJECT-CLASS ::= {
      SUBCLASS OF      {oRAddressElement}
      KIND             structural
      MUST CONTAIN     {mHSADMDName}
      ID               id-oc-mhs-admd }

mHSPRMD                OBJECT-CLASS ::= {
      SUBCLASS OF      {oRAddressElement}
      KIND             structural
      MUST CONTAIN     {mHSPRMDName}
      ID               id-oc-mhs-prmd }


-- Mnemonic OR-address object classes

mHSOrganization        OBJECT-CLASS ::= {
      SUBCLASS OF      {oRAddressElement}
      KIND             structural
      MUST CONTAIN     {mHSOrganizationName}
      ID               id-oc-mhs-organization }

mHSOrganizationalUnit  OBJECT-CLASS ::= {
      SUBCLASS OF      {oRAddressElement}
      KIND             structural
      MUST CONTAIN     {mHSOrganizationalUnitName}
      ID               id-oc-mhs-organizational-unit }

mHSCommonName          OBJECT-CLASS ::= {
      SUBCLASS OF      {oRAddressElement}
      KIND             structural
      MUST CONTAIN     {mHSCommonNameAttribute}
      ID               id-oc-mhs-common-name }
```

```
mHSSurname              OBJECT-CLASS ::= {
      SUBCLASS OF       {oRAddressElement}
      KIND              structural
      MUST CONTAIN      {mHSSurnameAttribute}
      ID                id-oc-mhs-surname }

mHSGivenName            OBJECT-CLASS ::= {
      SUBCLASS OF       {oRAddressElement}
      KIND              structural
      MUST CONTAIN      {mHSGivenNameAttribute}
      ID                id-oc-mhs-given-name }

mHSInitials             OBJECT-CLASS ::= {
      SUBCLASS OF       {oRAddressElement}
      KIND              structural
      MUST CONTAIN      {mHSInitialsAttribute}
      ID                id-oc-mhs-initials }

mHSGenerationQualifier OBJECT-CLASS ::= {
      SUBCLASS OF       {oRAddressElement}
      KIND              structural
      MUST CONTAIN      {mHSGenerationQualifierAttribute}
      ID                id-oc-mhs-generation-qualifier }

-- Terminal OR-address object classes

mHSNetworkAddress       OBJECT-CLASS ::= {
      SUBCLASS OF       {oRAddressElement}
      KIND              structural
      MUST CONTAIN      {mHSNetworkAddressAttribute}
      ID                id-oc-mhs-network-address }

mHSExtendedNetworkAddress OBJECT-CLASS ::= {
      SUBCLASS OF       {oRAddressElement}
      KIND              structural
      MUST CONTAIN      {mHSExtendedNetworkAddressAttribute}
      ID                id-oc-mhs-extended-network-address }

mHSTerminalIdentifier  OBJECT-CLASS ::= {
      SUBCLASS OF       {oRAddressElement}
      KIND              structural
      MUST CONTAIN      {mHSTerminalIdentifierAttribute}
      ID                id-oc-mhs-terminal-identifier }

mHSTerminalType         OBJECT-CLASS ::= {
      SUBCLASS OF       {oRAddressElement}
      KIND              structural
      MUST CONTAIN      {mHSTerminalTypeAttribute}
      ID                id-oc-mhs-terminal-type }

-- Numeric OR-address object classes

mHSNumericUserIdentifier OBJECT-CLASS ::= {
      SUBCLASS OF       {oRAddressElement}
      KIND              structural
      MUST CONTAIN      {mHSNumericUserIdentifierAttribute}
      ID                id-oc-mhs-numeric-user-identifier }

-- Postal OR-address object classes

mHSPDSName              OBJECT-CLASS ::= {
      SUBCLASS OF       {oRAddressElement}
      KIND              structural
      MUST CONTAIN      {mHSPDSNameAttribute}
      ID                id-oc-mhs-pds-name }

mHSPhysicalDeliveryCountry OBJECT-CLASS ::= {
      SUBCLASS OF       {mHSCountry}
      KIND              structural
      ID                id-oc-mhs-physical-delivery-country-name }

mHSPostalCode           OBJECT-CLASS ::= {
      SUBCLASS OF       {oRAddressElement}
      KIND              structural
      MUST CONTAIN      {mHSPostalCodeAttribute}
      ID                id-oc-mhs-postal-code }

-- OR-address element name

oRAddressElementName    ATTRIBUTE ::= {
      SUBTYPE OF        name
      SINGLE VALUE      TRUE
      ID                id-at-oraddress-element-name }
```

```
-- Common OR-address element names

mHSCountryName          ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName -- contains ISO 3166 and X.121 codes only --
      WITH SYNTAX       DirectoryString {ub-country-name-numeric-length}
      ID                id-at-mhs-country-name }

mHSADMDName             ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-domain-name-length}
      ID                id-at-mhs-admd-name }

mHSPRMDName             ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-domain-name-length}
      ID                id-at-mhs-prmd-name }

-- Mnemonic OR-address element names

mHSOrganizationName     ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-organization-name-length}
      ID                id-at-mhs-organization-name }

mHSOrganizationalUnitName ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-organizational-unit-name-length}
      ID                id-at-mhs-organizational-unit-name }

mHSCommonNameAttribute  ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-common-name-length}
      ID                id-at-mhs-common-name }

mHSSurnameAttribute     ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-surname-length}
      ID                id-at-mhs-surname }

mHSGivenNameAttribute   ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-given-name-length}
      ID                id-at-mhs-given-name }

mHSInitialsAttribute    ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-initials-length}
      ID                id-at-mhs-initials }

mHSGenerationQualifierAttribute ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-generation-qualifier-length}
      ID                id-at-mhs-generation-qualifier }

-- Terminal OR-address element names

mHSNetworkAddressAttribute ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-x121-address-length}
      ID                id-at-mhs-network-address }

mHSExtendedNetworkAddressAttribute ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-extended-network-address-length}
      ID                id-at-mhs-extended-network-address }

ub-extended-network-address-length INTEGER ::= 256

mHSTerminalIdentifierAttribute ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-terminal-id-length}
      ID                id-at-mhs-terminal-identifier }

mHSTerminalTypeAttribute ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-terminal-type-length}
      ID                id-at-mhs-terminal-type }

ub-terminal-type-length INTEGER ::= 5

-- Numeric OR-address element names

mHSNumericUserIdentifierAttribute ATTRIBUTE ::= {
      SUBTYPE OF        oRAddressElementName
      WITH SYNTAX       DirectoryString {ub-numeric-user-id-length}
      ID                id-at-mhs-numeric-user-identifier }
```

```
-- Postal OR-address element names

mHSPDSNameAttribute ATTRIBUTE ::= {
      SUBTYPE OF       oRAddressElementName
      WITH SYNTAX      DirectoryString {ub-pds-name-length}
      ID               id-at-mhs-pds-name-attribute }

mHSPostalCodeAttribute ATTRIBUTE ::= {
      SUBTYPE OF       oRAddressElementName
      WITH SYNTAX      DirectoryString {ub-postal-code-length}
      ID               id-at-mhs-postal-code }

-- OR-address-subtree name forms

mHSCountryNameForm     NAME-FORM ::= {
      NAMES            mHSCountry
      WITH ATTRIBUTES  {mHSCountryName}
      ID               id-nf-mhs-country }

mHSADMDNameForm        NAME-FORM ::= {
      NAMES            mHSADMD
      WITH ATTRIBUTES  {mHSADMDName}
      ID               id-nf-mhs-admd }

mHSPRMDNameForm        NAME-FORM ::= {
      NAMES            mHSPRMD
      WITH ATTRIBUTES  {mHSPRMDName}
      ID               id-nf-mhs-prmd }

mHSOrganizationNameForm NAME-FORM ::= {
      NAMES            mHSOrganization
      WITH ATTRIBUTES  {mHSOrganizationName}
      ID               id-nf-mhs-organization }

mHSOrganizationalUnitNameForm NAME-FORM ::= {
      NAMES            mHSOrganizationalUnit
      WITH ATTRIBUTES  {mHSOrganizationalUnitName}
      ID               id-nf-mhs-organizational-unit }

mHSCommonNameForm      NAME-FORM ::= {
      NAMES            mHSCommonName
      WITH ATTRIBUTES  {mHSCommonNameAttribute}
      ID               id-nf-mhs-common-name }

mHSSurnameNameForm     NAME-FORM ::= {
      NAMES            mHSSurname
      WITH ATTRIBUTES  {mHSSurnameAttribute}
      ID               id-nf-mhs-surname }

mHSGivenNameNameForm   NAME-FORM ::= {
      NAMES            mHSGivenName
      WITH ATTRIBUTES  {mHSGivenNameAttribute}
      ID               id-nf-mhs-given-name }

mHSInitialsNameForm    NAME-FORM ::= {
      NAMES            mHSInitials
      WITH ATTRIBUTES  {mHSInitialsAttribute}
      ID               id-nf-mhs-initials }

mHSGenerationQualifierNameForm NAME-FORM ::= {
      NAMES            mHSGenerationQualifier
      WITH ATTRIBUTES  {mHSGenerationQualifierAttribute}
      ID               id-nf-mhs-generation-qualifier }

mHSNetworkAddressNameForm NAME-FORM ::= {
      NAMES            mHSNetworkAddress
      WITH ATTRIBUTES  {mHSNetworkAddressAttribute}
      ID               id-nf-mhs-network-address }

mHSExtendedNetworkAddressNameForm NAME-FORM ::= {
      NAMES            mHSExtendedNetworkAddress
      WITH ATTRIBUTES  {mHSExtendedNetworkAddressAttribute}
      ID               id-nf-mhs-extended-network-address }

mHSTerminalIdentifierNameForm NAME-FORM ::= {
      NAMES            mHSTerminalIdentifier
      WITH ATTRIBUTES  {mHSTerminalIdentifierAttribute}
      ID               id-nf-mhs-terminal-identifier }

mHSTerminalTypeNameForm NAME-FORM ::= {
      NAMES            mHSTerminalType
      WITH ATTRIBUTES  {mHSTerminalTypeAttribute}
      ID               id-nf-mhs-terminal-type }

mHSNumericUserIdentifierNameForm NAME-FORM ::= {
      NAMES            mHSNumericUserIdentifier
      WITH ATTRIBUTES  {mHSNumericUserIdentifierAttribute}
      ID               id-nf-mhs-numeric-user-identifier }
```

```
mHSPDSNameNameForm NAME-FORM ::= {
      NAMES            mHSPDSName
      WITH ATTRIBUTES  {mHSPDSNameAttribute}
      ID               id-nf-mhs-pds-name }

mHSPhysicalDeliveryCountryNameForm NAME-FORM ::= {
      NAMES            mHSPhysicalDeliveryCountry
      WITH ATTRIBUTES  {mHSCountryName}
      ID               id-nf-mhs-physical-delivery-country }

mHSPostalCodeNameForm NAME-FORM ::= {
      NAMES            mHSPostalCode
      WITH ATTRIBUTES  {mHSPostalCodeAttribute}
      ID               id-nf-mhs-postal-code }

END -- of MHS-routing OR-address-subtree
```

Annex D
(normative)
OR-address-subtree structure

This annex describes the DIT structure of the OR-address-subtree using the name forms defined in 8.6.

The integer identifiers assigned in this annex and used in Figure D-1 are arbitrary and have no global (or standardized) significance. A particular structure rule identifier has significance only within the scope of the subschema in which it applies. However, the relationships shown between OR-address elements reflect those defined in 9.1.3.4 and are normative.

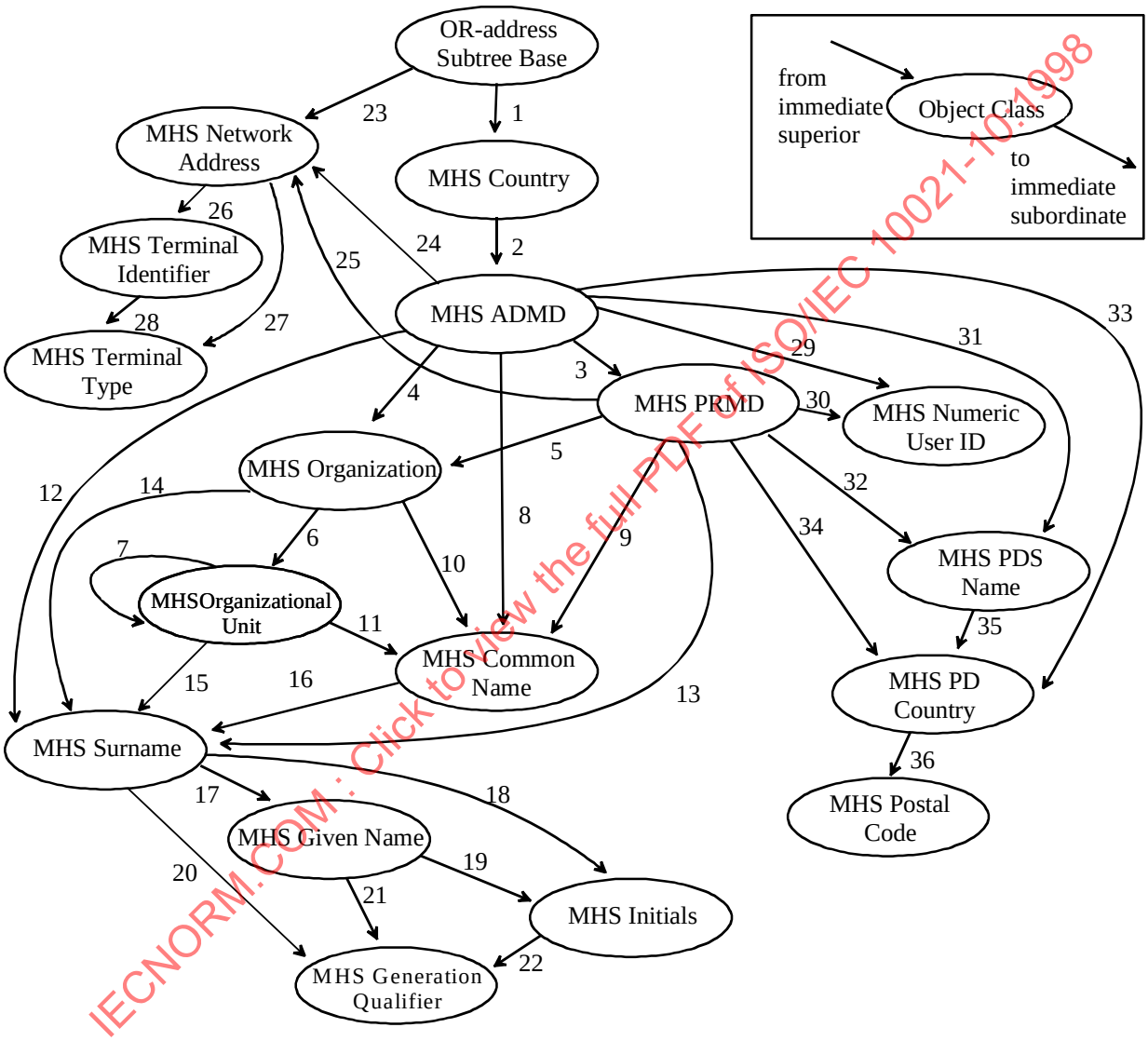

Figure D.1 - OR-address subtree structure

D.1 Common OR-address elements

NOTE – Certain structure rules may be disallowed under MHS ADMD, for an ADMD value of single space, in those countries where this value is permitted. Other structure rules may be disallowed under an ADMD value of single zero.

D.1.1 MHS Country

The attribute MHS Country Name is used for naming.

OR-address Subtree Base may be the immediate superior of entries of object class MHS Country.

```
sr1   STRUCTURE-RULE ::= {
      NAME FORM        mHSCountryNameForm
      ID               1 }
```