INTERNATIONAL TELECOMMUNICATION UNION

CCITT

THE INTERNATIONAL
TELEGRAPH AND TELEPHONE
CONSULTATIVE COMMITTEE

X.411

(11/1988)

SERIES X: DATA COMMUNICATION NETWORKS:
MESSAGE HANDLING SYSTEMS

MESSAGE HANDLING SYSTEMS: MESSAGE TRANSFER SYSTEM: ABSTRACT SERVICE DEFINITION AND PROCEDURES

Reedition of CCITT Recommendation X.411 published in the Blue Book, Fascicle VIII.7 (1988)

NOTES

1 CCITT Recommendation X.411 was published in Fascicle VIII.7 of the *Blue Book*. This file is an extract from the *Blue Book*. While the presentation and layout of the text might be slightly different from the *Blue Book* version, the contents of the file are identical to the *Blue Book* version and copyright conditions remain unchanged (see below).

2 In this Recommendation, the expression “Administration” is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Recommendation X.411

MESSAGE HANDLING SYSTEMS: MESSAGE TRANSFER SYSTEM: ABSTRACT SERVICE DEFINITION AND PROCEDURES[1)]

(Malaga-Torremolinos, 1984; amended at Melbourne, 1988)

The establishment in various countries of telematic services and computer-based store-and-forward message services in association with public data networks creates a need to produce standards to facilitate international message exchange between subscribers to such services.

The CCITT,

considering

(a) the need for Message Handling systems;

(b) the need to transfer messages of different types;

(c) that Recommendation X.200 defines the reference model of open systems interconnection for CCITT applications;

(d) that Recommendations X.208, X.217, X.218 and X.219 provide the foundation for CCITT applications;

(e) that the X.500-series Recommendations define directory systems;

(f) that Message Handling systems are defined in a series of Recommendations: X.400, X.402, X.403, X.407, X.408, X.411, X.413 and X.419; and

(g) that interpersonal messaging is defined in RecommendationsX.420 and T.330,

unanimously declares

(1) that the message transfer system (MTS) abstract service is defined in Section 2;

(2) that the message transfer agent (MTA) abstract service is defined in Section 3;

(3) that the procedures performed by message-transfer-agents (MTAs) to ensure that correct distributed operation of the message transfer system (MTS) are defined in Section 4.

TABLE OF CONTENTS

1) Recommendation X.411 and ISO 10021-4, Information Processing Systems - Text Communication - MOTIS - Message Transfer System: Abstract Service Definition and Procedures, were developed in close collaboration and are technically aligned, except for the differences noted in Annex C.

SECTION 1 – INTRODUCTION

0 Introduction

This Recommendation is one of a set of Recommendations 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*).

The MTS comprises a number of message-transfer-agents (MTAs), which transfer messages and deliver them to their intended recipients.

This Recommendation was developed jointly by CCITT and ISO. The equivalent ISO document is ISO 10021-4.

1 Scope

This Recommendation defines the abstract service provided by the MTS (the MTS abstract service), and specifies the procedures to be performed by MTAs to ensure the correct distributed operation of the MTS.

Recommendation X.402 identifies the other Recommendations which define other aspects of Message Handling Systems.

Access to the MTS abstract service defined in this Recommendation may be provided by the MTS access protocol (P3) defined in Recommendation X.419. The distributed operation of the MTS defined in this Recommendation may be provided by the use of the MTS transfer protocol (P1) also defined in Recommendation X.419.

Section 2 of this Recommendation defines the MTS abstract service. Paragraph 6 describes the message transfer system model. Paragraph 7 provides an overview of the MTS abstract service. Paragraph 8 defines the semantics of the parameters of the MTS abstract service. Paragraph 9 defines the abstract syntax of the MTA abstract service.

Section 3 of this Recommendation defines the MTA abstract service. Paragraph 10 refines the model of the MTS, first presented in § 6, to show that the MTS comprises a number of MTAs that interwork with one another to provide the MTS abstract service. Paragraph 11 provides an overview of the MTA abstract service. Paragraph 12 defines the semantics of the parameters of the MTA abstract service. Paragraph 13 defines the abstract-syntax of the MTA abstract service.

Section 4 of this Recommendation specifies the procedures performed by MTAs to ensure the correct distributed operation of the MTS.

Annex A provides a reference definition of the MTS object identifiers cited in the ASN.1 modules of this Recommendation.

Annex B provides a reference definition of the upper bounds of the size constraints imposed upon variable length data types defined in ASN.1 modules in the body of this Recommendation.

Annex C identifies the technical differences between ISO/IEC and CCITT versions of CCITT Recommendations X.411 and ISO/IEC 10021-4.

2 References

References are listed in Recommendation X.402.

3 Definitions

Definitions are listed in Recommendation X.402.

4 Abbreviations

Abbreviations are listed in Recommendation X.402.

5 Conventions

This Recommendation uses the descriptive conventions described below.

5.1 *Terms*

Throughout this Recommendation, the words of defined terms and the names and values of the parameters of the MTS abstract service and the MTA abstract service, unless they are proper names, begin with a lower-case letter and are linked by a hyphen thus: defined-term. Proper names begin with an upper-case letter and are not linked by a hyphen thus: Proper Name. In §§ 8 and 12, the names and values of the parameters of the MTS abstract service and the MTA abstract service are printed in bold.

5.2 *Presence of parameters*

In the Tables of parameters in §§ 8 and 12, the presence of each parameter is qualified as follows:

– Mandatory (M): A mandatory parameter shall always be present.

– Optional (O): An optional argument shall be present at the direction of the invoker of the abstract-operation; an optional result at the discretion of the performer of the abstract-operation.

– Conditional (C): A conditional parameter shall be present as defined by this [Recommendation/International Standard].

Where a conditional parameter shall be present due to some action on the message, probe or report by the MTS, this is explicitly defined. The presence of other conditional parameters is dependent on the presence of those parameters in other abstract-operations (for example, the presence of a conditional argument of the Message-transfer abstract-operation is dependent on the presence of the same optional argument in the related Message-submission abstract-operation).

5.3 *Abstract syntax definitions*

This Recommendation defines the abstract-syntax of the MTS abstract service and the MTA abstract service using the abstract syntax notation (ASN.1) defined in Recommendation X.208, and the abstract service definition conventions defined in Recommendation X.407.

Where there are changes implied to the protocols defined in Recommendation X.411 (1984), these are highlighted in the abstract syntax definitions by means of <u>underlining</u>.

SECTION 2 – MESSAGE TRANSFER SYSTEM ABSTRACT SERVICE

6 Message transfer system model

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*).

The MTS is described using an abstract model in order to define the services provided by the MTS as a whole – the MTS abstract service.

The MTS is modelled as an *object*, whose overall behaviour can be described without reference to its internal structure. The services provided by the MTS object are made available at *ports*. A type of port represents a particular view of the services provided by the MTS object.

A user of the MTS is also modelled as an object, which obtains the services provided by the MTS through a port which is *paired* with an MTS port of the same type.

A type of port corresponds to a set of a *abstract-operations* which can occur at the port; those which can be performed by the MTS object (invoked by the MTS-user object), and those which can be invoked by the MTS object (performed by the MTS-user object).

A port may be symmetrical, in which case the set of operations performed by the MTS object may also be invoked by the MTS object, and vice versa. Otherwise, the port is asymmetrical, in which case the object is said to be the *supplier* or *consumer* with respect to the type of port. The terms *supplier* and *consumer* are used only to distinguish between the roles of a pair of ports in invoking or performing operations. The assignment of the terms is usually intuitive when one object is providing a service used by another object; the service object (e.g., the MTS) is usually regarded as being the *supplier*, and the user object (e.g., an MTS-user object) is usually regarded as being the *consumer*.

Before objects can invoke operations on one another, they must be bound into an abstract *association*. The binding of an assocation between the objects establishes a relationship between the objects which lasts until the association is released. An association is always released by the initiator of the association. The binding of an association establishes the *credentials* of the objects to interact, and the *application-context* and *security-context* of the association. The *application-context* of an association may be one or more types of port paired between two objects.

The model presented is abstract. That is, it is not always possible for an outside observer to identify the boundaries between objects, or to decide on the moment or the means by which operations occur. However, in some cases the abstract model will be *realised*. For example, a pair of objects which communicate through paired ports may be located in different open systems. In this case, the boundary between the objects is visible, the ports are exposed, and the operations may be supported by instances of OSI communication.

The MTS object supports ports of three different types: a *submission-port*, a *delivery-port* and an *administration-port*.

A submission-port enables an MTS-user to submit messages to the MTS for transfer and delivery to one or more recipient MTS-users, and to probe the ability of the MTS to deliver a subject-message.

A delivery-port enables an MTS-user to accept delivery of messages from the MTS, and to accept reports on the delivery or non-delivery of messages and probes.

An administration-port enables an MTS-user to change long term parameters held by the MTS associated with message delivery, and enables either the MTS or the MTS-user to change their *credentials* with one another.

A message submitted by one MTS-user via a submission-port will normally be delivered to one or more recipient MTS-users via delivery ports. The originating MTS-user may elect to be notified of the delivery of a message via its delivery-port.

Figure 1/X.411 models the message transfer system (MTS).

Paragraph 7 provides an overview of the MTS Abstract Service.

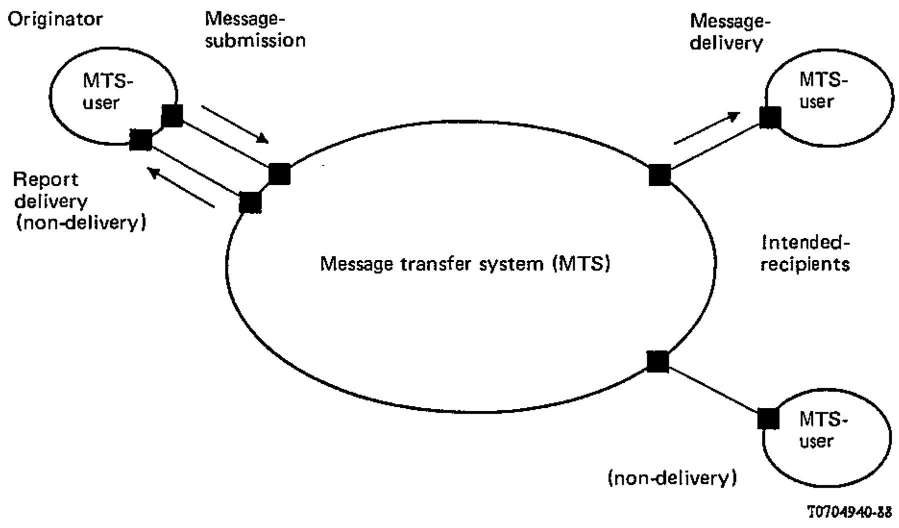


FIGURE 1/X.411

Message transfer system model

7 Message transfer system abstract service overview

This Recommendation defines the following services that comprise the MTS abstract service:

MTS bind and unbind

a) MTS-bind

b) MTS-unbind

Submission port abstract operations

c) Message-submission

d) Probe-submission

e) Cancel-deferred-delivery

f) Submission-control

Delivery port abstract operations

g) Message-delivery

h) Report-delivery

i) Delivery-control

Administration port abstract operations

j) Register

k) Change-credentials.

7.1 *MTS bind and unbind*

The **MTS-bind** enables either the MTS-user to establish an association with the MTS, or the MTS to establish an association with the MTS-user. Abstract-operations other than MTS-bind can only be invoked in the context of an established association.

The **MTS-unbind** enables the release of an established association by the initiator of the association.

7.2 *Submission port*

The **message-submission** abstract-operation enables an MTS-user to submit a message to the MTS for transfer and delivery to one or more recipient MTS-users.

The **probe-submission** abstract-operation enables an MTS-user to submit a probe in order to determine whether or not a message could be transferred and delivered to one or more recipient MTS-users if it were to be submitted.

The **cancel-deferred-delivery** abstract-operation enables an MTS-user to request cancellation of a message previously submitted (for deferred delivery) by invocation of the message-submission-abstract-operation.

The **submission-control** abstract-operation enables the MTS to constrain the use of the submission-port abstract-operations by the MTS-user.

The **message-submission** and **Probe-submission** abstract-operations may cause subsequent invocation of the Report-delivery abstract-operation by the MTS.

7.3 *Delivery port*

The **message-delivery** abstract-operation enables the MTS to deliver a message to the MTS-user.

The **report-delivery** abstract-operation enables the MTS to acknowledge to the MTS-user the outcome of a previous invocation of the message-submission or probe-submission abstract-operations. For the message- submission abstract-operation, the report-delivery abstract-operation indicates the delivery or non-delivery of the submitted message. For the probe-submission abstract-operation, the report-delivery abstract-operation indicates whether or not a message could be delivered if it were to be submitted. The report-delivery abstract-operation may also convey a notification of physical-delivery by a PDS.

The **delivery-control** abstract-operation enables an MTS-user to constrain the use of the delivery-port abstract-operations by the MTS.

7.4 *Administration port*

The **register** abstract-operation enables an MTS-user to change long term parameters of the MTS-user held by the MTS, associated with message delivery.

The **change-credentials** abstract-operation enables either an MTS-user to change its **credentials** with the MTS, or the MTS to change its **credentials** with the MTS-user.

8 Message transfer system abstract service definition

This section defines the semantics of the parameters of the MTS abstract service.

Paragraph 8.1 defines the MTS-bind and MTS-unbind. Paragraph 8.2 defines the submission-port. Paragraph 8.3 defines the delivery-port. Paragraph 8.4 defines the administration-port. Paragraph 8.5 defines some common parameter types.

The abstract-syntax of the MTS abstract service is defined in § 9.

8.1 *MTS-bind and MTS-unbind*

This section defines the MTS-bind and MTS-unbind used to establish and release associations between an MTS-user and the MTS.

8.1.1 *Abstract-bind and abstract-unbind*

This section defines the following abstract-bind and abstract-unbind operations:

a) MTS-bind

b) MTS-unbind.

8.1.1.1 *MTS-bind*

The MTS-bind enables an MTS-user to establish an association with the MTS, or the MTS to establish an association with an MTS-user.

The MTS-bind establishes the **credentials** of an MTS-user and the MTS to interact, and the **application-context** and **security-context** of the association. An association can only be released by the initiator of that association (using MTS-unbind).

Abstract-operations other than MTS-bind can only be invoked in the context of an established association.

The successful completion of the MTS-bind signifies the establishment of an association.

The disruption of the MTS-bind by a bind-error indicates that an association has not been established.

8.1.1.1.1 *Arguments*

Table 1/X.411 lists the arguments of the MTS-bind, and for each argument qualifies its presence and indicates the clause in which the argument is defined.

TABLE 1/X.411

MTS-bind arguments

Argument	Presence	Clause
Bind arguments		
Initiator-name	M	8.1.1.1.1.1
Initiator-credentials	M	8.1.1.1.1.2
Security-context	O	8.1.1.1.1.3
Messages-waiting	O	8.1.1.1.1.4

8.1.1.1.1.1 *Initiator-name*

This argument contains a name for the initiator of the association. It shall be generated by the initiator of the association.

If the initiator is an MTS-user, the name is the **OR-name** of the MTS-user, which is registered with the MTS (see § 8.4.1.1.1.1). The **initiator-name** shall contain the **OR-address**, and may optionally also contain the **directory-name**, of the MTS-user (**OR-address-and-optional-directory-name**). For secure messaging, when an MS is involved, the **initiator-name** may also indicate whether the initiator is a UA or an MS.

If the initiator is the MTS (or an MTA - see § 11), the name is an **MTA-name**, which is known to the MTS-user.

8.1.1.1.1.2 *Initiator-credentials*

This argument contains the **credentials** of the initiator of the association. It shall be generated by the initiator of the association.

The **initiator-credentials** may be used by the responder to authenticate the identity of the initiator (see Recommendation X.509).

If only simple-authentication is used, the **initiator-credentials** comprise a simple **password** associated with the **initiator-name**.

If strong-authentication is used, the **initiator-credentials** comprise an **initiator-bind-token** and, optionally, an **initiator-certificate**.

The **initiator-bind-token** is a token generated by the initiator of the association. If the **initiator-bind-token** is an **asymmetric-token**, the **signed-data** comprises a **random number**. The **encrypted-data** of an **asymmetric-token** may be used to convey secret security-relevant information (e.g., one or more symmetric-encryption-keys) used to secure the association, or may be absent from the **initiator-bind-token**.

The **initiator-certificate** is a **certificate** of the initiator of the association, generated by a trusted source (e.g., a certification-authority). It may be supplied by the initiator of the association, if the **initiator-bind-token** is an **asymmetric-token**. The **initiator-certificate** may be used to convey a verified copy of the public-asymmetric-encryption-key (**subject-public-key**) of the initiator of the association. The initiator's public-asymmetric-encryption-key may be used by the responder to compute the **responder-bind-token**. If the responder is known to have, or have access to, the initiator's **certificate** (e.g., via the change-credentials abstract-operation, or via the directory), the **initiator-certificate** may be omitted.

8.1.1.1.1.3 *Security-context*

This argument identifies the **security-context** that the initiator of the association proposes to operate at. It may be generated by the initiator of the association.

The **security-context** comprises one or more **security-labels** that define the sensitivity of interactions that may occur between the MTS-user and the MTS for the duration of the association, in line with the security-policy in force. The **security-context** shall be one that is allowed by the registered **user-security-labels** of the MTS-user and by the **security-labels** associated with the MTA of the MTS.

Once established, the **security-context** of the submission-port and delivery-port can be temporarily restricted using the submission-control (see § 8.2.1.4.5) and delivery-control (see § 8.3.1.3.1.7) abstract-operation, respectively.

If **security-contexts** are not established between the MTS-user and the MTS, the sensitivity of interactions that may occur between the MTS-user and the MTS may be at the discretion of the invoker of an abstract-operation.

8.1.1.1.1.4 *Messages-waiting*

This argument indicates that the number of messages and total number of octets waiting to be delivered by the MTS to the MTS-user, for each **priority**. It may be generated by the initiator of the association.

This argument shall only be present when the MTS is initiating an association with an MTS-user, and when the MTS-user subscribes to the hold for delivery element-of-service (defined in Recommendation X.400).

8.1.1.1.2 *Results*

Table 2/X.411 lists the results of the MTS-bind, and for each result qualifies its presence and indicates the clause in which the result is defined.

TABLE 2/X.411

MTS-bind results

Result	Presence	Clause
Bind results		
Responder-name	M	8.1.1.1.2.1
Responder-credentials	M	8.1.1.1.2.2
Messages-waiting	O	8.1.1.1.2.3

8.1.1.1.2.1 *Responder-name*

This argument contains a name for the responder of the association. It shall be generated by the responder of the association.

If the responder is an MTS-user, the name is the **OR-name** of the MTS-user, which is registered with the MTS (see § 8.4.1.1.1.1). The **responder-name** shall contain the **OR-address**, and may optionally also contain the **directory-name**, of the MTS-user (**OR-address-and-optional-directory-name**). For secure messaging, when an MS is involved, the **responder-name** may also indicate whether the initiator is a UA or an MS.

If the responder is the MTS (or an MTA - see § 11), the name is an **MTA-name**, which is known to the MTS-user.

8.1.1.1.2.2 *Responder-credentials*

This argument contains the **credentials** of the responder of the association. It shall be generated by the responder of the association.

The **responder-credentials** may be used by the initiator to authenticate the identity of the responder (see Recommendation X.509).

If only simple-authentication is used, the **responder-credentials** comprise a simple **password** associated with the **responder-name**.

If strong-authentication is used, the **responder-credentials** comprise a **responder-bind-token**. The responder-bind-token is a **token** generated by the responder of the association. The **responder-bind-token** shall be the same type of **token** as the **initiator-bind-token**. If the **responder-bind-token** is an **asymmetric-token**, the **signed-data** comprises a **random-number** (which may be related to the **random-number** supplied in the **initiator-bind-token**). The **encrypted-data** of an **assymetric-token** may be used to convey secret security-relevant information (e.g., one or more symmetric-encryption-keys) used to secure the association, or may be absent from the **responder-bind-token**.

8.1.1.1.2.3 *Messages-waiting*

This argument indicates the number of messages and total number of octets waiting to be delivered by the MTS to the MTS-user, for each **priority**. It may be generated by the responder of the association.

This argument shall only be present when the MTS is responding to an association initiated by an MTS-user, and when the MTS-user subscribes to the hold for delivery element-of-service (defined in Recommendation X.400).

8.1.1.1.3 *Bind-errors*

The bind-errors that may disrupt the MTS-bind are defined in § 8.1.2.

8.1.1.2 *MTS-unbind*

The MTS-unbind enables the release of an established association by the initiator of the association.

8.1.1.2.1 *Arguments*

The MTS-unbind has no arguments.

8.1.1.2.2 *Results*

The MTS-unbind returns an empty result as indication of release of the association.

8.1.1.2.3 *Unbind-errors*

There are no unbind-errors that may disrupt the MTS-unbind.

8.1.2 *Bind-errors*

This section defines the following bind-errors:

a) Authentication-error

b) Busy

c) Unacceptable-dialogue-mode

d) Unacceptable-security-context.

8.1.2.1 *Authentication-error*

The authentication-error bind-error reports that an association cannot be established due to an authentication error; the initiator's **credentials** are not acceptable or are improperly specified.

The authentication-error bind-error has no parameters.

8.1.2.2 *Busy*

The busy bind-error reports that an association cannot be established because the responder is busy.

The busy-bind-error has no parameters.

8.1.2.3 *Unacceptable-dialogue-mode*

The unacceptable-dialogue-mode bind-error reports that the dialogue-mode proposed by the initiator of the association is unacceptable to the responder (see Recommendation X.419).

The unacceptable-dialogue-mode bind-error has no parameters.

8.1.2.4 *Unacceptable-security-context*

The unacceptable-security-context bind-error reports that the **security-context** proposed by the initiator of the association is unacceptable to the responder.

The unacceptable-security-context bind-error reports that the **security-context** proposed by the initiator of the association is unacceptable to the responder.

The unacceptable-security-context bind-error has no parameters.

8.2 *Submission port*

This section defines the abstract-operations and abstract-errors which occur at a submission-port.

8.2.1 *Abstract-operations*

This section defines the following submission-port abstract-operations.

a) Message-submission

b) Probe-submission

c) Cancel-deferred-delivery

d) Submission-control.

8.2.1.1 *Message-submission*

The message-submission abstract-operation enables an MTS-user to submit a message to the MTS for transfer and delivery to one or more recipient MTS-users.

The successful completion of the abstract-operation signifies that the MTS has accepted responsibility for the message (but not that it has yet delivered it to its intended recipients).

The disruption of the abstract-operation by an abstract-error indicates that the MTS cannot assume responsibility for the message.

8.2.1.1.1 *Arguments*

Table 3/X.411 lists the arguments of the message-submission abstract-operation, and for each argument qualifies its presence and identifies the clause in which the argument is defined.

8.2.1.1.1.1 *Originator-name*

This argument contains the **OR-name** of the originator of the message. It shall be generated by the originating MTS-user.

The **originator-name** contains the **OR-name** of an individual originator, i.e., it shall not contain the **OR-name** of a DL.

8.2.1.1.1.2 *Recipient-name*

This argument contains the **OR-name** of a recipient of the message. It shall be generated by the originator of the message. A different value of this argument shall be specified for each recipient of the message.

The **recipient-name** contains the **OR-name** of an individual recipient or DL.

8.2.1.1.1.3 *Alternate-recipient-allowed*

This argument indicates whether the message may be delivered to an alternate-recipient assigned by the recipient-MD, if the specified **recipient-name** does not identify an MTS-user. It may be generated by the originator of the message.

This argument may have one of the following values: **alternate-recipient-allowed** or **alternate-recipient-prohibited**.

TABLE 3/X.411

Message-submission arguments

Argument	Presence	Clause
Originator argument		
Originator-name	M	8.2.1.1.1.1
Recipient arguments		
Recipient-name	M	8.2.1.1.1.2
Alternate-recipient-allowed	O	8.2.1.1.1.3
Recipient-reassignment-prohibited	O	8.2.1.1.1.4
Originator-requested-alternate-recipient	O	8.2.1.1.1.5
DL-expansion-prohibited	O	8.2.1.1.1.6
Disclosure-of-recipients	O	8.2.1.1.1.7
Priority argument		
Priority	O	8.2.1.1.1.8
Conversion arguments		
Implicit-conversion-prohibited	O	8.2.1.1.1.9
Conversion-with-loss-prohibited	O	8.2.1.1.1.10
Explicit-conversion	O	8.2.1.1.1.11
Delivery time arguments		
Deferred-delivery-time	O	8.2.1.1.1.12
Latest-delivery-time	O	8.2.1.1.1.13
Delivery method argument		
Requested-delivery-method	O	8.2.1.1.1.14
Physical delivery arguments		
Physical-forwarding-prohibited	O	8.2.1.1.1.15
Physical-forwarding-address-request	O	8.2.1.1.1.16
Physical-deli very-modes	O	8.2.1.1.1.17
Registered-mail-type	O	8.2.1.1.1.18
Recipient-number-for-advice	O	8.2.1.1.1.19
Physical-rendition-attributes	O	8.2.1.1.1.20
Originator-return-address	O	8.2.1.1.1.21
Report request arguments		
Originator-report-request	M	8.2.1.1.1.22
Content-return-request	O	8.2.1.1.1.23
Physical-delivery-report-request	O	8.2.1.1.1.24
Security arguments		
Originator-certificate	O	8.2.1.1.1.25
Message-token	O	8.2.1.1.1.26
Content-confidentiality-algorithm-identifier	O	8.2.1.1.1.27
Content-integrity-check	O	8.2.1.1.1.28
Message-origin-authentication-check	O	8.2.1.1.1.29
Message-security-label	O	8.2.1.1.1.30
Proof-of-submission-request	O	8.2.1.1.1.31
Proof-of-delivery-request	O	8.2.1.1.1.32
Content arguments		
Original-encoded-information-types	O	8.2.1.1.1.33
Content-type	M	8.2.1.1.1.34
Content-identifier	O	8.2.1.1.1.35
Content-correlator	O	8.2.1.1.1.36
Content	M	8.2.1.1.1.37

If this argument has the value **alternate-recipient-allowed** and the **recipient-name** (specified by the originator of the message, or added by DL-expansion, or substituted by redirection to the **recipient-assigned-alternate-recipient** or the **originator-requested-alternate-recipient**, or present by any combination of redirection and expansion) does not identify an MTS-user, the message may be redirected to an alternate-recipient assigned by the recipient-MD to receive such messages. If no such alternate-recipient has been assigned by the recipient-MD, or if this argument has the value **alternate-recipient-prohibited**, a non-delivery report shall be generated.

In the absence of this argument, the default **alternate-recipient-prohibited** shall be assumed.

8.2.1.1.1.4 *Recipient-reassignment-prohibited*

This argument indicates whether the message may be reassigned to a **recipient-assigned-alternate-recipient** registered by the intended-recipient. It may be generated by the originator of the message.

This argument may have one of the following values: **recipient-reassignment-prohibited** or **recipient-reassignment-allowed**.

If this argument has the value **recipient-reassignment-allowed** and the intended-recipient has registered a **recipient-assigned-alternate-recipient**, the message shall be redirected to the **recipient-assigned-alternate-recipient**.

If this argument has the value **recipient-reassignment-prohibited** and the intended-recipient has registered a **recipient-assigned-alternate-recipient**, then if an **originator-requested-alternate-recipient** has been specified by the originator of the message, the message shall be redirected to the **originator-requested-alternate-recipient**, or if no **originator-requested-alternate-recipient** has been specified by the originator of the message, a non-delivery-report shall be generated.

In the absence of this argument, the default **recipient-reassignment-allowed** shall be assumed.

8.2.1.1.1.5 *Originator-requested-alternate-recipient*

This argument contains the **OR-name** of the alternate-recipient requested by the originator of the message. It may be generated by the originator of the message. A different value of this argument may be specified for each recipient of the message.

The **originator-requested-alternate-recipient** contains the **OR-name** of an individual, or DL, alternate-recipient.

If this argument is present and delivery of the message to the **recipient-name** (specified by the originator of the message, or added by DL-expansion, or substituted by redirection to the **originator-requested-alternate-recipient** specified by this argument.

If an **originator-requested-alternate-recipient** has been specified by the originator of the message, this message shall be redirected to that alternate recipient in preference to one assigned by the recipient-MD.

8.2.1.1.1.6 *DL-expansion-prohibited*

This argument indicates whether DL-expansion within an MTS shall occur for any **recipient-name** which denotes a DL. It may be generated by the originator of the message.

This argument may have one of the following values: **DL-expansion-prohibited** or **DL-expansion-allowed**.

In the absence of this argument, the default **DL-expansion-allowed** shall be assumed.

8.2.1.1.1.7 *Disclosure-of-recipients*

This argument indicates whether the **recipient-name** of all recipients are to be indicated to each recipient MTS-user when the message is delivered. It may be generated by the originator of the message.

This argument may have one of the following values: **disclosure-of-recipients-allowed** or **disclosure-of-recipients-prohibited**.

In the absence of this argument, the default **disclosure-of-recipients-prohibited** shall be assumed.

8.2.1.1.1.8 *Priority*

This argument specifies the relative priority of the message: **normal**, **non-urgent** or **urgent**. It may be generated by the originator of the message.

In the absence of this argument, a default **priority** of **normal** shall be assumed.

8.2.1.1.1.9 *Implicit-conversion-prohibited*

This argument indicates whether implicit-conversion may be performed on the message **content**. It may be generated by the originator of the message.

This argument may have one of the following values: **implicit-conversion-prohibited** or **implicit-conversion-allowed**.

In the absence of this argument, the default **implicit-conversion-allowed** shall be assumed.

See also § 8.2.1.1.1.10.

8.2.1.1.1.10 *Conversion-with-loss-prohibited*

This argument indicates whether **encoded-information-type** conversion(s) may be carried out on the message **content**, if such conversion(s) would result in loss of information. Loss of information is defined in Recommendation X.408. It may be generated by the originator of the message.

This argument may have one of the following values: **conversion-with-loss-prohibited** or **conversion-with-loss-allowed**.

In the absence of this argument, the default **conversion-with-loss-allowed** shall be assumed.

The combined effect of the **implicit-conversion-prohibited** and **conversion-with-loss-prohibited** arguments relate to implicit-conversion only and is defined in Table 4/X.411.

TABLE 4/X.411

Combined effect of conversion arguments

Implicit conversion	Conversion with loss	Combined effect
allowed	with-loss-allowed	allowed
allowed	with-loss-prohibited	with-loss-prohibited
prohibited	with-loss-allowed	prohibited
prohibited	with-loss-prohibited	prohibited

8.2.1.1.1.11 Explicit-conversion

This argument indicates the type of conversion of the message **content** explicitly requested by the originator for the recipient. It may be generated by the originator of the message. A different value of this argument may be specified for each recipient of the message.

This argument may have one of the following values: **no-explicit-conversion**, **ia5-text-to-teletex**, **teletex-to-telex**, **telex-to-ia5-text**, **telex-to-teletex**, **telex-to-g4-class-1**, **telex-to-videotex**, **ia5-text-to-telex**, **telex-to-g3-facsimile**, **ia5-text-to-g3-facsimile**, **ia5-text-to-g4-class-1**, **ia5-text-to-videotex**, **teletex-to-ia5-text**, **teletex-to-g3-facsimile**, **teletex-to-g4-class-1**, **teletex-to-videotex**, **videotex-to-telex**, **videotex-to-ia5-text**, or **videotext-to-teletex**. Other types of **explicit-conversion** may be defined by future versions of this Recommendation. **Explicit-conversion** shall be performed as specified in Recommendation X.408.

In the absence of this argument, the default **no-explicit conversion** shall be assumed.

Note – When specified for a recipient DL, **explicit-conversion** applies to all members of the DL.

8.2.1.1.1.12 *Deferred-delivery-time*

This argument specifies the **time** before which the message should not be delivered to the recipient(s). It may be generated by the originator of the message.

8.2.1.1.1.13 *Latest-delivery-time*

This argument contains the **time** after which the message should not be delivered to the recipient(s). It may be generated by the originator of the message.

The handling of non-delivery because of **latest-delivery-time** is described in § 14.3.2.4.

8.2.1.1.1.14 *Requested-delivery-method*

This argument indicates the requested method of delivery of the message to the recipient. It may be generated by the originator of the message. A different value of this argument may be specified for each recipient of the message.

This argument may have one or more of the following values: **any-delivery-method**, **mhs-delivery**, **physical-delivery**, **teletex-delivery**, **g3-facsimile-delivery**, **g4-facsimile-delivery**, **ia5-terminal-delivery**, **videotex-delivery** or **telephone-delivery**.

If more than one value of this argument is specified for a recipient, the sequence of the values shall be assumed to imply the originator's order of preference of delivery-methods.

In the absence of this argument, the default **any-delivery-method** shall be assumed.

If the **recipient-name** generated by the originator of the message contains a **directory-name** but omits an **OR-address**, the MTS may use the **requested-delivery-method** as an indication of which form of **OR-address** the **directory-name** should be mapped to by the MTS (e.g., using the Directory). If a form of **OR-address** appropriate to a **requested-delivery-method** cannot be found, a **recipient-improperly-specified** abstract error shall be returned to the originator of the message.

If the **recipient-name** generated by the originator of the message contains an **OR-address** of a form not appropriate to a **requested-delivery-method**, a non-delivery report shall be returned to the originator of the message.

If the originator-supplied **requested-delivery-method** conflicts with the recipient's preferred delivery-method (e.g., as registered in the Directory in the mhs-preferred-delivery-method attribute), the originator's **requested-delivery-method** takes precedence. If the originator's conversion requirements (see §§ 8.2.1.1.1.9 to 8.2.1.1.1.11), a non-delivery report shall be returned to the originator of the message.

8.2.1.1.1.15 *Physical-forwarding-prohibited*

This argument indicates whether physical-forwarding of the message is prohibited. It may be generated by the originator of the message if the **requested-delivery-method** argument specifies that physical-delivery is required to the recipient, or if the originator of the message supplied a **postal-OR-address** for the recipient. A different value of this argument may be specified for each recipient of the message.

This argument may have one of the following values: **physical-forwarding-allowed**, or **physical-forwarding-prohibited**.

In the absence of this argument, the default **physical-forwarding-allowed** shall be assumed.

8.2.1.1.1.16 *Physical-forwarding-address-request*

This argument indicates whether the physical-forwarding-address of the recipient is to be returned in this report. It may be generated by the originator of the message if the **requested-delivery-method** argument specifies that physical-delivery is required to the recipient, or if the originator of the message supplied a **postal-OR-address** for the recipient. A different value of this argument may be specified for each recipient of the message.

This argument may have one of the following values: **physical-forwarding-address-requested** or **physical-forwarding-address-not-requested**.

In the absence of this argument, the default **physical-forwarding-address-not-requested** shall be assumed.

A physical-forwarding-address may be requested when physical-forwarding is prohibited or allowed (see § 8.2.1.1.1.15).

8.2.1.1.1.17 *Physical-delivery-modes*

This argument indicates the mode of physical-delivery to the recipient to be used. It may be generated by the originator of the message if the **requested-delivery-method** argument specifies that physical-delivery is required to the recipient, or if the originator of the message supplied a **postal-OR-address** for the recipient. A different value of this argument may be specified for each recipient of the message.

This argument may have one of the following values: **ordinary-mail**, **special-delivery**, **express-mail**, **counter-collection**, **counter-collection-with-telephone-advice**, **counter-collection-with-telex-advice**, **counter-collection-with-teletex-advice**, or **bureau-fax-delivery**.

Note that **bureau-fax-delivery** comprises all A to H modes of delivery defined in Recommendation F.170, i.e., A – regular delivery, B – special delivery, C – express mail, D – counter collection, E – counter collection with telephone advice, F – telefax, G – counter collection with telex advice, and H – counter collection with teletex advice.

In the absence of this argument, the default **ordinary-mail** shall be assumed.

8.2.1.1.1.18 *Registered-mail-type*

This argument indicates the type of registered mail service to be used physically deliver the message to the recipient. It may be generated by the originator of the message if the **requested-delivery-method** argument specifies that physical delivery is required to the recipient, or if the originator of the message supplied a **postal-OR-address** for the recipient. A different value of this argument may be specified for each recipient of the message.

This argument may have one of the following values: **non-registered-mail**, **registered-mail**, or **registered-mail-to-addressee-in-person**.

In the absence of this argument, the default **ordinary-mail** shall be assumed.

8.2.1.1.1.19 *Recipient-number-for-advice*

This argument contains the telephone, telex or teletex number of the recipient, to be used in conjunction with the **counter-collection-with-advice** and **bureau-fax-delivery physical-delivery-modes**. It may be generated by the originator of the message if the **requested-delivery-method** argument specifies that physical-delivery is required to the recipient, or if the originator of the message supplied a **postal-OR-address** for the recipient, and the **physical-delivery-modes** argument specifies a **counter-collection-with-advice** or **bureau-fax-delivery physical-delivery-mode**. A different value of this argument may be specified for each recipient of the message.

8.2.1.1.1.20 *Physical-rendition-attributes*

This argument indicates the **physical-rendition-attributes** of the message. It may be generated by the originator of the message if the **requested-delivery-method** argument specifies that physical-delivery is required to the recipient, or if the originator of the message supplied a **postal-OR-address** for the recipient. A different value of this argument may be specified for each recipient of the message.

This argument may have one of the following values: **basic**. Future versions of this Recommendation may define other values of this argument. Other values of this argument may be used by bilateral agreement between MDs.

In the absence of this argument, the default **basic** shall be assumed.

8.2.1.1.1.21 Originator-return-address

This argument contains the **postal-OR-address** of the originator of the message. It shall be generated by the originator of the message if the **requested-delivery-method** argument specifies that physical-delivery is required to one or more recipients of the message, or if the originator of the message supplied one or more **postal-OR-address** for the recipients. It may also be generated by the originator of the message if a recipient DL contains, or is likely to contain, one or more members for whom physical-delivery is required.

The **originator-return-address** shall contain the **postal-OR-address** of an individual originator (**OR-address**), i.e., shall not contain the **directory-name** of an individual originator nor the **directory-name** of a DL.

8.2.1.1.1.22 *Originator-report-request*

This argument indicates the kind of report requested by the originator of the message. It shall be generated by the originator of the message. A different value of this argument may be specified for each recipient of the message.

This argument may have one of the following values:

– **no-report**: the originator of the message requested the suppression of non-delivery-reports;

– **non-delivery-report**: a report is returned only in case of non-delivery;

– **report**: a report is returned in case of delivery or non-delivery.

Note that the value of this argument may be changed at a DL expansion-point in line with the reporting-policy of the DL. Such a change may affect the number and type of reports the originator of the message may receive about delivery to a DL.

8.2.1.1.1.23 *Content-return-request*

This argument indicates whether the message **content** is to be returned with any non-delivery-report(s). It may be generated by the originator of the message.

This argument may have one of the following values: **content-return-requested** or **content-return-not-requested**.

In the absence of this argument, the default **content-return-not-requested** shall be assumed.

Note that the suppression of non-delivery-reports by the originator of the message (see § 8.2.1.1.1.22) takes precedence over a request for the return of the **content**.

Note that in the case of non-delivery-reports delivered to the owner of a DL (see § 8.3.1.2.1.4), the message **content** shall not be present.

8.2.1.1.1.24 *Physical-delivery-report-request*

This argument indicates the type of physical-delivery-report requested by the originator of the message. It may be generated by the originator of the message if the **requested-delivery-method** argument specifies that physical-delivery is required to the recipient or if the originator of the message supplied a **postal-OR-address** for the recipient. A different value of this argument may be specified for each recipient of the message.

This argument may have one of the following values: **return-of-undeliverable-mail-by-PDS, return-of-notification-by PDS, return-of-notification-by-MHS**, or **return-of-notification-by-MHS-and-PDS**.

In the absence of this argument, the default **return-of-undeliverable-mail-by-PDS** shall be assumed.

8.2.1.1.1.25 *Originator-certificate*

This argument contains the **certificate** of the originator of the message. It shall be generated by a trusted source (e.g. a certification-authority), and may be supplied by the originator of the message.

The **originator-certificate** may be used to convey a verified copy of the public-asymmetric-encryption-key (**subject-public-key**) of the originator of the message.

The originator's public-asymmetric-encryption-key may be used by the recipient(s) of the message to validate the **message-token**, if an **asymmetric-token**, if an **asymmetric-token** is used.

The originator's public-asymmetric-encryption-key may also be used by the recipient(s) of the message, and any MTA through which the message is transferred, to validate the **message-origin-authentication-check**.

8.2.1.1.1.26 *Message-token*

This argument contains the **token** associated with the message. It may be generated by the originator of the message. A different value of this argument may be specified for each recipient of the message.

If the **message-token** is an **asymmetric-token**, the **signed-data** may comprise:

– any of the following arguments: the content-confidentiality-algorithm-identifier, the content-integrity-check, the message-security-label, and the proof-of-delivery-request; and

– a message-sequence-number, that identifies the position of the message in a sequence of messages from the originator to the recipient to which the message-token relates (to provide the Message Sequence Integrity element-of-service, as defined in Recommendaton X.400).

If the **message-token** is an **asymmetric-token**, the **encrypted-data** may comprise:

- a **content-confidentiality-key**: a symmetric-encryption-key used with the **content-confidentiality-algorithm-identifier** by the originator of the message to encrypt the message **content**, and by the recipient to decrypt the message **content**; and/or
- the **content-integrity-check**: may be included in the **encrypted-data**, rather than the **signed-data**, if confidentiality of the **content-integrity-check** is required, and/or if the **message-security-label** is included in the **encrypted-data** (for confidentiality of the **message-security-label**) and the association between **content-integrity-check** and the **message-security-label** is to be maintained;
- the **message-security-label**: may be included in the **encrypted-data**, rather than the **signed-data**, if confidentiality of the **message-security-label** is required;
- a **content-integrity-key**: a symmetric-encryption-key used with the **content-integrity-algorithm-identifier** by the originator of the message to compute the **content-integrity-check**, and by the recipient to validate the **content-integrity-check**;
- a **message-sequence-number**: as defined for the **signed-data** above, but may be included in the **encrypted-data** instead if confidentiality of the sequence is required.

If the **message-token** is an **asymmetric-token** and the **signed-data** of the **message-token** includes the **content-integrity-check**, the **message-token** provides for non-repudiation-of-origin of message **content** (the non-repudiation of origin element-of-service, as defined in Recommendation X.400). If the **signed-data** of the **message-token** includes both the **content-integrity-check** and the **message-security-label**, the **message-token** provides proof of association between the **message-security-label** and the message **content**.

8.2.1.1.1.27 *Content-confidentiality-algorithm-identifier*

This argument contains an **algorithm-identifier**, which identifies the algorithm used by the originator of the message to encrypt the message **content** (to provide the content confidentiality element-of-service as defined in Recommendation X.400). It may be generated by the originator of the message.

The algorithm may be used by the recipient(s) of the message to decrypt the message **content**.

The content-confidentiality altorithm may be either a symmetric- or an asymmetric-encryption-algorithm.

If a symmetric-encryption-algorithm is used, the **content-confidentiality-key** used by the originator to encrypt the message **content**, and which the recipient may use to decrypt the message **content**, may be derived from the **message-token** sent with the message. Alternatively, **content-confidentiality-key** may be distributed by some other means.

If an asymmetric-encryption-algorithm is used, the intended-recipient's public-asymmetric-encryption-key may be used by the originator of the message to encrypt the message **content**. The recipient may use the recipient's secret-asymmetric-encryption-key to decrypt the message **content**. Note that if an asymmetric-encryption-algorithm is used, the message can only be addressed to a single recipient, or to a set of recipients which share the same asymmetric-encryption-key pair.

8.2.1.1.1.28 *Content-integrity-check*

This argument provides the recipient(s) of the message with a means of validating that the message **content** has not been modified (to provide the content integrity element-of-service as defined in Recommendation X.400). It may be generated by the originator of the message. A different value of the argument may be specified for each recipient of the message.

The **content-integrity-check** enables content-integrity to be validated on a per-recipient basis using either a symmetric- or an asymmetric-encryption-algorithm. Note that the **message-origin-authentication-check** provides a means of validating content-integrity on a per-message basis using an asymmetric-encryption-algorithm.

The **content-integrity-check** may be included in the **signed-data** or the **encrypted-data** of the **message-token** to provide for non-repudiation-of-origin of the message **content**, and proof of association between the **message-security-label** and the message **content**.

The **content-integrity-check** is computed using the algorithm identified by the **content-integrity-algorithm-identifier** (an **algorithm-identifier**).

The **content-integrity-check** contains the **content-integrity-algorithm-identifier**, and an encrypted function (e.g., a compressed or hashed version) of the **content-integrity-algorithm-identifier** and the message **content**. Note that the **content-integrity-check** is computed using the clear (i.e. unencrypted) message **content**.

The content-integrity-algorithm may be either a symmetric- or an asymmetric-encryption-algorithm. Note that the use of a symmetric-encryption algorithm may permit simultaneous compression and encryption of the message **content**.

If a symmetric-encryption-algorithm is used, the **content-integrity-key** used to compute the **content-integrity-check**, and which the recipient may use to validate the **content-integrity-check**, may be derived from the **message-token** sent with the message. Alternatively, the **content-integrity-key** may be distributed by some other means.

If an asymmetric-encryption-algorithm is used, the originator's secret-asymmetric-encryption-key may be used by the originator of the message to compute the **content-integrity-check**. The recipient may use the originator's public-asymmetric-encryption-key (**subject-public-key**) derived from the **originator-certificate** to validate the **content-integrity-check**.

8.2.1.1.1.29 *Message-origin-authentication-check*

This argument provides the recipient(s) of the message, and any MTA through which the message is transferred, with a means of authenticating the origin of the message (to provide the Message Origin Authentication element-of-service as defined in Recommendation X.400). It may be generated by the originator of the message.

The **message-origin-authentication-check** provides proof of the origin of the message (message origin authentication), assurance that the message **content** has not been modified (the content integrity element-of-service as defined in Recommendation X.400), and proof of association between the **message-security-label** and the message.

The **message-origin-authentication-check** is computed using the algorithm (asymmetric-encryption-algorithm and hash-function) identified by the **message-origin-authentication-algorithm-identifier** (an **algorithm-identifier**).

The **message-origin-authentication-check** contains the **message-origin-authentication-algorithm-identifier**, and an asymmetrically encrypted, hashed version of the **message-origin-authentication-algorithm-identifier**, the message **content**, the **content-identifier** and the **message-security-label**. Optional components are included in the **message-origin-authentication-check** if they are present in the message.

If content-confidentiality (see § 8.2.1.1.1.27) is also used, the **message-origin-authentication-check** is computed using the encrypted version of the message **content** (to allow the **message-origin-authentication-check** to be validated by other than the intended-recipient (e.g. by an MTA) without compromising the confidentiality of the message **content**). Note that if the clear (i.e. unencrypted) version of the message **content** is used to compute the **message-origin-authentication-check**, the **message-origin-authentication-check** provides for both message-origin authentication and non-repudiation of origin of the message **content** (a signature), as defined in Recommendation X.400. If, however, the encrypted version of the message **content** is used, the **message-origin-authentication-check** provides for message-origin authentication, but not for non-repudiation of origin of the message **content**.

The **message-origin-authentication-check** may be computed by the originator of the message using the originator's secret-asymmetric-encryption-key. The **message-origin-authentication-check** may be validated by the recipient(s) of the message, and any MTA through which the message is transferred, using the public-asymmetric-encryption-key **(subject-public-key**) of the originator of the message derived from the **originator-certificate**.

Future version of this Recommendation may define other forms of **message-origin-authentication-check** (e.g., based on symmetric-encryption-techniques) which may be used by MTAs through which the message is transferred to authenticate the origin of the message.

8.2.1.1.1.30 *Message-security label*

This argument associates a **security-label** with the message (or probe). It may be generated by the originator of the message (or probe), in line with the security-policy in force.

The **message-security label** of a report shall be the same as the **message-security label** of the subject-message (or subject-probe).

If **security-labels** are assigned to MTS-users, MTAs and other objects in the MHS, the handling by those objects of messages, probes and reports bearing **message-security-labels** may be determined by the security-policy in force. If **security-labels** are not assigned to MTS-users, MTAs and other objects in the MHS, the handling by those objects of messages, probes and reports bearing **message-security-labels** may be discretionary.

If **security-contexts** are established between the originator and an MTA (the originating-MTA) of the MTS (see §§ 8.1.1.1.1.3 and 8.2.1.4.1.5), the **message-security-label** that the originator may assign to a message (or probe) may be determined by the **security-context** (submission-security-context), in line with the security-policy in force. If

security-contexts are not established between the originator and the originating-MTA, the assignment of a **message-security-label** to a message (or probe) may be at the discretion of the originator.

If **security-contexts** are established between two MTAs (see § 12.1.1.1.1.3), the transfer of messages, probes or reports between the MTAs may be determined by the **message-security-labels** of the messages, probes or reports, and the **security-context**, in line with the security-policy in force. If **security-contexts** are not established between the MTAs, the transfer of messages, probes and reports may be at the discretion of the sender.

If **security-contexts** are established between an MTS-user and an MTA (the delivering-MTA) of the MTS (see §§ 8.1.1.1.1.3 and 8.3.1.3.1.7), the delivery of messages and reports may be determined by the **message-security-labels** of the messages and reports, and the **security-context** (delivery-security-context), in line with the security-policy in force. If the **message-security-label** of a message or report is allowed by the registered **user-security-labels** of the recipient, but disallowed by the recipient's current **security-context** (delivery-security-context), then the delivering-MTA may hold-for-delivery. If **security-contexts** are not established between the MTS-user and the delivering-MTA, the delivery of messages and reports may be at the discretion of the delivering-MTA.

8.2.1.1.1.31 *Proof-of-submission-request*

This argument indicates whether or not the originator of the message requires **proof-of-submission** (to provide the proof of submission element-of-service) as defined in Recommendation X.400) of the message to the MTS. It may be generated by the originator of the message.

This argument may have one of the following values: **proof-of-submission-requested** or **proof-of-submission-not-requested**.

In the absence of this argument, the default **proof-of-submission-not-requested** shall be assumed.

8.2.1.1.1.32 *Proof-of-delivery-request*

This argument indicates whether or not the originator of the message requires **proof-of-delivery** (to provide the proof of delivery element-of-service as defined in Recommendation X.400) of the message to the recipient. It may be generated by the originator of the message. A different value of this argument may be specified for each recipient of the message.

This argument may have one of the following values: **proof-of-delivery-requested** or **proof-of-delivery-not-requested**.

In the absence of this argument, the default **proof-of-delivery-not-requested** shall be assumed.

8.2.1.1.1.33 *Original-encoded-information-types*

This argument identifies the original **encoded-information-types** of the message **content**. It may be generated by the originator of the message.

The absence of this argument indicates that the **original-encoded-information-types** of the message **content** are unspecified.

8.2.1.1.1.34 *Content-type*

This argument identifies the type of the content of the message. It shall be generated by the originator of the message. The **content-type** shall be either built-in or extended.

A built-in **content-type** may have one of the following values:

– **unidentified:** denotes a **content-type** unidentified and unconstrained; the of this **unidentified content-type** is by bilateral agreement between MTS-users;

– **external:** denotes a **content-type** which is reserved for use when interworking between 1988 systems and 1984 systems (see Recommendation X.419);

– **interpersonal-messaging-1984**: identifies the **interpersonal-messaging-1984 content-type** defined in Recommendation X.420;

– **interpersonal-messaging-1988**: identifies the **interpersonal-messaging-1988 content-type** defined in Recommendation X.420;

– one specific value of an extended **content-type** which has been defined by this Recommendation is **inner-envelope**: an extended **content-type** that is itself a message (envelope and content), for forwarding by the recipient named on the outer-envelope to those named on the inner-envelope. The type of the **content** OCTET STRING in an **MTS-APDU**, encoded using the Basic Encoding Rules of ASN.1. [Note that the inner-envelope and using the security arguments (see §§ 8.2.1.1.1.25 to 8.2.1.1.1.32).]

Other standardized **extended content-types** may be defined by future versions of this Recommendation. Other values of this argument may be used by bilateral agreement between MTS-users.

8.2.1.1.1.35 *Content-identifier*

This argument contains an identifier for the **content** of the message. It may be generated by the originator of the message.

The **content-identifier** may be delivered to the recipient(s) of the message, and is returned to the originator with any report(s). This argument is not altered by the MTS.

8.2.1.1.1.36 *Content-correlator*

This argument contains information to enable correlation of the **content** of the message by the originator of the message. It may be generated by the originator of the message.

The **content-correlator** is not delivered to the recipient(s) of the message, but is returned to the originator with any report(s). This argument is not altered by the MTS.

8.2.1.1.1.37 *Content*

This argument contains the information the message is intended to convey to the recipient(s). It shall be generated by the originator of the message.

Except when conversion is performed, the **content** of the message is not modified by the MTS, but rather is passed transparently through it.

The **content** may be encrypted to ensure its confidentiality (see § 8.2.1.1.1.27).

The **content** may be an **external-content**. The **content** is an **external-content** when the **content-type** argument has the value **external**. When the **content** is an **external-content**, the **external-content-type** is specified by the object identifier of the **external-content**. An **external-content** may be used to convey an **inner-envelope** (see § 8.2.1.1.1.34), or for interworking between 1988 systems and 1984 systems (see Recommendation X.419).

8.2.1.1.2 *Results*

Table 5/X.411 lists the results of the message-submission abstract-operation, and for each result qualifies its presence and identifies the clause in which the result is defined.

TABLE 5/X.411

Message-submission results

Result	Presence	Clause
Message-submission-identifier	M	8.2.1.1.2.1
Message-submission-time	M	8.2.1.1.2.2
Originating-MTA-certificate	O	8.2.1.1.2.3
Proof-of-submission	C	8.2.1.1.2.4
Content-identifier	C	8.2.1.1.1.35

8.2.1.1.2.1 *Message-submission-identifier*

This result contains an **MTS-identifier** that uniquely and unambiguously identifies the message-submission. It shall be generated by the MTS.

The MTS provides the **message-submission-identifier** when notifying the MTS-user, via the report-delivery abstract-operation, of the delivery or non-delivery of the message.

The MTS-user provides the **message-submission-identifier** when cancelling, via the cancel-deferred-delivery abstract-operation, a message whose delivery it deferred.

8.2.1.1.2.2 *Message-submission-time*

This result indicates the **time** at which the MTS accepts responsibility for the message. It shall be generated by the MTS.

8.2.1.1.2.3 *Originating-MTA-certificate*

This result contains the **certificate** of the MTA to which the message has been submitted (the originating-MTA). It shall be generated by a trusted source (e.g. a certification-authority), and may be supplied by the originating-MTA, if the originator of the message requested **proof-of-submission** (see § 8.2.1.1.1.31) and an asymmetric-encryption-algorithm is used to compute the **proof-of-submission**.

The **originating-MTA-certificate** may be used to convey to the originator of the message a verified copy of the public-asymmetric-encryption-key (**subject-public-key**) of the originating MTA.

The originating-MTA's public-asymmetric-encryption-key may be used by the originator of the message to validate the **proof-of-submission**.

8.2.1.1.2.4 *Proof-of-submission*

This result provides the originator of the message with proof of submission of the message to the MTS (to provide the proof of submission element-of-service as defined in Recommendation X.400). Depending on the encryption-algorithm used and the security policy in force, this argument may also provide the non-repudiation of submission element-of-service (as defined in Recommendation X.400). It shall be generated by the originating-MTA of the MTS, if the originator of the message requested **proof-of-submission** (see § 8.2.1.1.1.31).

The **proof-of-submission** is computed using the algorithm identified by the **proof-of-submission-algorithm-identifier** (an **algorithm-identifier**.)

The **proof-of-submission** contains the **proof-of-submission-algorithm-identifier**, and an encrypted function (e.g., a compressed or hashed version) of the **proof-of-submission-algorithm-identifier**, the arguments of the submitted message (see § 8.2.1.1.1), and the **message-submission-identifier** and **message-submission-time**. Optional components are included in the **proof-of-submission** if they are present in the message.

Note that receipt of this result provides the originator of the message with proof of submission of the message. Non-receipt of this result provides neither proof of submission nor proof of non-submission (unless a secure link and trusted functionality are employed).

If an asymmetric-encryption-algorithm is used, the **proof-of-submission** may be computed by the originating-MTA using the originating-MTA's secret-asymmetric-encryption-key. The originator of the message may validate the **proof-of-submission** using the originating-MTA's public-asymmetric-encryption-key (**subject-public-key**) derived from the **originating-MTA-certificate**. An asymmetric **proof-of-submission** may also provide for non-repudiation of submission.

If a symmetric-encryption-algorithm is used, the summetric-encryption-key that the originating-MTA used to compute the **proof-of-submission**, and which the originator may use to validate the **proof-of-submission**, may be derived from the **bind-tokens** (see §§ 8.1.1.1.1.3 and 8.1.1.1.2.2) exchanged when the association was initiated. Alternatively, the symmetric-encryption-key used for **proof-of-submission** may be exchanged by some other means. Note that if a symmetric-encryption-algorithm is used then the **proof-of-submission** can only support non-repudiation of submission if the security-policy in force provides for the involvement of a third party acting as a notary.

8.2.1.1.3 *Abstract-errors*

Table 6/X.411 lists the abstract-errorsthat may disrupt the message-submission abstract-operation, and for each abstract-error identifies the clause in which the abstract-error is defined.

TABLE 6/X.411

Message-submission abstract-errors

Abstract-error	Clause
Submission-control-violated	8.2.2.1
Element-of-service-not-subscribed	8.2.2.2
Originator-invalid	8.2.2.4
Recipient-improperly-specified	8.2.2.5
Inconsistent-request	8.2.2.7
Security-error	8.2.2.8
Unsupported-critical-function	8.2.2.9
Remote-bind-error	8.2.2.10

8.2.1.2 *Probe-submission*

The probe-submission abstract-operation enables an MTS-user to submit a probe in order to determine whether or not a message (the subject-message) could be transferred and delivered to one or more recipient MTS-users if it were to be submitted.

Success of a probe does not guarantee that a subsequently submitted message can actually be delivered but rather that, currently, the recipient is valid and the message would encounter no major obstacles to delivery.

For any **recipient-names** that denote a DL, the probe-submission abstract-operation determines whether expansion of the specified DL (but not of any nested DLs) would occur.

For any **recipient-names** for which redirection would occur, the probe-submission abstract-operation determines whether the message could be transferred and delivered to the alternate-recipient.

The MTS-user supplies most of the arguments used for message-submission and the length of the content of the subject-message. The probe-submission abstract-operation does not culminate in delivery to the intended recipients of the subject-message, but establishes whether or not the message-submission abstract-operation would be likely to do so.

The successful completion of the abstract-operation signifies that the MTS has agreed to undertake the probe (but not that it has yet performed the probe).

The disruption of the abstract-operation by an abstract-error indicates that the MTS cannot undertake the probe.

8.2.1.2.1 *Arguments*

Table 7/X.411 lists the arguments of the probe-submission abstract-operation, and for each argument qualifies its presence and identifies the clause in which the argument is defined.

8.2.1.2.1.1 *Probe-origin-authentication-check*

This argument provides any MTA through which the probe is transferred, with a means of authenticating the origin of the probe (to provide the probe origin authentication element-of service as defined in Recommendation X.400). It may be generated by the originator of the probe.

The **probe-origin-authentication-check** provides proof of the origin of the probe (Probe Origin Authentication), and proof of association between the **message-security-label** and the **content-identifier** of the subject-message.

The **probe-origin-authentication-check** is computed using the algorithm identified by the **probe-origin-authentication-algorithm-identifier** (an **algorithm-identifier**).

TABLE 7/X.411

Probe-submission arguments

Argument	Presence	Clause
Originator argument		
Originator-name	M	8.2.1.1.1.1
Recipient arguments		
Recipient-name	M	8.2.1.1.1.2
Alternate-recipient-allowed	O	8.2.1.1.1.3
Recipient-reassignment-prohibited	O	8.2.1.1.1.4
Originator-requested-alternate-recipient	O	8.2.1.1.1.5
DL-expansion-prohibited	O	8.2.1.1.1.6
Conversion arguments		
Implicit-conversion-prohibited	O	8.2.1.1.1.9
Conversion-with-loss-prohibited	O	8.2.1.1.1.10
Explicit-conversion	O	8.2.1.1.1.11
Delivery method argument		
Requested-delivery-method	O	8.2.1.1.1.14
Physical delivery argument		
Physical-rendition-attributes	O	8.2.1.1.1.20
Report request argument		
Originator-report-request	M	8.2.1.1.1.22
Security arguments		
Originator-certificate	O	8.2.1.1.1.25
Probe-origin-authentication-check	O	8.2.1.2.1.1
Message-security-label	O	8.2.1.1.1.30
Content arguments		
Original-encoded-information-types	O	8.2.1.1.1.33
Content-type	M	8.2.1.1.1.34
Content-identifier	O	8.2.1.1.1.35
Content-correlator	O	8.2.1.1.1.36
Content-length	O	8.2.1.2.1.2

The **probe-origin-authentication-check** contains the **probe-origin-authentication-algorithm-identifier**, and an asymmetrically encrypted, hashed version of the **probe-origin-authentication-algorithm-identifier**, and the **content-identifier** and **message-security-label** of the subject-message. Optional components are included in the **probe-origin-authentication-check** if they are present in the probe.

Future versions of this Recommendation may define other forms of **probe-origin-authentication-check** (e.g., based on symmetric-encryption-techniques) which may be used by MTAs through which the probe is transferred to authenticate the origin of the probe.

8.2.1.2.1.2 *Content-length*

This argument specifies the length, in octets, of the **content** of the subject-message. It may be generated by the originator of the probe.

8.2.1.2.2 *Results*

Table 8/X.411 lists the results of the probe-submission abstract-operation, and for each result qualifies its presence and identifies the clause in which the result is defined.

TABLE 8/X.411

Probe-submission results

Result	Presence	Clause
Probe-submission-identifier	M	8.2.1.2.2.1
Probe-submission-time	M	8.2.1.2.2.2
Content-identifier	C	8.2.1.1.1.35

8.2.1.2.2.1 *Probe-submission-identifier*

This result contains an **MTS-identifier** that uniquely and unambiguously identifies the probe-submission. It shall be generated by the MTS.

The MTS provides the **probe-submission-identifier** when notifying the MTS-user, via the report-delivery abstract-operation, of its ability or otherwise to deliver the subject-message.

8.2.1.2.2.2 *Probe-submission-time*

This result indicates the **time** at which the MTS agreed to undertake the probe. It shall be generated by the MTS.

8.2.1.2.3 *Abstract-errors*

Table 9/X.411 lists the abstract-errors that may disrupt the probe-submission abstract-operation, and for each abstract-error identifies the clause in which the abstract-error is defined.

TABLE 9/X.411

Probe-submission abstract-errors

Abstract-error	Clause
Submission-control-violated	8.2.2.1
Element-of-service-not-subscribed	8.2.2.2
Originator-invalid	8.2.2.4
Recipient-improperly-specified	8.2.2.5
Inconsistent-request	8.2.2.7
Security-error	8.2.2.8
Unsupported-critical-function	8.2.2.9
Remote-bind-error	8.2.2.10

8.2.1.3 *Cancel-deferred-delivery*

The cancel-deferred-delivery abstract-operation enables an MTS-user to abort the deferred-delivery of a message previsouly submitted via the message-submission abstract-operation.

The MTS-user identifies the message whose delivery is to be cancelled by means of the **message-submission-identifier** returned by the MTS as a result of the previous invocation of the message-submission abstract-operation.

The successful completion of the abstract-operation signifies that the MTS has cancelled the deferred-delivery of the message.

The disruption of the abstract-operation by an abstrar-error indicates that the deferred-delivery cannot be cancelled. The deferred-delivery of a message cannot be cancelled if the message has already been progressed for delivery and/or transfer within the MTS. The MTS may refuse to cancel the deferred-delivery of a message if the MTS provided the originator of the message with **proof-of-submission**.

8.2.1.3.1 *Arguments*

Table 10/X.411 lists the arguments of the cancel-deferred-delivery abstract-operation, and for each argument qualifies its presence and identifies the clause in which the argument is defined.

TABLE 10/X.411

Cancel-deferred-delivery arguments

Argument	Presence	Clause
Submission argument		
Message-submission-identifier	M	8.2.1.3.1.1

8.2.1.3.1.1 *Message-submission-identifier*

This argument contains the **message-submission-identifier** of the message whose deferred-delivery is to be cancelled. It shall be suppled by the MTS-user.

The **message-submission-identifier** (an **MTS-identifier**) is that returned by the MTS as a result of a previous invocation of the message-submission abstract-operation (see § 8.2.1.1.2.1), when the message was submitted for deferred-delivery.

8.2.1.3.2 *Results*

The cancel-deferred-delivery abstract-operation returns an emply result as indication of success.

8.2.1.3.3 *Abstract-errors*

Table 11/X.411 lists the abstract-errors that may disrupt the cancel-deferred-delivery abstract-operation, and for each abstract-error identifies the clause in which the abstract-error is defined.

TABLE 11/X.411

Cancel-deferred-delivery abstract-errors

Abstract-error	Clause
Deferred-delivery-cancellation-rejected	8.2.2.3
Message-submission-identifier-invalid	8.2.2.6
Remote-bind-error	8.2.2.10

8.2.1.4 *Submission-control*

The submission-control abstract-operation enables the MTS to temporarily limit the submission-port abstract-operations that the MTS-user may invoke, and the messages that the MTS-user may submit to the MTS via the Message-submission abstract-operation.

The MTS-user should hold until a later time, rather than abandon, abstract-operations and messages presently forbidden.

The successful completion of the abstract-operation signifies that the specified controls are now in force. These controls supersede any previously in force, and remain in effect until the association is released or the MTS re-invokes the submission-control abstract-operation.

The abstract-operation returns and indication of any abstract-operations that the MTS-user would invoke, or any message types that the MTS-user would submit, were it not for the prevailing controls.

8.2.1.4.1 *Arguments*

Table 12/X.411 lists the arguments of the submission-control abstract-operation, and for each argument qualifies its presence and identifies the clause in which the argument is defined.

TABLE 12/X.411

Submission-control arguments

Argument	Presence	Clause
Submission control arguments		
Restrict	O	8.2.1.4.1.1
Permissible-operations	O	8.2.1.4.1.2
Permissible-lowest-priority	O	8.2.1.4.1.3
Permissible-maximum-content-length	O	8.2.1.4.1.4
Permissible-security-context	O	8.2.1.4.1.5

8.2.1.4.1.1 *Restrict*

This argument indicates whether the controls on submission-port abstract-operations are to be updates or removed. It may be generated by the MTS.

This argument may have one of the following values:

– **update**: the other arguments update the prevailing controls;

– **remove**: all controls are to be removed; the other arguments are to be ignored.

In the absence of this argument, the defauld **update** shall be assumed.

8.2.1.4.1.2 *Permissible-operations*

This argument indicates the abstract-operations that the MTS-user may invoke on the MTS.

This argument may have the value **allowed** or **prohibited** for each of the following:

– **message-submission**: the MTS-user may/may not invoke the message-submission abstract-operation; and

– **probe-submission**: the MTS-user may/may not invoke the probe-submission abstract-operation.

Other submission-port abstract-operations are not subject to controls, and may be invoked at any time.

In the absence of this argument, the abstract-operation that the MTS-user may invoke on the MTS are unchanged. If no previous controls are in force, the MTS-user may invoke both the message-submission abstract-operation and the probe-submission abstract-operation.

8.2.1.4.1.3 *Permissible-lowest-priority*

This argument contains the **priority** of the lowest priority message that the MTS-user shall submit to the MTS via the message-submission abstract-operation. It may be generated by the MTS.

This argument may have one of the following values of the **priority** argument of the message-submission abstract-operation: **normal, non-urgent** or **urgent**

In the absence of this argument, the **priority** of the lowest priority message that the MTS-user shall submit to the MTS is unchanged. If no previous controls are in force, the MTS-user may submit messages of any priority.

8.2.1.4.1.4 *Permissible-maximum-content-length*

This argument contains the **content-lenght**, in octets, of the longest-content message that the MTS-user shall submit to the MTS via the message-submission abstract-operation. It may be generated by the MTS.

In the absence of this argument, the **permissible-maximum-content-length** of a message that the MTS-user may submit to the MTS is unchanged. If no previous controls are in force, the content length is not explicitly limited.

8.2.1.4.1.5 *Permissible-security-context*

This argument temporarily limits the sensitivity of submission-port abstract-operations (submission-security-context) that the MTS-user may invoke on the MTS. It is a temporary restriction of the **security-context** established when the association was initiated (see § 8.1.1.1.1.3). It may be generated by the MTS.

The **permissible-security-context** comprises one or more **security-labels** from the set of **security-labels** established as the **security-context** when the association was established.

In the absence of this argument, the **security-context** of submission-port abstract-operations is unchanged.

8.2.1.4.2 *Results*

Table 13/X.411 lists the results of the submission-control abstract-operation, and for each result qualifies its presence and identifies the clause in which the result is defined.

TABLE 13/X.411

Submission-control results

Result	Presence	Clause
"Waititing" results		
Waiting-operations	O	8.2.1.4.2.1
Waiting-messages	O	8.2.1.4.2.2
Waiting-encoded-information-types	O	8.2.1.4.2.3
Waiting-content-types	O	8.2.1.4.2.4

8.2.1.4.2.1 *Waiting-operations*

This result indicates the abstract-operations being held by the MTS-user, and that the MTS-user would invoke on the MTS if it were not for the prevailing controls. It may be generated by the MTS-user.

This result may have the value **holding** or **not-holding** for each of the following:

– **message-submission**: the MTS user is/is not holding messages, and would invoke the message-submission abstract-operation on the MTS if it were not for the prevailing controls; and

– **probe-submission**: the MTS-user is/is not holding probes, and would invoke the probe-submission abstract-operation on the MTS if it were not for the prevailing controls.

In the absence of this result, it may be assumed that the MTS-user is not holding any messages or probes for submission to the MTS due to the prevailing controls.

8.2.1.4.2.2 Waiting-messages

This result indicates the kind of messages the MTS-user is holding for submission to the MTS, and would submit via the message-submission abstract-operation, if it were not for the prevailing controls. It may be generated by the MTS-user.

This result may have one or more of the following values:

– **long-content**: the MTS-user has messages held for submission to the MTS which exceed the **permissible-maximum-content-length** control currently in force;

– **low-priority**: the MTS-user has messages held for submission to the MTS of a lower **priority** than the **permissible-lowest-priority** control currently in force;

– **other-security-labels**: the MTS-user has messages held for submission to the MTS bearing **message-security-labels** other than those permitted by the current security-context.

In the absence of this result, it may be assumed that the MTS-user is not holding any messages or probes for submission to the MTS due to the **permissible-maximum-content-length**, **permissible-lowest-priority** or **permissible-security-context** controls currently in force.

8.2.1.4.2.3 *Waiting-encoded-information-types*

This result indicates the **encoded-information-types in the content** of any messages held by the MTS-user for submission to the MTS due to prevailing controls. It may be generated by the MTS-user.

In the absence of this result, the **encoded-information-types** of any messages held by the MTS-user for submission to the MTS are **unspecified**.

8.2.1.4.2.4 *Waiting-content-types*

This result indicates the **content-types** of any messages held by the MTS-user for submission to the MTS due to prevailing controls. It may be generated by the MTS-user.

In the absence of this result, the **content-types** of any messages held by the MTS-user for submission to the MTS are unspecified.

8.2.1.4.3 *Abstract-errors*

Table 14/X.411 lists the abstract-errors that may disrupt the submission-control abstract-operation, and for each abstract-error identifies the clause in which the abstract-error is defined.

TABLE 14/X.411

Submission-control abstract-errors

Abstract-error	Clause
Security-error	8.2.2.8
Remote-bind-error	8.2.2.10

8.2.2 *Abstract-errors*

This section defines the following submission-port abstract-errors:

a) submission-control-violated

b) element-of-service-not-subscribed

c) deferred-delivery-cancellation-rejected

d) originator-invalid

e) recipient-improperly-specified

f) message-submission-identifier-invalid

g) inconsistent-request

h) security-error

i) unsupported-critical-function

j) remote-bind-error.

8.2.2.1 *Submission-control-violated*

The submission-control-violated abstract-error reports the violation by the MTS-user of a control on submission-port services imposed by the MTS via the submission-control service.

The submission-control-violated abstract-error has no parameters.

8.2.2.2 *Element-of-service-not-subscribed*

The element-of-service-not-subscribed service reports that the requested abstract-operation cannot be provided by the MTS because the MTS-user has not subscribed to one of the elements-of-service the request requires.

The element-of-service-not-subscribed abstract-error has no parameters.

8.2.2.3 *Deferred-delivery-cancellation-rejected*

The deferred-delivery-cancellation-rejected abstract-error reports that the MTS cannot cancel the deferred-delivery of a message, either because the message has already been progressed for transfer and/or delivery, or because the MTS had provided the originator with **proof-of-submission**.

The deferred-delivery-cancellation-rejected abstract-error has no parameters.

8.2.2.4 *Originator-invalid*

The originator-invalid abstract-error reports that the message or probe cannot be submitted because the originator is incorrectly identified.

The originator-invalid abstract-error has no parameters.

8.2.2.5 *Recipient-improperly-specified*

The recipient-improperly-specified abstract-error reports that the message or probe cannot be submitted because one or more recipients are improperly specified.

The recipient-improperly-specified abstract-error has the following parameters, generated by the MTS.

– **improperly-specified-recipients**: the improperly specified **recipient-name(s)**.

8.2.2.6 *Message-submission-identifier-invalid*

The message-submission-identifier-invalid abstract-error reports that the deferred-delivery of a message cannot be cancelled because the specified **message-submission-identifier** is invalid.

The message-submission-identifier-invalid abstract-error has no parameters.

8.2.2.7 *Inconsistent-request*

The inconsistent-request abstract-error reports that the requested abstract-operation cannot be provided by the MTS because the MTS-user has made an inconsistent request.

The inconsistent-request abstract-error has no parameters.

8.2.2.8 *Security-error*

The security-error abstract-error reports that the requested abstract-operation could not be provided by the MTS because it would violate the security-policy in force.

The security-error abstract-error has the following parameters, generated by the MTS:

– **security-problem**: an identifier for the cause of the violation of the security-policy.

8.2.2.9 *Unsupported-critical-function*

The unsupported-critical-function abstract-error reports that an argument of the abstract-operation was marked as **critical-for-submission** (see § 9.1) but is unsupported by the MTS.

The unsupported-critical-function abstract-error has no parameters.

8.2.2.10 *Remote-bind-error*

The remote-bind-error abstract-error reports that the requested abstract-operation cannot be provided by the MS because the MS is unable to bind to the MTS. Note that this abstract-error only occurs on indirect submission to the MTS via an MS.

The remote-bind-error abstract-error has no parameters.

8.3 *Delivery port*

This paragraph defines the abstract-operations and abstract-errors which occur at a delivery-port.

8.3.1 *Abstract-operations*

This clause defines the following delivery-port abstract-operations:

a) message-delivery

b) report-delivery

c) delivery-control.

8.3.1.1 *Message-delivery*

The message-delivery abstract-operation enables the MTS to deliver a message to an MTS-user.

The MTS-user shall not refuse delivery of a message unless the delivery would violate the delivery-control restrictions then in force.

8.3.1.1.1 *Arguments*

Table 15/X.411 lists the arguments of the message-delivery abstract-operation, and for each argument qualifies its presence and identifies the clause in which the argument is defined.

8.3.1.1.1.1 *Message-delivery-identifier*

This argument contains an **MTS-identifer** that distinguishes the message from all other messages at the delivery-port. It shall be generated by the MTS, and shall have the same value as the **message-submission-identifier** supplied to the originator of the message when the message was submitted.

8.3.1.1.1.2 *Message-delivery-time*

This argument contains the **time** at which delivery occurs and at which the MTS is relinquishing responsibility for the message. It shall be generated by the MTS.

In the case of physical delivery, this argument indicates the **time** at which the PDAU has taken responsibility for printing and further delivery of the message.

The value of this argument shall be the same as the value of the **message-delivery-time** argument reported to the originator of the message (see § 8.3.1.2.1.8) in a delivery-report.

8.3.1.1.1.3 *This-recipient-name*

This argument contains the **OR-name** of the recipient to whom the message is being delivered. It shall be generated by the MTS.

The value of this argument shall be the same as the value of the **actual-recipient-name** argument reported to the originator of the message (see § 8.3.1.2.1.2) in a delivery-report.

The **this-recipient-name** contains the **OR-name** of the individual recipient, it shall not contain the **OR-name** of a DL.

The **OR-name** of the intended-recipient (if different, and the message has been redirected) is contained in the **intended-recipient-name** argument.

8.3.1.1.1.4 *Intended-recipient-name*

This argument contains the **OR-name** of the intended-recipient of the message if the message has been redirected and the time at which the redirection was performed. It may be generated by the MTS. A different value of this argument may be present for each occasion the message was redirected.

This argument comprises an **originally-intended-recipient-name** and an **intended-recipient-name**. On the first occasion a message is redirected, both the **originally-intended-recipient-name** and the **intended-recipient-name** contain the **recipient-name** originally-specified by the originator of the message. Subsequent redirections cause further **recipient-names** to be appended to the list of **intended-recipient-names**.

The **intended-recipient-name** contains the **OR-name** of an individual or DL intended-recipient and the time at which the message was redirected to an alternate recipient.

TABLE 15/X.411

Message-delivery arguments

Argument	Presence	Clause
Delivery arguments		
Message-delivery-identifier	M	8.3.1.1.1.1
Message-delivery-time	M	8.3.1.1.1.2
Message-submission-time	M	8.2.1.1.2.2
Originator argument		
Originator-name	M	8.2.1.1.1.1
Recipient arguments		
This-recipient-name	M	8.3.1.1.1.3
Intended-recipient-name	C	8.3.1.1.1.4
Redirection-reason	C	8.3.1.1.1.5
Other-recipient-names	C	8.3.1.1.1.6
DL-expansion-history	C	8.3.1.1.1.7
Priority argument		
Priority	C	8.2.1.1.1.8
Conversion arguments		
Implicit-conversion-prohibited	C	8.2.1.1.1.9
Conversion-with-loss-prohibited	C	8.2.1.1.1.10
Converted-encoded-information-types	C	8.3.1.1.1.8
Delivery method argument		
Requested-delivery-method	C	8.2.1.1.1.14
Physical delivery arguments		
Physical-forwarding-prohibited	C	8.2.1.1.1.15
Physical-forwarding-address-request	C	8.2.1.1.1.16
Physical-delivery-modes	C	8.2.1.1.1.17
Registered-mail-type	C	8.2.1.1.1.18
Recipient-number-for-advice	C	8.2.1.1.1.19
Physical-rendition-attributes	C	8.2.1.1.1.20
Originator-return-address	C	8.2.1.1.1.21
Physical-delivery-report-request	C	8.2.1.1.1.24
Security arguments		
Originator-certificate	C	8.2.1.1.1.25
Message-token	C	8.2.1.1.1.26
Content-confidentiality-algorithm-identifier	C	8.2.1.1.1.27
Content-integrity-check	C	8.2.1.1.1.28
Message-origin-authentication-check	C	8.2.1.1.1.29
Message-security-label	C	8.2.1.1.1.30
Proof-of-delivery-request	C	8.2.1.1.1.32
Content arguments		
Original-encoded-information-types	C	8.2.1.1.1.33
Content-type	M	8.2.1.1.1.34
Content-identifier	C	8.2.1.1.1.35
Content	M	8.2.1.1.1.37

8.3.1.1.1.5 *Redirection-reason*

This argument indicates the reason the message has been redirected to an alternate-recipient. It shall be generated by the MTS on each occasion that redirection occurs. A different value of this argument may be present for each occasion the message is redirected.

This argument may have one of the following values:

– **recipient-assigned-alternate-recipient**: the intended-recipient of the message requested that the message be redirected to a **recipient-assigned-alternate-recipient**; the originator of the message did not prohibit recipient-reassignment (see § 8.2.1.1.1.4); the MTS redirected the message to the **recipient-assigned-alternate-recipient**;

– **originator-requested-alternate-recipient**: the message could not be delivered to the intended-recipient or **recipient-assigned-alternate-recipient** (if registered); the **originator-requested-alternate-recipient** argument identified an alternate-recipient requested by the originator of the message; the MTS redirected the message to the **originator-requested-alternate-recipient;**

– **recipient-MD-assigned-alternate-recipient**: the **recipient-name** argument did not identify a recipient MTS-user; the **alternate-recipient-allowed** argument generated by the originator of the message allowed delivery to an alternate-recipient; the MTS redirected the message to an alternate-recipient assigned by the recipient-MD to receive such messages.

8.3.1.1.1.6 *Other-recipient-names*

This argument contains the originally-specified **OR-names** of all recipients other than those identified by the **originally-intended-recipient-name** argument, if present, and the **this-recipient-name** argument, if the originator of the message requested disclosure of other recipients (with the **disclosure-of-recipients** argument of the message-submission abstract-operation). It may be generated by the MTS. A different value of this argument may be present for each originally-specified recipient other than the **this-recipient-name** to which the message is being delivered.

Each **other-recipient-name** contains the **OR-name** of an individual recipient or a DL.

8.3.1.1.1.7 *DL-expansion-history*

This argument contains the sequence of **OR-names** of any DLs which have been expanded to add recipients to the copy of the message delivered to the recipient and the time of each expansion. It shall be generated by the MTS if any DL-expansion has occured.

8.3.1.1.1.8 *Converted-encoded-information-types*

This argument identifies the **encoded-information-types** of the message **content** after conversion, if conversion took place. It may be generated by the MTS.

8.3.1.1.2 *Results*

Table 16/X.411 lists the results of the message-delivery abstract-operation, and for each result qualifies its presence and identifies the clause in which the result is defined.

TABLE 16/X.411

Message-delivery results

Result	Presence	Clause
Proof of delivery results		
Recipient-certificate	O	8.3.1.1.2.1
Proof-of-delivery	C	8.3.1.1.2.2

8.3.1.1.2.1 *Recipient-certificate*

This argument contains the **certificate** of the recipient of the message. It shall be generated by a trusted source (e.g. certification-authority), and may be supplied by the recipient of the message, if the originator of the message requested **proof-of-delivery** (see § 8.2.1.1.1.32) and an asymmetric-encryption-algorithm is used to compute the **proof-of-delivery**.

The **recipient-certificate** may be used to convey a verified copy of the public-asymmetric-encryption-key (**subject-public-key**) of the recipient of the message.

The recipient's public-asymmetric-encryption-key may be used by the originator of the message to validate the **proof-of-delivery**.

8.3.1.1.2.2 *Proof-of-delivery*

This argument provides the originator of the message with proof that the message has been delivered to the recipient (to provide the proof of delivery element-of-service as defined in Recommendation X.400). Depending on the encryption-algorithm used and the security-policy in force, this argument may also provide the non-repudiation of delivery element-of-service (as defined in Recommendation X.400). It shall be generated by the recipient of the message, if the originator of the message requested **proof-of-delivery** (see § 8.2.1.1.1.32).

The **proof-of-delivery** is computed using the algorith identified by the **proof-of-delivery-algorithm-identifier** (an **algorithm-identifier**).

The **proof-of-delivery** contains the **proof-of-delivery-algorithm-identifier**, and an encrypted function (e.g., a compressed or hashed version) of the **proof-of-delivery-algorithm-identifier**, the **delivery-time**, and the **this-recipient-name**, the **originally-intended-recipient-name**, the message **content**, the **content-identifier**, and the **message-security-label** of the delivered message. Optional components are included in the **proof-of-delivery** if they are present in the delivered message. Note that the **proof-of-delivery** is computed using the clear (i.e. unencrypted) message **content**.

Note that receipt of this argument provides the originator of the message with proof of delivery of the message to the recipient. Non-receipt of this argument provides neither proof of delivery nor proof of non-delivery (unless a secure route and trusted functionality are employed).

If an asymmetric-encryption-algorithm is used, the **proof-of-delivery** may be computed by the recipient of the message using the recipient's secret-asymmetric-encryption-key. The originator of the message may validate the **proof-of-delivery** using the recipient's public asymmetric-encryption-key (**subject-public-key**) derived from the **recipient-certificate**. An asymmetric **proof-of-delivery** may also provide for non-repudiation of delivery.

If a symmetric-algorithm is used, a symmetric-encryption-key is used by the recipient to compute the **proof-of-delivery**, and by the originator to validate the **proof-of-delivery**. Note that if a symmetric-encryption-algorithm is used then the **proof-of-delivery** can only provide non repudiation of delivery if the security-policy in force provides for the involvement of a third party acting as a notary. The means by which the symmetric-encryption-key is distributed is not currently defined by this Recommendation.

8.3.1.1.3 *Abstract-errors*

Table 17/X.411 lists the abstract-errors that may disrupt the message-delivery abstract-operation, and for each abstract-error identifies the clause in which the abstract-error is defined.

TABLE 17/X.411

Message-delivery abstract-errors

Abstract-error	Clause
Delivery-control-violated	8.3.2.1
Security-error	8.3.2.3
Unsupported-critical-function	8.3.2.4

8.3.1.2 Report-delivery

The **report-delivery** abstract-operation enables the MTS to acknowledge to the MTS-user one or more outcomes of a previous invocation of the message-submission or probe-submission abstract-operations.

For the message-submission abstract-operation, the report-delivery abstract-operation indicates the delivery or non-delivery of the submitted message to one or more recipients.

For the probe-submission abstract-operation, the report-delivery abstract-operation indicates whether or not a message could be delivered, or a DL-expansion could occur, if the message were to be submitted.

A single invocation of the message-submission or probe-submission abstract-operation may provoke several occurences of the report-delivery abstract-operation, each covering one or more intended recipients. A single occurence of the report-delivery abstract-operation may report on both delivery and non-delivery to different recipients.

An invocation of the message-submission or probe-submission abstract-operation by one MTS-user may provoke occurences of the report-delivery abstract-operation to another MTS-user, i.e., reports delivered to the owner of a DL.

The MTS-user shall not refuse to accept the delivery of a report unless the delivery of the report would violate the delivery-control restrictions then in force.

8.3.1.2.1 *Arguments*

Table 18/X.411 lists the arguments of the report-delivery abstract-operation, and for each argument qualifies its presence and identifies the clause in which the argument is defined.

8.3.1.2.1.1 *Subject-submission-identifier*

This argument contains the **message-submission-identifer** or the **probe-submission-identifier** of the subject of the report. It shall be supplied by the MTS.

8.3.1.2.1.2 *Actual-recipient-name*

This argument contains the **OR-name** of a recipient of the message. It shall be generated by the originator of the message, or by the MTS if the message has been redirected. A different value of this argument shall be specified for each recipient of the subject to which this report relates.

In the case of a delivery report, the **actual-recipient-name** is the name of the actual recipient of the message, and has the same value as the **this-recipient-name** argument of the delivered message. In the case of a non-delivery-report, the **actual-recipient-name** is the **OR-name** of the recipient to which the message was being directed when the reason for non-delivery was encountered.

The **actual-recipient-name** may be an originally-specified **recipient-name**, or the **OR-name** of an alternate-recipient if the message has been redirected. If the message has been redirected, the **OR-name** of the intended-recipient is contained in the **intended-recipient-name** argument.

The **actual-recipient-name** contains the **OR-name** of an individual recipient or DL.

8.3.1.2.1.3 *Originator-and-DL-expansion-history*

This argument contains a sequence of **OR-names** and associated times which document the history of the origin of the subject-message. This first **OR-name** in the sequence is the **OR-name** of the originator of the subject, and the remainder of the sequence is a sequence of **OR-names** of the DLs that have been expanded in directing the subject towards the recipient (the latter being the same as the **DL-expansion-history**). It shall be generated by the originating-MTA of the report if any DL-expansion has occurred on the subject.

The **originator-and-DL-expansion-history** contains the **OR-name** of the originator of the subject and each DL and the time at which the associated event occurred.

8.3.1.2.1.4 *Reporting-DL-name*

This argument contains the **OR-name** of the DL that forwarded the report to the owner of the DL. It shall be generated by a DL-expansion-point (an MTA) when forwarding a report to the owner of the DL, in line with the reporting-policy of the DL.

The **reporting-DL-name** contains the **OR-name** of the DL forwarding the report.

TABLE 18/X.411

Report-delivery arguments

Argument	Presence	Clause
Subject submission argument		
Subject-submission-identifier	M	8.3.1.2.1.1
Recipient arguments		
Actual-recipient-name	M	8.3.1.2.1.2
Intended-recipient-name	C	8.3.1.1.1.4
Redirection-reason	C	8.3.1.1.1.5
Originator-and-DL-expansion-history	C	8.3.1.2.1.3
Reporting-DL-name	C	8.3.1.2.1.4
Conversion arguments		
Converted-encoded-information-types	C	8.3.1.2.1.5
Supplementary information arguments		
Supplementary-information	C	8.3.1.2.1.6
Physical-forwarding-address	C	8.3.1.2.1.7
Delivery arguments		
Message-delivery-time	C	8.3.1.2.1.8
Type-of-MTS-user	C	8.3.1.2.1.9
Non-delivery arguments		
Non-delivery-reason-code	C	8.3.1.2.1.10
Non-delivery-diagnostic-code	C	8.3.1.2.1.11
Security arguments		
Recipient-certificate	C	8.3.1.1.2.1
Proof-of-delivery	C	8.3.1.1.2.2
Reporting-MTA-certificate	C	8.3.1.2.1.12
Report-origin-authentication-check	C	8.3.1.2.1.13
Message-security-label	C	8.2.1.1.1.30
Content arguments		
Original-encoded-information-types	C	8.2.1.1.1.33
Content-type	C	8.2.1.1.1.34
Content-identifier	C	8.2.1.1.1.35
Content-correlator	C	8.2.1.1.1.36
Returned-content	C	8.3.1.2.1.14

8.3.1.2.1.5 *Converted-encoded-information-types*

This argument identifies the **encoded-information-types** of the subject-message **content** after conversion, if conversion took place. For a report on a message, this argument indicates the actual **encoded-information-types** of the converted message **content**. For a report on a probe, this argument indicates the **encoded-information-types** the subject-message **content** would have contained after conversion, if the subject-message were to have been submitted. It may be generated by the MTS. A different value of this parameter may be specified for each recipient of the subject to which the report relates.

8.3.1.2.1.6 *Supplementary-information*

This argument may contain information supplied by the originator of the report, as a printable string. It may be generated by the originating-MTA of the report or an associated access-unit. A different value of this argument may be specified for each intended recipient of the subject to which the report relates.

Supplementary-information may be used by a Teletex-access-unit or a Teletex/Telex conversion facility. It may contain a received answer-back, Telex transmission duration, or note and received recorded message as a printable string.

Supplementary-information may also be used by other access-units, or by the originating-MTA of the report itself, to convey printable information to the originator of the message.

8.3.1.2.1.7 *Physical-forwarding-address*

This argument contains the new **postal-OR-address** of the physical-recipient of the message. It may be generated by the associated PDAU of the originating-MTA of the report, if the originator of the message requested the physical-forwarding-address of the recipient (see § 8.2.1.1.1.16). A different value of this argument may be specified for each intended recipient of the subject-message to which the report relates.

8.3.1.2.1.8 *Message-delivery-time*

This argument contains the **time** at which the subject-message was (or would have been) delivered to the recipient MTS-user. It shall be generated by the MTS if the message was (or would have been) successfully delivered. A different value of this argument may be specified for each intended-recipient of the subject to which the report relates.

In the case of physical delivery, this argument indicates the **time** at which the PDAU has taken responsibility for printing and further delivery of the message.

If the subject-message was delivered, the value of this argument shall be the same as the value of the **message-delivery-time** argument of the delivered message (see § 8.3.1.1.1.2).

8.3.1.2.1.9 *Type-of-MTS-user*

This argument indicates the type of recipient MTS-user to which the message was (or would have been) delivered. It shall be generated by the MTS if the message was (or would have been) successfully delivered. A different value of this argument may be specified for each intended-recipient of the subject to which the report relates.

This argument may have one of the following values:

– **public**: a UA owned by an Administration;

– **private**: a UA owned by other than an Administration;

– **ms**: a message-store;

– **DL**: a distribution-list;

– **PDAU**: a physical-delivery-access-unit (PDAU);

– **physical-recipient**: a physical-recipient of a PDS;

– **other**: an access-unit of another kind.

8.3.1.2.1.10 *Non-delivery-reason-code*

This argument contains a code indicating the reason the delivery of the subject-message failed (or, in the case of a probe, would have failed). It shall be generated by the MTS if the message was (or would have been) unsuccessfully delivered. A different value of this argument may be specified for each intended-recipient of the subject to which the report relates.

This argument may have one of the following values:

– **transfer-failure**: indicates that, while the MTS was attempting to deliver or probe delivery of the subject-message, some communication failure prevented it from doing so;

– **unable-to-transfer**: indicates that, due to some problem with the subject itself, the MTS could not deliver or probe delivery of the subject-message;

– **conversion-not-performed**: indicates that a conversion necessary for the delivery of the subject-message was (or would be) unable to be performed;

– **physical-rendition-not-performed**: indicates that the PDAU was unable to physically render the subject-message;

– **physical-delivery-not-performed**: indicates that the PDS was unable to physically deliver the subject-message;

– **restricted-delivery**: indicates that the recipient subscribes to the restricted-delivery element-of-service (as defined in Recommendation X.400) which prevented (or would prevent) the delivery of the subject-message;

– **directory-operation-unsuccessful**: indicates that the outcome of a required directory operation was unsuccessful.

Other **non-delivery-reason-codes** may be specified in future versions of this Recommendation.

Further information on the nature of the problem preventing delivery is contained in the **non-delivery-diagnostic-code** argument.

8.3.1.2.1.11 *Non-delivery-diagnostic-code*

This argument contains a code indicating the nature of the problem which caused delivery or probing of delivery of the subject-message to fail. The reason for failure is indicated by the **non-delivery-reason-code** argument. It may be generated by the MTS if the message was (or would have been) unsuccessfully delivered. A different value of this argument may be specified for each intended-recipient of the subject to which the report relates.

This argument may have one of the following values:

– **unrecognised-OR-name**: the **recipient-name** argument of the subject does not contain an **OR-name** recognised by the MTS;

– **ambiguous-OR-name**: the **recipient-name** argument of the subject identifies more than one potential recipient (i.e., is ambiguous);

– **MTS-congestion**: the subject could not be progressed, due to congestion in the MTS;

– **loop-detected**: the subject was detected looping within the MTS;

– **recipient-unavailable**: the recipient MTS-user was (or would be) unavailable to take delivery of the subject-message;

– **maximum-time-expired**: the maximum time for delivering the subject-message, or performing the subject-probe, expired;

– **encoded-information-types-unsupported**: the **encoded-information-types** of the subject-message are unsupported by the recipient MTS-user;

– **content-too-long**: the **content-length** of the subject-message is too long for the recipient MTS-user to take delivery (exceeds the **deliverable-maximum-content-length**);

– **conversion-impractical**: a conversion required for the subject-message to be delivered is impractical;

– **implicit-conversion-prohibited**: a conversion required for the subject-message to be delivered has been prohibited by the originator of the subject (see § 8.2.1.1.1.9);

– **implicit-conversion-not-subscribed**: a conversion required for the subject-message to be delivered has not been subscribed to by the recipient;

– **invalid-arguments**: one or more arguments in the subject was detected as being invalid;

– **content-syntax-error**: a syntax error was detected in the **content** of the subject-message (not applicable to subject-probes);

– **size-constraint-violation**:indicates that the value of one or more parameter(s) of the subject violated the size constraints defined in this Recommendation, and that the MTS was not prepared to handle the specified value(s);

– **protocol-violation**: indicates that one or more mandatory argument(s) were missing from the subject;

– **content-type-not-supported**: indicates that processing of a **content-type** not supported by the MTS was (or would be) required to deliver the subject-message;

– **too-many-recipients**: indicates that the MTS was (or would be) unable to deliver the subject-message due to the number of specified recipients of the subject-message (see § 8.2.1.1.1.2);

– **no-bilateral-agreement**: indicates that delivery of the subject-message required (or would require) a bilateral agreement where no such agreement exists;

– **unsupported-critical-function**: indicates that a critical function required for the transfer or delivery of the subject-message was not suported by the originating-MTA of the report;

– **conversion-with-loss-prohibited**: a conversion required for the subject-message to be delivered would have resulted in loss of information; conversion with loss of information was prohibited by the originator of the subject (see § 8.2.1.1.1.10);

– **line-too-long**: a conversion required for the subject-message to be delivered would have resulted in loss of information because the original line length was too long;

– **page-split**: a conversion required for the subject-message to be delivered would have resulted in loss of information because an original page would be split;

– **pictorial-symbol-loss**: a conversion required for the subject-message to be delivered would have resulted in loss of information because of a loss of one or more pictorial symbols;

– **punctuation-symbol-loss**: a conversion required for the subject-message to be delivered would have resulted in loss of information because of a loss of one or more punctuation symbols;

– **alphabetic-character-loss**: a conversion required for the subject-message to be delivered would have resulted in loss of information because of a loss of one or more alphabetic characters;

– **multiple-information-loss**: a conversion required for the subject-message to be delivered would have resulted in multiple loss of information;

– **recipient-reassignment-prohibited**: indicates that the MTS was (or would be) unable to deliver the subject-message because the originator of the subject prohibited redirection to a **recipient-assigned-alternate-recipient** (see § 8.2.1.1.1.4);

– **redirection-loop-detected**: the subject-message could not be redirected to an alternate-recipient because that recipient had previously redirected the message (redirection-loop);

– **DL-expansion-prohibited**: indicates that the MTS was (or would be) unable to deliver the subject-message because the originator of the subject prohibited the expansion of DLs (see § 8.2.1.1.1.6);

– **no-DL-submit-permission**: the originator of the subject (or the DL of which this DL is a member, in the case of nested DLS) does not have permission to submit messages to this DL;

– **DL-expansion-failure**: indicates that the MTS was unable to complete the expansion of a DL;

– **physical-rendition-attributes-not-supported**: the PDAU does not support the physical-rendition-attributes requested (see § 8.2.1.1.1.20);

– **undeliverable-mail-physical-delivery-address-incorrect**: the subject-message was undeliverable because the specified recipient **postal-OR-address** was incorrect;

– **undeliverable-mail-physical-delivery-office-incorrect-or-invalid**: the subject-message was undeliverable because the physical-delivery-office identified by the specified recipient **postal-OR-address** was incorrect or invalid (does not exit);

– **undeliverable-mail-physical-delivery-address-incomplete**: the subject-message was undeliverable because the specified recipient **postal-OR-address** was incompletely specified;

– **undeliverable-mail-recipient-unknown**: the subject-message was undeliverable because the recipient specified in the recipient **postal-OR-address** was not known at that address;

– **undeliverable-mail-recipient-deceased**: the subject-message was undeliverable because the recipient specified in the recipient **postal-OR-address** is deceased;

– **undeliverable-mail-organization-expired**: the subject-message was undeliverable because the recipient organization specified in the recipient **postal-OR-address** has expired;

– **undeliverable-mail-recipient-refused-to-accept**: the subject-message was undeliverable because the recipient specified in the recipient **postal-OR-address** refused to accept it;

– **undeliverable-mail-recipient-did-not-claim**: the subject-message was undeliverable because the recipient specified in the recipient **postal-OR-address** did not collect the mail;

- **undeliverable-mail-recipient-changed-address-permanently**: the subject-message was undeliverable because the recipient specified in the recipient **postal-OR-address** has changed address permanently (‘moved'), and forwarding was not applicable;
- **undeliverable-mail-recipient-changed-address-temporarily**: the subject-message was undeliverable because the recipient specified in the recipient **postal-OR-address** has changed address temporarily (‘on travel'), and forwarding was not applicable;
- **undeliverable-mail-recipient-changed-temporary-address**: the subject-message was undeliverable because the recipient specified in the recipient **postal-OR-address** had changed temporary address (‘departed'), and forwarding was not applicable;
- **undeliverable-mail-new-address-unknown**: the subject-message was undeliverable because the recipient has moved and the recipient's new address is unknown;
- **undeliverable-mail-recipient-did-not-want-forwarding**: the subject-message was undeliverable because delivery would have required physical-forwarding which the recipient did not want;
- **undeliverable-mail-originator-prohibited-forwarding**: the physical-forwarding required for the subject-message to be delivered has been prohibited by the originator of the subject-message (see § 8.2.1.1.1.15);
- **secure-messaging-error**: the subject could not be progressed because it would violate the security-policy in force;
- **unable-to-downgrade**: the subject could not be transferred because it could not be downgraded (see Annex B to Recommendation X.419).

Other **non-delivery-diagnostic-codes** may be specified in future versions of this Recommendation.

8.3.1.2.1.12 *Reporting-MTA-certificate*

This argument contains the **certificate** of the MTA that generated the report. It shall be generated by a trusted source (e.g., a certification-authority), and may be supplied by the reporting-MTA if a **report-origin-authentication-check** is supplied.

The **reporting-MTA-certificate** may be used to convey a verified copy of the public-asymmetric-encription-key (**subject-public-key**) of the reporting-MTA.

The reporting-MTA's public-asymmetric-encryption-key may be used by the originator of the message, and any MTA through which the report is transferred, to validate the **report-origin-authentication-check**.

8.3.1.2.1.13 *Report-origin-authentication-check*

This argument provides the originator of the subject-message (or -probe), and any other MTA through which the report is transferred, with a means of authenticating the origin of the report (to provide the report origin authentication element-of-service as defined in Recommendation X.400). It may be generated by the reporting-MTA if a **message-** (or **probe-**) **origin-authentication-check** was present in the subject.

The **report-origin-authentication-check** provides proof of the origin of the report (report origin authentication), and proof of association between the **message-security-label** and the report.

The **report-origin-authentication-check** is computed using the algorithm identified by the **report-origin-authentication-algorithm-identifier** (an **algorithm-identifier**).

The **report-origin-authentication-check** contains the **report-origin-authentication-algorithm-identifier**, and an asymmetrically encrypted, hashed version of the **report-origin-authentication-algorithm-identifier**, the **content-identifier** and **message-security-label** of the subject, and all values of the following (per-recipient) arguments: the **actual-recipient-name**, the **originally-intended-recipient-name**, and:

- for a delivery-report: the **message-delivery-time**, the **type-of-MTS-user**, and if requested by the originator of the message for recipients to which the report relates, the **recipient-certificate**, and the **proof-of-delivery** (not present in a report on a probe); or
- for a non-delivery-report: the **non-delivery-reason-code** and **non-delivery-diagnostic-code**.

Optional components are included in the **report-authentication-check** if they are present in the report.

The **report-origin-authentication-check** may be computed by the reporting-MTA using the reporting-MTA's secret-asymmetric-encryption-key. The **report-origin-authentication-check** may be validated by the originator of the subject, and any MTA through which the report is transferred, using the reporting-MTA's public-asymmetric-encryption-key (**subject-public-key**) derived from the **reporting-MTA-certificate**.

Future versions of this Recommendation may define other forms of **report-origin-authentication-check** (e.g., based on symmetric-encryption-techniques) which may be used by MTAs through which the report is transferred to authenticate the origin of the report.

8.3.1.2.1.14 *Returned-content*

This argument contains the **content** of the subject-message if the originator of the subject-message indicated that the **content** was to be returned (see § 8.2.1.1.1.23). It shall be generated by the originator of the message, and may be returned by the MTS (if the reporting-MTA or originating-MTA supports the return of content element-of-service).

This argument may only be present if there is at least one non-delivery report in the Report-delivery, and if the recipient of the report is the originator of the subject-message (and not, for example, the owner of a DL (see § 8.3.1.2.1.4)).

This argument shall not be present if any **encoded-information-type** conversion has been performed on the **content** of the subject-message.

8.3.1.2.2 *Results*

The report-delivery abstract-operation returns an empty result as indication of success.

8.3.1.2.3 *Abstract-errors*

Table 19/X.411 lists the abstract-errors that may disrupt the report-delivery abstract-operation, and for each abstract-error identifies the clause in which the abstract-error is defined.

TABLE 19/X.411

Report-delivery abstract-errors

Abstract-error	Clause
Delivery-control-violated	8.3.2.1
Security-error	8.3.2.3
Unsupported-critical-function	8.3.2.4

8.3.1.3 *Delivery-control*

The delivery-control abstract-operation enables the MTS-user to temporarily limit the delivery-port abstract-operations that the MTS may invoke, and the messages that the MTS may deliver to the MTS-user via the message-delivery abstract-operation.

The MTS shall hold until a later time, rather than abandon, abstract-operations and messages presently forbidden.

The successful completion of the abstract-operation signifies that the specified controls are now in force. These controls supersede any previously in force, and remain in effect until the association is released, the MTS-user re-invokes the delivery-control abstract-operation, or the MTS-user invokes the administration-port register abstract-operation to impose constraints more severe than the specified controls.

The abstract-operation returns an indication of any abstract-operations that the MTS would invoke, or any message types that the MTS would deliver or report, were it not for the prevailing controls.

8.3.1.3.1 *Arguments*

Table 20/X.411 lists the arguments of the delivery-control abstract-operation, and for each argument qualifies its presence and identifies the clause in which the argument is defined.

8.3.1.3.1.1 *Restrict*

This argument indicates whether the controls on delivery-port abstract-operations are to be updated or removed. It may be generated by the MTS-user.

This argument may have one of the following values:

– **update**: the other arguments update the prevailing controls;

– **remove**: all temporary controls are to be removed (the default controls registered with the MTS by means of the administration-port register abstract-operation shall apply); the other arguments are to be ignored.

In the absence of this argument, the default **update** shall be assumed.

TABLE 20/X.411

Delivery-control arguments

Arguments	Presence	Clause
Delivery control arguments		
Restrict	O	8.3.1.3.1.1
Permissible-operations	O	8.3.1.3.1.2
Permissible-lowest-priority	O	8.3.1.3.1.3
Permissible-encoded-information-types	O	8.3.1.3.1.4
Permissible-content-types	O	8.3.1.3.1.5
Permissible-maximum-content-length	O	8.3.1.3.1.6
Permissible-security-context	O	8.3.1.3.1.7

8.3.1.3.1.2 *Permissible-operations*

This argument indicates the abstract-operations that the MTS may invoke on the MTS-user. It may be generated by the MTS-user.

This argument may have the value **allowed** or **prohibited** for each of the following:

– **message-delivery**: the MTS may/may not invoke the message-delivery abstract-operation; and

– **report-delivery**: the MTS may/may not invoke the report-delivery abstract-operation.

Other delivery-port abstract-operations are not subject to controls, and may be invoked at any time.

In the absence of this argument, the abstract-operations that the MTS may invoke on the MTS-user are unchanged. If there has been no previous invocation of the delivery-control abstract-operation on the association, the default control registerd with the MTS by means of the administration-port Register abstract-operation shall apply.

8.3.1.3.1.3 *Permissible-lowest-priority*

This argument contains the **priority** of the lowest priority message that the MTS shall deliver to the MTS-user via the message-delivery abstract-operation. It may be generated by the MTS-user.

This argument may have one of the following values of the **priority** argument of the message-submission abstract-operation: **normal**, **non-urgent** or **urgent**.

In the absence of this argument, the **priority** of the lowest priority message that the MTS shall deliver to the MTS-user is unchanged. If there has been no previous invocation of the delivery-control abstract-operation on the association, the default control registered with the MTS by means of the adminsitration-port Register abstract-operation shall apply.

8.3.1.3.1.4 *Permissible-encoded-information-types*

This argument indicates the only **encoded-information-types** that shall appear in messages that the MTS shall deliver to the MTS-user via the message-delivery abstract-operation. It may be generated by the MTS-user.

The **permissible-encoded-information-types** specified shall be among those allowed long-term due to a previous invocation of the administration-port register abstract-operation (**deliverable-encoded-information-types**).

In the absence of this argument, the **permissible-encoded-information-types** that the MTS may deliver to the MTS-user are unchanged. If there has been no previous invocation of the delivery-control abstract-operation on the association, the default control registered with the MTS by means of the administration-port register abstract-operation shall apply.

8.3.1.3.1.5 *Permissible-content-types*

This argument indicates the only **content-types** that shall appear in messages that the MTS shall deliver to the MTS-user via the message-delivery abstract-operation. It may be generated by the MTS-user.

The **permissible-content-types** specified shall be among those allowed long-term due to a previous invocation of the administration-port register abstract-operation (**deliverable-content-types**).

In the absence of this argument, the **permissible-content-types** that the MTS may deliver to the MTS-user are unchanged. If there has been no previous invocation of the delivery-control abstract-operation on the association, the default control registered with the MTS by means of the administration-port register abstract-operation shall apply.

8.3.1.3.1.6 *Permissible-maximum-content-length*

This argument contains the **content-length**, in octets, of the longest-content message that the MTS shall deliver to the MTS-user via the message-delivery abstract-operation. It may be generated by the MTS-user.

The **persmissible-maximum-content-length** shall not exceed that allowed long-term due to a previous invocation of the administration-port register abstract-operation (**deliverable-maximum-content-length**).

In the absence of this argument, the **permissible-maximum-content-length** of a message that the MTS may deliver to the MTS-user is unchanged. If there has been no previous invocation of the delivery-control abstract-operation on the association, the default control registered with the MTS by means of the administration port register abstract-operation shall apply.

8.3.1.3.1.7 *Permissible-security-context*

This argument temporarily limits the sensitivity of delivery-port abstract-operations (delivery-security-context) that the MTS may invoke on the MTS-user. It is a temporary restriction of the **security-context** established when the association was initiated (see § 8.1.1.1.1.4). It may be generated by the MTS-user.

The **permissible-security-context** comprises one or more **security-labels** from the set of **security-labels** established as the **security-context** when the association was established.

In the absence of this argument, the **security-context** of delivery-port abstract-operations is unchanged.

8.3.1.3.2 *Results*

Table 21/X.411 lists the results of the delivery-control abstract-operation, and for each result qualifies its presence and identifies the clause in which the result is defined.

TABLE 21/X.411

Delivery-control results

Results	Presence	Clause
"Waiting" results		
Waiting-operations	O	8.3.1.3.2.1
Waiting-messages	O	8.3.1.3.2.2
Waiting-encoded-information-types	O	8.3.1.3.2.3
Waiting-content-types	O	8.3.1.3.2.4

8.3.1.3.2.1 *Waiting-operations*

This result indicates the abstract-operations being held by the MTS,and that the MTS would invoke on the MTS-user if it were not for the prevailing controls. It may be generated by the MTS.

This result may have the value **holding** or **not-holding** for each of the following:

– **message-delivery**: the MTS is/is not holding messages, and would invoke the message-delivery abstract-operation on the MTS-user if it were not for the prevailing controls; and

– **report-delivery**: the MTS is/is not holding reports, and would invoke the report-delivery abstract-operation on the MTS-user if it were not for the prevailing controls.

In the absence of this result, it may be assumed that the MTS is not holding any messages or reports for delivery due to the prevailing controls.

8.3.1.3.2.2 *Waiting-messages*

This result indicates the kind of messages the MTS is holding for delivery to the MTS-user, and would deliver via the message-delivery abstract-operation, if it were not for the prevailing controls. It may be generated by the MTS.

This result may have one or more of the following values:

– **long-content**: the MTS has messages held for delivery to the MTS-user which exceed the **permissible-maximum-content-length** control currently in force;

– **low-priority**: the MTS has messages held for delivery to the MTS-user of a lower priority than the **permissible-lowest-priority** control currently in force;

– **other-security-labels**: the MTS has messages held for delivery to the MTS-user bearing **message-security-labels** other than those permitted by the current security-context.

In the absence of this result, it may be assumed that the MTS is not holding any messages for delivery to the MTS-user due to the **permissible-maximum-content-length**, **permissible-lowest-priority** or **permissible-security-context** controls currently in force.

8.3.1.3.2.3 *Waiting-encoded-information-types*

This result indicates the **encoded-information-types** in the **content** of any messages held by the MTS for delivery to the MTS-user due to prevailing controls. It may be generated by the MTS.

In the absence of this result, the **encoded-information-types** of any messages held by the MTS for delivery to the MTS-user are **unspecified**.

8.3.1.3.2.4 *Waiting-content-types*

This result indicates the **content-types** of any messages held by the MTS for delivery to the MTS-user due to prevailing controls. It may be generated by the MTS.

In the absence of this result, the **content-types** of any messages held by the MTS for delivery to the MTS-user are **unspecified**.

8.3.1.3.3 *Abstract-errors*

Table 22/X.411 lists the abstract-errors that may disrupt the delivery-control abstract-operation, and for each abstract-error identifies the clause in which the abstract-error is defined.

TABLE 22/X.411

Delivery-control abstract-errors

Abstract-error	Clause
Control-violates-registration	8.3.2.2
Security-error	8.3.2.3

8.3.2 *Abstract-errors*

This clause defines the following delivery-port abstract-errors:

a) delivery-control-violated

b) control-violates-registration

c) security-error

d) unsupported-critical-function.

8.3.2.1 *Delivery-control-violated*

The delivery-control-violated abstract-error reports the violation by the MTS of a control on delivery-port abstract-operations imposed by the MTS-user via the delivery-control abstract-operation.

The delivery-control-violated abstract-error has no parameters.

8.3.2.2 *Control-violates-registration*

The control-violates-registration abstract-error reports that the MTS is unable to accept the controls that the MTS-user attempted to impose on delivery-port abstract-operations because they violate existing registration parameters.

The control-violates-registration abstract-error has no parameters.

8.3.2.3 *Security-error*

The security-error abstract-error reports that the requested abstract-operation could not be provided by the MTS-user because it would violate the security-policy in force.

The security-error abstract-error has the following parameters, generated by the MTS-user:

– **security-problem**: an identifier for the cause of the violation of the security-policy.

8.3.2.4 *Unsupported-critical-function*

The unsupported-critical-function abstract-error reports that an argument of the abstract-operation was marked **critical-for-delivery** (see § 9.1) but is unsupported by the MTS-user.

The unsupported-critical-function abstract-error has no parameters.

8.4 *Administration port*

This section defines the abstract-operations and abstract-errors which occur at an administration-port.

8.4.1 *Abstract-operations*

This section defines the following administration-port abstract-operations:

a) register

b) change-credentials.

8.4.1.1 *Register*

The register abstract-operation enables an MTS-user to make long-term changes to various parameters of the MTS-user held by the MTS concerned with delivery of messages to the MTS-user.

Such changes remain in effect until overridden by re-invocation of the register abstract-operation. However, some parameters may be temporarily overridden by invocation of the delivery-control abstract-operation.

Note 1 – This abstract-operation shall be invoked before any other submission-port, delivery-port or administration-port abstract-operation may be used, or an equivalent registration by local means shall have taken place.

Note 2 – This abstract-operation does not encompass the standing parameters implied by the alternate recipient allowed element-of-service and the restricted delivery element-of-service defined in Recommendation X.400. The manner in which those parameters are supplied and modified are a local matter.

8.4.1.1.1 *Arguments*

Table 23/X.411 lists the arguments of the register abstract-operation, and for each argument qualifies its presence and identifies the section in which the argument is defined.

TABLE 23/X.411

Register arguments

Argument	Presence	Clause
Registration arguments		
User-name	O	8.4.1.1.1.1
User-address	O	8.4.1.1.1.2
Deliverable-encoded-information-types	O	8.4.1.1.1.3
Deliverable-content-types	O	8.4.1.1.1.4
Deliverable-maximum-content-length	O	8.4.1.1.1.5
Recipient-assigned-alternate-recipient	O	8.4.1.1.1.6
User-security-labels	O	8.4.1.1.1.7
Default delivery control arguments		8.4.1.1.1.8
Restrict	O	8.3.1.3.1.1
Permissible-operations	O	8.3.1.3.1.2
Permissible-lowest-priority	O	8.3.1.3.1.3
Permissible-encoded-information-types	O	8.3.1.3.1.4
Permissible-content-types	O	8.3.1.3.1.5
Permissible-maximum-content-length	O	8.3.1.3.1.6

8.4.1.1.1.1 *User-name*

This argument contains the **OR-name** of the MTS-user, if the **user-name** is to be changed. It may be generated by the MTS-user.

In the absence of this argument, the **user-name** of the MTS-user remains unchanged.

An MD is not required to provide MTS-users with the ability to change their **OR-names**. If it does so, the MD may restrict that ability. It may prohibit certain MTS-users from changing their **OR-names**, or it may restrict the scope of the change to a locally defined subset of the components of their **OR-names**. A proposed new **OR-names** shall be rejected if it is already assigned to another MTS-user.

8.4.1.1.1.2 *User-address*

This argument contains the **user-address** of the MTS-user, if it is required by the MTS and if it is to be changed. It may be generated by the MTS-user.

The user-address may contain one of the following forms of address of the MTS-user.

– the **X.121-address** and/or the **TSAP-ID** (transport service access point identifier); or

– the **PSAP-address** (presentation service access point address).

Other forms of **user-address** may be defined in future versions of this Recommendation.

In the absence of this argument, the **user-address** of the MTS-user (if any) remains unchanged.

8.4.1.1.1.3 *Deliverable-encoded-information-types*

This argument indicates the **encoded-information-types** that the MTS shall permit to appear in messages delivered to the MTS-user, if they are to be changed. It may be generated by the MTS-user.

The MTS shall reject as undeliverable any message for an MTS-user for which the MTS-user is not registered to accept delivery of all the **encoded-information-types** of the message. Note that the MTS-user may register to receive the **undefined encoded-information-type**. Deliverable-encoded-information-types also indicates the possible encoded-information-types to which implicit conversion can be performed.

In the absence of this argument, the **deliverable-encoded-information-types** shall remain unchanged.

8.4.1.1.1.4 *Deliverable-content-types*

This argument indicates the **content-types** that the MTS shall permit to appear in messages delivered to the MTS-user, if they are to be changed. It may be generated by the MTS-user.

The MTS shall reject as undeliverable any message for an MTS-user for which the MTS-user is not registered to accept delivery of the **content-types** of the message. Note that the MTS-user may register to receive the **undefined content-type**.

In the absence of this argument, the **deliverable-content-types** shall remain unchanged.

8.4.1.1.1.5 *Deliverable-maximum-content-length*

This argument contains the **content-length**, in octets, of the longest-content message that the MTS shall permit to appear in messages delivered to the MTS-user, if it is to be changed. It may be generated by the MTS-user.

The MTS shall reject as undeliverable any message for an MTS-user for which the MTS-user is not registered to accept delivery of messages of its size.

In the absence of this argument, the **deliverable-maximum-content-length** of messages shall remain unchanged.

8.4.1.1.1.6 *Recipient-assigned-alternate-recipient*

This argument contains the **OR-name** of an alternate-recipient, specified by the MTS-user, to which messages are to be redirected, if the alternate-recipient is to be changed. It may be generated by the MTS-user. A different value of this argument may be specified for each value of **user-security-labels**.

If a **recipient-assigned-alternate-recipient** is registered and associated with a value of **user-security-labels**, messages bearing a matching **message-security-label** shall be redirected to the alternate-recipient. Messages bearing a **message-security-label** for which no **recipient-assigned-alternate-recipient** has been registered, shall not be redirected to a **recipient-assigned-alternate-recipient**.

If a single **recipient-assigned-alternate-recipient** is registered, and not associated with a value of **user-security-labels**, all messages shall be redirected to the alternate-recipient.

The **recipient-assigned-alternate-recipient** shall contain the **OR-name** of the alternate-recipient. If the **recipient-assigned-alternate-recipient** contains the **OR-names** of the MTS-user (see § 8.4.1.1.1.1), no **recipient-assigned-alternate-recipient** is registered.

In the absence of this argument, the **recipient-assigned-alternate-recipient**, if any, remains unchanged.

8.4.1.1.1.7 *User-security-labels*

This argument contains the **security-labels** of the MTS-user, if they are to be changed. It may be generated by the MTS-user.

A **recipient-assigned-alternate-recipient** may be registered for any value of **user-security-labels**.

In the absence of this argument, the **user-security-labels** remain unchanged.

Note that some security-policies may only permit the **user-security-labels** to be changed in this way if a secure link is employed. Other local means of changing the **user-security-labels** in a secure manner may be provided.

8.4.1.1.1.8 *Default delivery control arguments*

The default control arguments are the same as the arguments of the delivery-control abstract-operation, and are defined in § 8.3.1.3.1. Except for **permissible-security-context**, they may be generated by the MTS-user.

The default controls are registered as arguments of the register abstract-operation. These defaults come into effect at the beginning of an association, and remain in effect until they are overridden by an invocation of the delivery-control abstract-operation.

The default control arguments shall not admit messages whose delivery are prohibited by the prevailing registered values of the **deliverable-encoded-information-types** argument, the **deliverable-content-types** argument or the **deliverable-maximum-content-length** argument.

8.4.1.1.2 *Results*

The register abstract-operation returns an empty result as indication of success.

8.4.1.1.3 *Abstract-errors*

Table 24/X.411 lists the abstract-errors that may disrupt the register abstract-operation, and for each abstract-error identifies the clause in which the abstract-error is defined.

TABLE 24/X.411

Register abstract-error

Abstract-error	Clause
Register-rejected	8.4.2.1

8.4.1.2 *Change-credentials*

The change-credentials abstract-operation enables the MTS-user to change the MTS-user's **credentials** held by the MTS, or enables the MTS to change the MTS's **credentials** held by the MTS-user.

The **credentials** are exchanged during the establishment of an association for the mutual authentication of identity of the MTS-user and the MTS.

The successful completion of the abstract-operation signifies that the **credentials** have been changed.

The disruption of the abstract-operation by an abstract-error indicates that the **credentials** have not been changed, either because the old **credentials** were incorrectly specified or that the new **credentials** are unacceptable.

8.4.1.2.1 *Arguments*

Table 25/X.411 lists the arguments of the change-credentials abstract-operation, and for each argument qualifies its presence and identifies the clause in which the argument is defined.

TABLE 25/X.411

Change-credentials arguments

Argument	Presence	Clause
Credential arguments		
Old-credentials	M	8.4.1.2.1.1
New-credentials	M	8.4.1.2.1.2

8.4.1.2.1.1 *Old-credentials*

This argument contains the current (old) **credentials** of the invoker of the abstract-operation, held by the performer of the abstract-operation. It shall be generated by the invoker of the abstract-operation.

If only simple-authentication is used, the **credentials** comprise a simple **password** associated with the **user-name**, or **MTA-name**, of the invoker.

If strong-authentication is used, the **credentials** comprise the **certificate** of the invoker, generated by a trusted source (e.g. a certification-authority), and supplied by the invoker.

8.4.1.2.1.2 *New-credentials*

This argument contains the proposed new **credentials** of the invoker of the abstract-operation, to be held by the performer of the abstract-operation. It shall be generated by the invoker of the abstract-operation.

The **new-credentials** shall be of the same type (i.e. simple or strong) as the **old-credentials**, as defined in § 8.4.1.2.1.1.

8.4.1.2.2 *Results*

The change-credentials abstract-operation returns an empty result as indication of success.

8.4.1.2.3 *Abstract-errors*

Table 26/X.411 lists the abstract-erros that may disrupt the change-credentials abstract-operation, and for each abstract-error identifies the paragraph in which the abstract-error is defined.

TABLE 26/X.411

Change-credentials abstract-errors

Abstract-error	Clause
New-credentials-unacceptable	8.4.2.2
Old-credential-incorrectly-specified	8.4.2.3

8.4.2 *Abstract-errors*

This section defines the following administration-port abstract-errors:

a) register-rejected

b) new-credentials-unacceptable

c) old-credentials-incorrectly-specified.

8.4.2.1 *Register-rejected*

The register-rejected abstract-error reports that the requested parameters cannot be registered because one or more are improperly specified.

The register-rejected abstract-error has no parameters.

8.4.2.2 *New-credentials-unacceptable*

The new-credentials-unacceptable abstract-error reports that the **credentials** cannot be changed because the **new-credentials** are unacceptable.

The new-credentials-unacceptable abstract-error has no parameters.

8.4.2.3 *Old-credentials-incorrectly-specified*

The old-credentials-incorrectly-specified abstract-error reports that the **credentials** cannot be changed because the current (**old-**) **credentials** were incorrectly specified.

The old-credentials-specified abstract-error has no parameters.

8.5 *Common parameter types*

This clause defines a number of common parameter types of the MTS abstract service.

8.5.1 *MTS-identifier*

MTS-identifiers are assigned by the MTS to distinguish between messages and probes at the MTS abstract service, and between messages, probes and reports within the MTS.

The **MTS-identifier** assigned to a message at a submission-port (**message-submission-identifier**) is identical to the corresponding **message-identifier** at a transfer-port and corresponding **message-delivery-identifier** at a delivery-port. Similarly, the **MTS-identifier** assigned to a probe at a submission-port (**probe-submission-identifier**) is identical to the corresponding **probe-identifier** at a transfer-port. **MTS-identifiers** are also assigned to reports at transfer-ports (**report-identifier**).

An **MTS-identifier** comprises:

– a **local-identifier** assigned by the MTA, which unambiguously identifies the related event within the MD;

– the **global-domain-identifier** of the MD, which ensures that the **MTS-identifier** is unambiguous throughout the MTS.

8.5.2 *Global-domain-identifier*

A **global-domain-identifier** unambiguously identifies an MD within the MHS.

A **global-domain-identifier** is used to ensure that an **MTS-identifier** is unambiguous throughout the MTS, and for identifying the source of a **trace-information-element.**

In the case of an ADMD, a **global-domain-identifier** consists of the **country-name** and the **administration-domain-name** of the MD. For a PRMD, it consists of the **contry-name** and the **administration-domain-name** of the associated ADMD, plus a **private-domain-identifier**. The **private-domain-identifier** is a unique identification of the PRMD, and may be identical to the PRMD's **private-domain-name**. As a national matter, this identification may be either relative to the country denoted by the **country-name** or relative to the associated ADMD.

Note 1 – The distinction between **private-domain-identifier** and **private-domain-name** has been retained for backward compatibility with Recommendation X.411 (1984). Often they will be identical.

Note 2 – In the **global-domain-identifier** of a PRMD, the **administration-domain-name** of the associated ADMD is optional ISO/IEC 10021-4.

8.5.3 *MTA-name*

An **MTA-name** is an identifier for an MTA that uniquely identifies the MTA within the MD to which it belongs.

8.5.4 *Time*

A **time** parameter is specified in terms of UTC (Coordinated Universal Time), and may optionally also contain an offset to UTC to convey the local time. The precision of the time of day is to either one second or one minute, determined by the generator of the parameter.

8.5.5 *OR-name*

An **OR-name** identifies the originator or recipient of a message according to the principles of naming and addressing described in Recommendation X.402.

At a submission-port, an **OR-name** comprises an **OR-address**, or a **directory-name**, or both (**OR-address-and-or-directory-name**). At all other types of port, an **OR-name** comprises an **OR-address** and, optionally, **directory-name (OR-address-and-optional-directory-name)**. A **directory-name** and an **OR-address** may each denote an individual originator or recipient, or a DL.

A **directory-name** is as defined in Recomendation X.501. The MTS uses the **directory-name** only when the **OR-address** is absent or invalid.

An **OR-address** comprises a number of **standard-attributes**, optionally a number of **extension-attributes**, and optionally a number of attributes defined by the MD to which the originator/recipient subscribes (**domain-defined-attributes**).

The **standard-** and **extension-attributes** used in an **OR-address** are selected from those defined in Recommendation X.402. Only those combinations of attributes explicitly defined in Recommendation X.402 can be used to form a valid **OR-address**.

8.5.6 *Encoded-information-types*

The **encoded-information-types** of a message are the kind(s) of information that appear in its **content**. Both basic **encoded-information-types** and externally-defined **encoded-information-types** may be specified, otherwise the **encoded-information-types** of a message are unspecified.

Externally-defined **encoded-information-types** are those to which object-identifiers are allocated by an appropriate authority. They include both standardised and private-defined **encoded-information-types**.

The basic **encoded-information-types** are those originally specified in the Recommendation X.411 (1984). The **undefined** type is any type other than the specified externally-defined **encoded-information-types** and other than the following types. The **telex** type is defined in Recommendation F.1. The **ia5-text** (teleprinter) type is defined in Recommendation T.50. The **g3-facsimile** type is defined in Recommendations T.4 and T.30. The **g4-class-1** type is defined in Recommendations T.5, T.6, T.400 and T.503. The **teletex** type is defined in Recommendations F.200, T.61 and T.60. The **videotex** type is defined in Recommendations T.100 and T.101. The **simple-formattable-document (sfd)** type is defined in Recommendation X.420 (1984) (Note that SFDs are no longer defined in any 1988 Recommendation). The **mixed-mode** type is defined in Recommendations T.400 and T.501.

Non-basic-parameters are defined for the **g3-facsimile, teletex, g4-class-1** and **mixed-mode** basic **encoded-information-types** for backwards compatibility with the Recommendation X.411 (1984) only. It is recommended that for each required combination of a basic **encoded-information-type** and a specific set of **non-basic-parameters**, an externally-defined **encoded-information-type** be defined and used in preference.

Note that **non-basic parameters** are likely to be removed from a future version of this Recommendation.

The **non-basic-parameters** for **g3-facsimile** correspond to the three- or four-octet Facsimile Information Field (FIF) conveyed by the Digital Command Signal (DCS) defined in Recommendation T.30. The parameters are: **two-dimensional, fine-resolution, unlimited-length, b4-length, a3-width, b4-width** and **uncompressed**.

The **non-basic-parameters** for **teletex** correspond to the non-basic terminal capability conveyed by the Command Document Start (CDS) defined in Recommendation T.62. The parameters are: optional **graphic-character-sets**, optional **control-character-sets**, optional **page-formats**, optional **miscellaneous-terminal-capabilities**, and a **private-use** parameter.

The **non-basic-parameters** for the **g4-class-1** and **mixed-mode** types specify optional resolution, optional graphic character sets, optional control character sets, and so on, which correspond to the parameters of the **presentation-capabilities** defined in Recommendations T.400, and T.503 and T.501.

Where **non-basic-parameters** are indicated, these parameters represent the logical ‘OR' of the **non-basic-parameters** of each instance on the **encoded-information-type** in a message **content**. Thus, this parameter only serves to indicate whether there is **encoded-information-type** compatibility, or whether conversion is required. If conversion is required, the message **content** shall be inspected to determine which **non-basic-parameters** apply to any instance of the **encoded-information-type**.

8.5.7 *Certificate*

A **certificate** may be used to convey a verified copy of the public-asymmetric-encryption-key of the subject of the **certificate**.

A **certificate** contains the following parameters:

– **signature-algorithm-identifier**: an **algorithm-identifier** for the algorithm used by the certification–authority that issued the **certificate** to compute the **signature**;

– **issuer**: the **directory-name** of the certification-authority that issued the **certificate**;

– **validity**: a date and time of day before which the **certificate** should not be used, and a date and time of day after which the **certificate** should not be relied upon;

– **subject**: the **directory-name** of the subject of the **certificate**;

– **subject-public-keys**: one or more public-asymmetric-encryption-keys of the subject (each used in conjunction with an **algorithm** and a secret-asymmetric-encryption-key of the subject);

– **algorithms**: one or more **algorithm-identifiers**, each associated with a **subject-public-key**;

– **signature**: an asymmetrically encrypted, hashed version of the above parameters computed by the certification-authority that issued the **certificate** using the algorithm identified by the **signature-algorithm-identifier** and the certification-authority's secret-asymmetric-encryption-key.

If the originator and a recipient of a **certificate** are served by the same certification-authority, the recipient may use the certification-authority's public-asymmetric-encryption-key to validate the **certificate**, and derive the originator's public-asymmetric-encryption-key (**subject-public-key**).

If the originator and a recipient of a **certificate** are served by different certification-authorities, the recipient may require a return-certification-path to authenticate the originator's **certificate**. The **certificate** may therefore include an associated **certification-path**.

The **certification-path** may comprise a **forward-certification-path** which includes the certificate of the certification-authority that issued the **certificate**, together with the certificates of all of its superior certfication-authorities. The **forward-certification-path** may also include the certificates of other certification-authorities, cross-certified by either the certification-authority that issued the **certificate**, or any of its superior certification-authorities.

A recipient of the **certificate** may complete the required return-certification-path between the recipient and the originator of the **certificate** by appending the recipient's own reverse-certification-path to the **forward-certification-path** supplied by the originator, at a common-point-of-trust. The reverse-certification-path includes the reverse-certificate of the certification-authority of the recipient of the **certificate**, together with the reverse-certificate of all of it superior certification-authorities. The reverse-certification-path may also include the reverse-certificates of other certification-authorities, cross-certified by the certification-authority of the recipient of the **certificate**, or any of its superior certification authorities.

The return-certification-path thus formed allows the recipient of the **certificate** to validate each certificate in the return-certification-path in turn, to derive the public-asymmetric-encryption-key of the certification-authority that issued the **certificate**. The recipient may then use the public-asymmetric-encryption-key of the certification-authority that issued the **certificate** to validate the **certificate**, and derive the originator's public-asymmetric-encryption-key (**subject-public-key**).

The form of a **certificate**and a **certification-path** are further defined in Recommendation X.509.

Future versions of this Recommendation may define other key distribution techniques (e.g., based on symmetric-encryption-techniques).

8.5.8 *Token*

A **token** may be used to convey to the recipient of the **token** protected security-relevant information. The **token** provides authentication of public security-relevant information, and confidentiality and authentication of secret security-relevant information.

The type of a **token** is identified by a **token-type-identifier**. One type of **token** is currently defined by this Recommendation: an **asymmetric-token**. Other types of **token** may be defined by future versions of this Recommendation; for example, **tokens** based on symmetric-encryption techniques.

An **asymmetric-token** contains the following parameters:

– **signature-algorithm-identifier**: an **algorithm-identifier** for the algorithm used by the originator of the **token** to compute the **signature**;

– **recipient-name**: the **OR address and or directory name** of the intended–recipient of the **token**;

– **time**: the date and time of day when the **token** was generated;

– **signed-data**: public security-relevant information;

– **encryption-algorithm-identifier**: an **algorithm-identifier** for the algorithm used by the originator of the **token** to compute the **encrypted-data**;

– **encrypted-data**: secret security-relevant information encrypted by the originator of the **token** using the algorithm identified by the **encryption-algorithm-identifier** and the public-asymmetric-encryption-key of the intended-recipient of the **token**;

– **signature**: an asymmetrically encrypted, hashed version of the above parameters computed by the originator of the **token** using the algorithm identified by the **signature-algorithm-identifier** and the originator's secret-asymmetric-encryption-key.

The form of a **token** is further defined in Recommendation X.509.

8.5.9 *Security-label*

Security-labels may be used to associate security-relevant information with objects within the MTS.

Security-labels may be assigned to an object in line with the security-policy in force for that object. The security-policy may also define how **security-labels** are to be used to enforce that security-policy.

Within the scope of this Recommendation, **security-labels** may be associated with messages, probes and reports (see § 8.2.1.1.1.30), MTS-user (see § 8.4.1.1.1.7), MDs, MTAs and associations between an MTS-user and an MD(or MTA) (see § 8.1.1.1.1.4), or between MDs (or MTAs) (see § 12.1.1.1.1.4). Beyond the scope of this Recommendation, a security-policy may, as a local matter or by bilateral agreement, additionally assign **security-labels** to other objects within the MTS (e.g., secure routes).

A **security-label** comprises a set of **security-attributes**. The **security-attributes** may include a **security-policy-identifier**, a **security-classification**, a **privacy-mark**, and a set of **security-categories**.

A **security-policy-identifier** may be used to identify the security-policy in force to which the **security-label** relates.

If present, a **security-classification** may have one of a hierarchical list of values. The basic **security-classification** hierarchy is defined in this Recommendation, but the use of these values is defined by the security-policy in force. Additional values of **security-classification**, and their position in the hierarchy, may also be defined by a security-policy as a local matter or by bilateral agreement. The basic **security-classification** hierarchy is, in ascending order: **unmarked, unclassified, restricted, confidential, secret, top-secret**.

If present, a **privacy-mark** is a printable string. The content of the printable string may be defined by a security-policy, which may define a list of values to be used, or allow the value to be determined by the originator of the **security-label**. Examples of privacy-marks include **'IN CONFIDENCE'** and **'IN STRICTEST CONFIDENCE'**.

If present, the set of **security-categories** provide further restrictions within the context of a **security-classification** and/or **privacy-mark**, typically on a 'need-to-know' basis. The **security-categories** and their values may be defined by a security-policy as a local matter or by bilateral agreement. Examples of possible **security-categories** include caveats to the **security-classification** and/or **privacy-mark** (e.g., **'PERSONAL-', 'STAFF-', 'COMMERCIAL-'**, etc), closed-user-groups, codewords, etc.

8.5.10 *Algorithm-identifier*

An **algorithm-identifier** identifies an **algorithm** and any **algorithm-parameters** required by the **algorithm**.

An **algorithm-identifier** may be drawn from an international register of algorithms, or defined by bilateral agreement.

9 Message transfer system abstract syntax definition

The abstract-syntax of the MTS abstract service is defined in Figure 2/X.411.

The abstract-syntax of the MTS abstract service is defined using the abstract syntax notation (ASN.1) defined in Recommendation X.208, and the abstract service definition conventions defined in Recommendation X.407.

The abstract-syntax definition of the MTS abstract service has the following major parts:

– *Prologue:* declarations of the exports from, and imports to, the MTS abstract service module (Figure 2/X.411, Part 1).

– *Objects and ports:* definitions of the MTS and MTS-user objects, and their submission-, delivery- and administration-ports (Figure 2/X.411, Part 2).

– *MTS-bind and MTS-unbind:* definitions of the MTS-bind and MTS-unbind used to establish and release associations between an MTS-user and the MTS (Figure 2/X.411, Parts 3 to 4).

– *Submission port:* definitions of the submission-port abstract-operations: Message-submission, Probe-submission, Cancel-deferred-delivery and Submission-control; and their abstract-errors (Figure 2/X.411, Parts 5 to 7).

– *Delivery port:* definitions of the delivery-port abstract-operations: Message-delivery, Report-delivery and Delivery-control; and their abstract-errors (Figure 2/X.411, Parts 8 to 9).

– *Administration port:* definitions of the administration-port abstract-operations: Register and Change-credentials; and their abstract-errors (Figure 2/X.411, Parts 10 to 11).

– *Message submission envelope:* definition of the message-submission-envelope (Figure 2/X.411, Part 12).

– *Probe submission envelope:* definition of the probe-submission-envelope (Figure 2/X.411, Part 13).

– *Message delivery envelope:* definition of the message-delivery-envelope (Figure 2/X.411, Part 14).

– *Report delivery envelope:* definition of the report-delivery-envelope (Figure 2/X.411, Part 15).

– *Envelope fields:* definitions of envelope fields (Figure 2/X.411, Parts 16 to 19).

– *Extension fields:* definitions of extension-fields (Figure 2/X.411, Parts 20 to 28).

– *Common parameter types:* definitions of common parameter types (Figure 2/X.411, Parts 29 to 41).

Note 1 – The module implies a number of changes to the P3 protocol defined in Recommendation X.411 (1984). These changes are highlighted by means of underlining.

Note 2 – The module applies size constraints to variable-length data types using the SIZE subtyping extension of ASN.1. Violation of a size constraint constitutes a protocol violation.

9.1 *Criticality mechanism*

Each **extension-field** defined in Figure 2/X.411 (Parts 20 to 27) carries with it an indication of its **criticality** for submission, transfer and delivery. The criticality mechanism is designed to support controlled transparency of extended functions. A non-critical function may be ignored or discarded on delivery but shall not be discarded by a relaying MTA except when downgrading a message (see Recommendation X.419, Annex B), while a critical function must be known and performed correctly for normal procedure to continue.

In general, an argument of an abstract-operation marked critical for the port type shall be correctly handled by the performer of the abstract-operation, or an error reported in an appropriate way. The invoker of an abstract-operation shall also correctly handle any functions marked critical for the port type.

If the abstract-operation is one that reports an unsuccessful outcome, failure to correctly perform a critical function is reported by returning an unsupported-critical-function abstract-error. If an abstract-operation is not one that reports an unsuccessful outcome, an abstract-operation (e.g., a report) shall be invoked to convey the unsuccessful outcome of the previous operation (e.g., using the **unsupported-critical-function non-delivery diagnostic-code** of a report).

An extension that appears in the result of an abstract-operation shall not be marked critical for the port type.

In the case of **critical-for-submission**, the MTS shall correctly perform the procedures defined for a function marked as **critical-for-submission** in a message-submission or probe-submission abstract-operation, or shall return an unsupported-critical-function abstract-error.

In the case of **critical-for-transfer**, a receiving MTA shall correctly perform the procedures defined for a function in a message or probe marked as **critical-for-transfer**, or shall return a non-delivery-report with the **non-delivery-diagnostic-code** set to **unsupported-critical-function**. An MTA unable to support a function marked **critical-for-transfer** in a report shall discard the report (note that a local policy or agreement may require that this action be audited). An extension marked as **critical-for-transfer** that appears as an argument of a message-submission or probe-submission operation shall appear unchanged in a resulting message-transfer or probe-transfer operation at a transfer-port.

In the case of **critical-for-delivery**, a delivering-MTA shall correctly perform the procedures defined for a function marked **critical-for-delivery**, or shall not deliver the message or probe and shall return a non-delivery report with the **non-delivery-diagnostic-code** set to **unsupported-critical-function**. A recipient MTS-user shall correctly perform the procedures defined for a function marked as **critical-for-delivery** or shall return an unsupported-critical-function abstract-error. An extension marked as **critical-for-delivery** that appears as an argument of a message-submission or probe-submission operation shall appear unchanged in a resulting message-transfer or probe-transfer operation at a transfer port. An extension marked as **critical-for-delivery** that appears as an argument of a message-transfer or probe-transfer operation shall appear unchanged in any resulting message-transfer of probe-transfer operation at a transfer port.

An MTA generating a report shall not copy unsupported critical functions from the subject into the report. When generating a report, an MTA shall indicate the **criticality** (for transfer and/or delivery) of any supported functions copied from the subject into the report; the **criticality** of a function in a report may be different from its **criticality** in the subject.

If the MTA or MTS-user cannot correctly perform the procedures defined for a function marked “critical-for-delivery” in a report, then the report is discarded.

The procedures related to **extension-fields** and their **criticality** indications are further defined in § 14.

This Recommendation defines by means of the macro notation of ASN.1 the default setting of the **criticality** indication of **extension-fields** to be supplied by the originator of a message. The originator of a message or probe may choose, on a per-message basis, or in accordance with some local policy (e.g., a security-policy), to set the **criticality** indication of an extension-field to other than that defined in this Recommendation, either to relax or further constrain its **criticality**.

```
MTSAbstractService { joint-iso-ccitt mhs-motis(6) mts(3) modules(0) mts-abstract-service(1) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- Prologue
-- Exports everything


IMPORTS
        -- Abstract service macros
        OBJECT, PORT, ABSTRACT-BIND, ABSTRACT-UNBIND, ABSTRACT-OPERATION, ABSTRACT-ERROR
                FROM AbstractServiceNotation { joint-iso-ccitt mhs-motis(6) asdc(2) modules(0)
                    notation(1) }


        -- MS Abstract service extension
        forwarding-request
                FROM MSAbstractService { joint-iso-ccitt mhs-motis(6) ms(4) modules(0)
                    abstract-service(1) }


        -- Object identifiers
        id-ot-mts, id-ot-mts-user,
        id-pt-submission, id-pt-delivery, id-pt-administration,
        id-att-physicalRendition-basic,
        id-tok-asymmetricToken
                FROM MTSObjectifiers { joint-iso-ccitt mhs-motis(6) mts(3) modules(0)
                    object-identifiers(0) }


        -- Directory definitions
        Name
                FROM InformationFramework { joint-iso-ccitt ds(5) modules(1)
                    information-framework(1) }
        PresentationAddress
                FROM SelectedAttributeTypes { joint-iso-ccitt ds(5) modules(1)
                    selectedAttributeTypes(5) }
        Certificates, AlgorithmIdentifier, ALGORITHM, SIGNED, SIGNATURE, ENCRYPTED
                FROM AuthenticationFramework { joint-iso-ccitt ds(5) modules(1)
                    authentication-framework(7) }
```

FIGURE 2/X.411 (Part 1 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Upper bounds

ub-bit-options, ub-built-in-content-type, ub-built-in-encoded-information-types,
ub-common-name-length, ub-content-id-length, ub-content-length,
ub-content-types, ub-country-name-alpha-length, ub-country-name-numeric-length,
ub-dl-expansions, ub-domain-defined-attribute-value-length,
ub-domain-defined-attributes, ub-domain-defined-attribute-type-length,
ub-domain-name-length, ub-e163-4-number-length, ub-e163-4-subaddress-length,
ub-encoded-information-types, ub-extension-attributes, ub-extension-types,
ub-generation-qualifier-length, ub-given-name-length, ub-initials-length,
ub-integer-options, ub-labels-and-redirections, ub-local-id-length,
ub-mta-name-length, ub-mts-user-types, ub-numeric-user-id-length,
ub-organization-name-length, ub-organizational-unit-name-length,
ub-organizational-units, ub-password-length, ub-pds-name-length,
ub-pds-parameter-length, ub-pds-physical-address-lines, ub-postal-code-length, ub-privacy-mark-length,
ub-queue-size, ub-reason-codes, ub-recipients,
ub-recipient-number-for-advice-length, ub-redirections, ub-security-categories,
ub-security-labels, ub-security-problems, ub-supplementary-info-length,
ub-surname-length, ub-terminal-id-length, ub-tsap-id-length,
ub-informated-address-length, ub-x121-address-length

        FROM MTSUpperBounds { joint-iso-ccitt mhs-motis(6) mts(3) modules(0)
                upper-bounds(3) };
```

FIGURE 2/X.411 (Part 1 *bis* of 41)

Abstract syntax definition of the MTS abstract service

```
-- Objects

mTS OBJECT
        PORTS { submission [S], delivery [S], administration [S] }
        ::= id-ot-mts

mTSUser OBJECT
        PORTS { submission [C], delivery [C], administration [C] }
        ::= id-ot-mts-user


-- Ports

submission PORT
        CONSUMER INVOKES { MessageSubmission, ProbeSubmission, CancelDeferredDelivery }
        SUPPLIER INVOKES { SubmissionControl }
        ::= id-pt-submission

delivery PORT
        CONSUMER INVOKES { DeliveryControl }
        SUPPLIER INVOKES { MessageDelivery, ReportDelivery }
        ::= id-pt-delivery

administration PORT
        CONSUMER INVOKES { ChangeCredentials, Register }
        SUPPLIER INVOKES { ChangeCredentials }
        ::= id-pt-administration
```

FIGURE 2/X.411 (Part 2 of 41)

Abstract syntax definition of the MTS abstract service

```
-- MTS-bind and MTS-unbind

MTSBind ::= ABSTRACT-BIND
        TO { submission, delivery, administration }
        BIND
        ARGUMENT SET {
                   initiator-name ObjectName,
                   messages-waiting [1] EXPLICIT MessagesWaiting OPTIONAL,
                   initiator-credentials [2] InitiatorCredentials,
                   security-context [3] SecurityContext OPTIONAL }
        RESULT SET {
                   responder-name ObjectName,
                   messages-waiting [1] EXPLICIT MessagesWaiting OPTIONAL,
                   responder-credentials [2] ResponderCredentials }
        BIND-ERROR INTEGER {
                   busy (0)
                   authentication-error (2),
                   unacceptable-dialogue-mode (3),
                   unacceptable-security-context (4) } (0. .ub-integer-options)

MTSUnbind ::=ABSTRACT-UNBIND
        FROM { submission, delivery, administration }


-- Association control parameters

ObjectName ::= CHOICE {
        mTS-userORAddressAndOptionalDirectoryName,
        mTA [0] MTAName,
        message-store [4] ORAddressAndOptionalDirectoryName }

MessagesWaiting ::= SET {
        urgent [0] DeliveryQueue,
        normal [1] DeliveryQueue,
        non-urgent [2] DeliveryQueue }

DeliveryQueue ::= SET {
        messages [0] INTEGER (0. .ub-queue-size),
        octets [1] INTEGER (0. .ub-content-length) OPTIONAL }

InitiatorCredentials ::= CHOICE {
        simple Password,
        strong [0] StrongCredentials (WITH COMPONENTS {
                   . . .,
                   bind-token PRESENT }) }
```

FIGURE 2/X.411 (Part 3 of 41)

Abstract syntax definition of the MTS abstract service

```
ResponderCredentials ::= CHOICE {
        simple Password,
        strong [0] StrongCredentials (WITH COMPONENTS {
                   bind-token }) }

Password ::= CHOICE {
        IA5String (SIZE (0. .ub-password-length))
        OCTET STRING (SIZE (0. .ub-password-length)) }

StrongCredentials ::= SET {
        bind-token [0] Token OPTIONAL,
        certificate [1] Certificates OPTIONAL }

Security Context ::= SET SIZE (1. .ub-security-labels) OF SecurityLabel
```

FIGURE 2/X.411 (Part 4 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Submission port

MessageSubmission ::= ABSTRACT-OPERATION
        ARGUMENT SEQUENCE {
                envelope MessageSubmissionEnvelope,
                content Content }
        RESULT SET {
                message-submission-identifier MessageSubmissionIdentifier,
                message-submission-time [0] MessageSubmissionTime,
                content-identifier ContentIdentifier OPTIONAL,
                extensions [1] EXTENSIONS CHOSEN FROM {
                    originating-MTA-certificate,
                    proof-of-submission } DEFAULT { } }

        ERRORS {
                SubmissionControlViolated,
                ElementOfServiceNotSubscribed,
                OriginatorInvalid,
                RecipientImproperlySpecified,
                InconsistentRequest,
                SecurityError,
                UnsupportedCriticalFunction,
                RemoteBindError }

ProbeSubmission ::= ABSTRACT-OPERATION
        ARGUMENT
                envelope ProbeSubmissionEnvelope
        RESULT SET {
                probe-submission-identifier ProbeSubmissionIdentifier,
                probe-submission-time [0] ProbeSubmissionTime,
                content-identifier ContentIdentifier OPTIONAL }
        ERRORS {
                SubmissionControlViolated,
                ElementOfServiceNotSubscribed,
                OriginatorInvalid,
                RecipientImproperlySpecified,
                InconsistentRequest,
                SecurityError,
                UnsupportedCriticalFunction,
                RemoteBindError }

CancelDeferredDelivery :: ABSTRACT-OPERATION
        ARGUMENT
                message-submission-identifier MessageSubmissionIdentifier
        RESULT
        ERRORS {
                DeferredDeliveryCancellationRejected,
                MessageSubmissionIdentifierInvalid,
                RemoteBindError }
```

FIGURE 2/X.411 (Part 5 of 41)

Abstract syntax definition of the MTS abstract service

```
SubmissionControl ::= ABSTRACT-OPERATION
        ARGUMENT
                  controls SubmissionControls
        RESULT
                  waiting Waiting
        ERRORS {
                  SecurityError,
                  RemoteBindError }

SubmissionControlViolated ::= ABSTRACT-ERROR
        PARAMETER   NULL

ElementOfServiceNotSubscribed ::= ABSTRACT-ERROR
        PARAMETER   NULL

DeferredDeliveryCancellationRejected ::= ABSTRACT-ERROR
        PARAMETER   NULL

OriginatorInvalid ::= ABSTRACT-ERROR
        PARAMETER   NULL

RecipientImproperlySpecified ::= ABSTRACT-ERROR
        PARAMETER
                  improperly-specified-recipients SEQUENCE SIZE (1. .ub-recipients OF
                      ORAddressAndOptionalDirectoryName

MessageSubmissionIdentifierInvalid ::= ABSTRACT-ERROR
        PARAMETER   NULL

InconsistentRequest ::= ABSTRACT-ERROR
        PARAMETER   NULL

SecurityError ::= ABSTRACT-ERROR
        PARAMETER
                  security-problem SecurityProblem

SecurityProblem ::= INTEGER (0. .ub-security-problems)

UnsupportedCriticalFunction ::= ABSTRACT-ERROR
        PARAMETER   NULL

RemoteBindError ::= ABSTRACT-ERROR
        PARAMETER   NULL
```

FIGURE 2/X.411 (Part 6 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Submision port parameters

MessageSubmissionIdentifier ::= MTSIdentifier

MessageSubmissionTime ::= Time

ProbeSubmissionIdentifier ::= MTSIdentifier

ProbeSubmissionTime ::= Time

SubmissionControls ::= Controls (WITH COMPONENTS {
        permissible-content-types ABSENT
        permissible-encoded-information-types ABSENT })

Waiting ::= SET {
        waiting-operations [0] Operations DEFAULT { },
        waiting-messages [1] WaitingMessages DEFAULT { },
        waiting-content-types [2] SET SIZE (0. .ub-content-types) OF ContentType DEFAULT { },
        waiting-encoded-information-types EncodedInformationTypes OPTIONAL }

Operations ::= BIT STRING {
        probe-submission-or-report-delivery (0),
        message-submission-or-message-delivery (1) } (SIZE (0. .ub-bit-options))
        --holding 'one', not-holding 'zero'.

WaitingMessages ::= BIT STRING {
        long-content (0),
        low-priority (1),
        other-security-labels (2) } (SIZE (0. .ub-bit-options))
```

FIGURE 2/X.411 (Part 7 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Delivery port

MessageDelivery ::= ABSTRACT-OPERATION
        ARGUMENT SEQUENCE {
                  COMPONENTS OF MessageDeliveryEnvelope,
                  content Content }
        RESULT SET {
                  recipient-certificate [0] RecipientCertificate OPTIONAL,
                  proof-of-delivery [1] ProofOfDelivery OPTIONAL } DEFAULT{ }
        ERRORS {
                  DeliveryControlViolated,
                  SecurityError,
                  UnsupportedCriticalFunction }

ReportDelivery ::= ABSTRACT-OPERATION
        ARGUMENT SET {
                  COMPONENTS OF ReportDeliveryEnvelope,
                  returned-content [0] Content OPTIONAL }
        RESULT
        ERRORS {
                  DeliveryControlViolated,
                  SecurityError,
                  UnsupportedCriticalFunction }

DeliveryControl ::= ABSTRACT-OPERATION
        ARGUMENT
                  controls DeliveryControls
        RESULT
                  waiting Waiting
        ERRORS {
                  ControlViolatesRegistration,
                  SecurityError }

DeliveryControlViolated ::= ABSTRACT-ERROR
        PARAMETER  NULL

ControlViolatesRegistration ::= ABSTRACT-ERROR
        PARAMETER  NULL


-- SecurityError — defined in Figure 2/X.411, Part 6 of 41

-- UnsupportedCriticalFunction — defined in Figure 2/X.411, Part 6 of 41
```

FIGURE 2/X.411 (Part 8 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Delivery port parameters

RecipientCertificate ::= Certificates

ProofOfDelivery ::= SIGNATURE SEQUENCE {
        algorithm-identifier ProofOfDeliveryAlgorithmIdentifier,
        delivery-timeMessageDeliveryTime,
        this-recipient-name ThisRecipientName,
        originally-intended-recipient-name OriginallyIntendedRecipientName OPTIONAL,
        content Content,
        content-identifier ContentIdentifier OPTIONAL,
        message-security-label MessageSecurityLabel OPTIONAL }

ProofOfDeliveryAlgorithmIdentifier ::= AlgorithmIdentifier

DeliveryControls ::= Controls

Controls ::= SET {
        restrict [0] BOOLEAN DEFAULT TRUE,
        -- update 'TRUE', remove 'FALSE'
        permissible-operations [1] Operations OPTIONAL,
        permissible-maximum-content-length [2] ContentLength OPTIONAL,
        permissible-lowest-priority Priority OPTIONAL,
        permissible-content-types [4] SET SIZE (1..ub-content-types) OF ContentType OPTIONAL,
        permissible-encoded-information-types EncodedInformationTypes OPTIONAL,
        permissible-security-context [5] SecurityContext OPTIONAL }

-- Note – The tags [0], [1] and [2] are altered for the register operation only.
```

FIGURE 2/X.411 (Part 9 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Administration port

Register ::= ABSTRACT-OPERATION
        ARGUMENT SET {
                  user-name UserName OPTIONAL,
                  user-address [0] UserAddress OPTIONAL,
                  deliverable-encoded-information-types EncodedInformationTypes OPTIONAL,
                  deliverable-maximum-content-length [1] EXPLICIT ContentLength OPTIONAL,
                  default-delivery-controls [2] EXPLICIT DefaultDeliveryControls OPTIONAL,
                  deliverable-content-types [3] SET SIZE (1 . .ub-content-types) OF ContentType OPTIONAL,
                  labels-and-redirections [4] SET SIZE (1. .ub-labels-and-redirections) OF
                      LabelAndRedirection OPTIONAL }
        RESULT
        ERRORS {
                  RegisterRejected }

ChangeCredentials ::= ABSTRACT-OPERATION
        ARGUMENT SET {
                  old-credentials [0] Credentials,
                  new-credentials [1] Credentials -- same CHOICE as for old-credentials -- }
        RESULT
        ERRORS {
                  NewCredentialsUnacceptable,
                  OldCredentialsIncorrectlySpecified }

RegisterRejected ::= ABSTRACT-ERROR
        PARAMETER   NULL

NewCredentialsUnacceptable ::= ABSTRACT-ERROR
        PARAMETER   NULL

OldCredentialsIncorrectlySpecified ::= ABSTRACT-ERROR
        PARAMETER   NULL
```

FIGURE 2/X.411 (Part 10 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Administration port parameters

UserName ::= ORAddressAndOptionalDirectoryName

UserAddress ::= CHOICE {
        x121 [0] SEQUENCE {
                x121-address NumericString (Size (1..ub-x121-address-length)) OPTIONAL,
                tsap-id PrintableString (SIZE (1..ub-tsap-id-length)) OPTIONAL },
        presentation [1] PSAPAddress }

PSAPAddress ::= PresentationAddress

DefaultDeliveryControls ::= Controls (WITH COMPONENTS {
        ...,
        permissible-security-context ABSENT })

Credentials ::= CHOICE {
        simple Password,
        strong [0] StrongCredentials (WITH COMPONENTS {
                certificate }) }

LabelAndRedirection ::= SET {
        user-security-label [0] UserSecurityLabel OPTIONAL,
        recipient-assigned-alternate-recipient [1] RecipientAssignedAlternateRecipient OPTIONAL }

UserSecurityLabel ::= SecurityLabel

RecipientAssignedAlternateRecipient ::= ORAddressAndOptionalDirectoryName
```

FIGURE 2/X.411 (Part 11 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Message submission envelope

MessageSubmissionEnvelope ::= SET {
        COMPONENTS OF PerMessageSubmissionFields,
        per-recipient-fields [1] SEQUENCE SIZE (1..ub-recipients) OF
                PerRecipientMessageSubmissionFields }

PerMessageSubmissionFields ::= SET {
        originator-name OriginatorName,
        original-encoded-information-types OriginalEncodedInformationTypes OPTIONAL,
        content-type ContentType,
        content-identifier ContentIdentifier OPTIONAL,
        priority Priority DEFAULT normal,
        per-message-indicators PerMessageIndicators DEFAULT { },
        deferred-delivery-time [0] DeferredDeliveryTime OPTIONAL,
        extensions [2] PerMessageSubmissionExtensions DEFAULT { } }

PerMessageSubmissionExtensions ::= EXTENSIONS CHOSEN FROM {
        recipient-reassignment-prohibited,
        dl-expansion-prohibited,
        conversion-with-loss-prohibited,
        latest-delivery-time,
        originator-return-address,
        originator-certificate,
        content-confidentiality-algorithm-identifier,
        message-origin-authentication-check,
        message-security-label,
        proof-of-submission-request,
        content-correlator,
        forwarding-request -- for MS Abstract Service only -- }

PerRecipientMessageSubmissionFields ::= SET {
        recipient-name RecipientName,
        originator-report-request [0] OriginatorReportRequest,
        explicit-conversion [1] ExplicitConversion OPTIONAL,
        extensions [2] PerRecipientMessageSubmissionExtensions DEFAULT { } }

PerRecipientMessageSubmissionExtensions ::= EXTENSIONS CHOSEN FROM {
        originator-requested-alternate-recipient,
        requested-delivery-method,
        physical-forwarding-prohibited,
        physical-forwarding-address-request,
        physical-delivery-modes,
        registered-mail-type,
        recipient-number-for-advice,
        physical-rendition-attributes,
        physical-delivery-report-request,
        message-token,
        content-integrity-check,
        proof-of-delivery-request }
```

FIGURE 2/X.411 (Part 12 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Probe submission envelope

ProbeSubmissionEnvelope ::= SET {
        COMPONENTS OF PerProbeSubmissionFields,
        per-recipient-fields [3] SEQUENCE SIZE (1. .ub-recipients) OF
                PerRecipientProbeSubmissionFields }

PerProbeSubmissionFields ::= SET {
        originator-name OriginatorName,
        original-encoded-information-types OriginalEncodedInformationTypes OPTIONAL
        content-type ContentType,
        content-identifier ContentIdentifier OPTIONAL,
        content-length [0] ContentLength OPTIONAL,
        per-message-indicators PerMessageIndicators DEFAULT {},
        extensions [2] EXTENSIONS CHOSEN FROM {
                recipient-reassignment-prohibited,
                dl-expansion-prohibited,
                conversion-with-loss-prohibited,
                originator-certificate,
                message-security-label,
                content-correlator,
                probe-origin-authentication-check } DEFAULT {} }

PerRecipientProbeSubmissionFields ::= SET {
        recipient-name RecipientName,
        originator-report-request [0] OriginatorReportRequest,
        explicit-conversion [1] ExplicitConversion OPTIONAL,
        extensions [2] EXTENSIONS CHOSEN FROM {
                originator-requested-alternate-recipient,
                requested-delivery-method
                physical-rendition-attributes } DEFAULT {} }
```

FIGURE 2/X.411 (Part 13 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Message delivery envelope

MessageDeliveryEnvelope ::= SEQUENCE {
        message-delivery-identifier MessageDeliveryIdentifier,
        message-delivery-time MessageDeliveryTime,
        other-fields OtherMessageDeliveryFields }

OtherMessageDeliveryFields ::= SET {
        content-type DeliveredContentType,
        originator-name OriginatorName,
        original-encoded-information-types [1] OriginalEncodedInformationTypes OPTIONAL,
        priority Priority DEFAULT normal,
        delivery-flags [2] DeliveryFlags OPTIONAL,
        other-recipient-names [3] OtherRecipientNames OPTIONAL,
        this-recipient-name [4] ThisRecipientName,
        originally-intended-recipient-name [5] OriginallyIntendedRecipientName OPTIONAL,
        converted-encoded-information-types [6] ConvertedEncodedInformationTypes OPTIONAL,
        message-submission-time [7] MessageSubmissionTime,
        content-identifier [8] ContentIdentifier OPTIONAL,
        extensions [9] EXTENSIONS CHOSEN FROM {
                conversion-with-loss-prohibited,
                requested-delivery-method,
                physical-forwarding-prohibited,
                physical-forwarding-address-request,
                physical-delivery-modes,
                registered-mail-type,
                recipient-number-for-advice,
                physical-rendition-attributes,
                originator-return-address,
                physical-delivery-report-request,
                originator-certificate,
                message-token,
                content-confidentiality-algorithm-identifier,
                content-integrity-check,
                message-origin-authentication-check,
                message-security-label,
                proof-of-delivery-request,
                redirection-history,
                dl-expansion-history } DEFAULT { } }
```

FIGURE 2/X.411 (Part 14 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Report delivery envelope

ReportDeliveryEnvelope ::= SET {
        COMPONENTS OF PerReportDeliveryFields,.
        per-recipient-fields SEQUENCE SIZE (1. .ub-recipients) OF PerRecipientReportDeliveryFields }

PerReportDeliveryFields ::= SET {
        subject-submission-identifier SubjectSubmissionIdentifier,
        content-identifier ContentIdentifer OPTIONAL,
        content-type ContentType OPTIONAL,
        original-encoded-information-types OriginalEncodedInformationTypes OPTIONAL,
        extension [1] EXTENSIONS CHOSEN FROM {
                message-security-label,
                content-correlator,
                originator-and-DL-expansion-history,
                reporting-DL-name,
                reporting-MTA-certificate,
                report-origin-authentication-check } DEFAULT { } }

PerRecipientReportDeliveryFields ::= SET {
        actual-recipient-name [0] ActualRecipientName,
        report-type [1] ReportType,
        converted-encoded-information-types ConvertedEncodedInformationTypes OPTIONAL,
        originally-intended-recipient-name [2] OriginallyIntendedRecipientName OPTIONAL,
        supplementary-information [3] SupplementaryInformation OPTIONAL,
        extensions [4] EXTENSIONS CHOSEN FROM {
                redirection-history,
                physical-forwarding-address,
                recipient-certificate,
                proof-of-delivery } DEFAULT { } }

ReportType ::= CHOICE {
        delivery [0] DeliveryReport,
        non-delivery [1] NonDeliveryReport }

DeliveryReport ::= SET {
        message-delivery-time [0] MessageDeliveryTime,
        type-of-MTS-user [1] TypeOfMTSUser DEFAULT public }

NonDeliveryReport ::= SET {
        non-delivery-reason-code [0] NonDeliveryReasonCode,
        non-delivery-diagnostic-code [1] NonDeliveryDiagnosticCode OPTIONAL }
```

FIGURE 2/X.411 (Part 15 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Envelope fields

OriginatorName :: = ORAddressAndOrDirectoryName

OriginalEncodedInformationTypes ::= EncodedInformationTypes

ContentType ::= CHOICE {
        built-in BuiltInContentType,
        external ExternalContentType }

BuiltInContentType ::= [APPLICATION 6] INTEGER {
        unidentified (0),
        external (1),                          -- identified by the object-identifier of the EXTERNAL content
        interpersonal-messaging-1984 (2),
        interpersonal-messaging-1988 (22) } (0. .ub-built-in-content-type)

ExternalContentType ::= OBJECT IDENTIFIER

DeliverContentType ::= CHOICE {
        built-in [0] BuiltInContentType,
        external ExternalContentType }

ContentIdentifier ::= [APPLICATION 10] PrintableString (SIZE (1. .ub-content-id-length))

PerMessageIndicators ::= [APPLICATION 8] BIT STRING {
        disclosure-of-recipients (0),          -- disclosure-of-recipients-allowed 'one',
                                               -- disclosure-of-recipient-prohibited 'zero';
                                               -- ignored for Probe-submission
        implicit-conversion-prohibited (1),    -- implicit-conversion-prohibited 'one';
                                               -- implicit-conversion-allowed 'zero'
        alternate-recipient-allowed (2),       -- alternate-recipient-allowed 'one',
                                               -- alternate-recipient-prohibited 'zero'
        content-return-request (3)             -- content-return-requested 'one',
                                               -- content-return-not-requested 'zero';
                                               -- ignored for Probe-submission -- }
        (SIZE (0. .ub-bit-options))

RecipientName ::= ORAddressAndOrDirectoryName

OriginatorReportRequest ::= BIT STRING {
        report (3),
        non-delivery-report (4)
        -- at most one bit shall be 'one':
        -- report bit 'one' requests a 'report';
        -- non-delivery-report bit 'one' requests a 'non-delivery-report';
        -- both bits 'zero' requests 'no-report' -- } (SIZE (0. .ub-bit-options))
```

FIGURE 2/X.411 (Part 16 of 41)

Abstract syntax definition of the MTS abstract service

```
ExplicitConversion ::= INTEGER {
        ia5-text-to-teletex (0),
        teletex-to-telex (1),
        telex-to-ia5-text (2),
        telex-to-teletex (3),
        telex-to-g4-class-1 (4),
        telex-to-videotex (5),
        ia5-text-to-telex (6),
        telex-to-g3-facsimile (7),
        ia5-text-to-g3-facsimile (8),
        ia5-text-to-g4-class-1 (9),
        ia5-text-to-videotex (10),
        teletex-to-ia5-text (11),
        teletex-to-g3-facsimile (12),
        teletex-to-g4-class-1 (13),
        teletex-to-videotex (14),
        videotex-to-telex (15),
        videotex-to-ia5-text (16),
        videotex-to-teletex (17) } (0. .ub-integer-options)

DeferredDeliveryTime ::= Time

Priority ::= [APPLICATION 7] ENUMERATED {
        normal (0),
        non-urgent (1),
        urgent (2) }

ContentLength ::= INTEGER (0. .ub-content-length)

MessageDeliveryIdentifier ::= MTSIdentifier

MessageDeliveryTime ::= Time

DeliveryFlags ::= BIT STRING {
        implicit-conversion-prohibited (1)          -- implicit-conversion-prohibited 'one',
                                                    -- implicit-conversion-allowed 'zero' -- }
        (SIZE (0. .ub-bit-options))

OtherRecipientNames ::= SEQUENCE SIZE (1. .ub-recipients) OF OtherRecipientName

OtherRecipientName ::= OrAddressAndOrDirectoryName

ThisRecipientName ::= OrAddressAndOrDirectoryName

OriginallyIntendedRecipientName ::= OrAddressAndOrDirectoryName
```

FIGURE 2/X.411 (Part 17 of 41)

Abstract syntax definition of the MTS abstract service

```
ConvertedEncodedInformationTypes ::= EncodedInformationTypes

SubjectSubmissionIdentifier ::= MTSIdentifier

ActualRecipientName ::= ORAddressAndOrDirectoryName

TypeOfMTSUser ::= INTEGER {
        public (0),
        private (1),
        ms (2),
        dl (3),
        pdau (4),
        physical-recipient (5),
        other (6) } (0. .ub-mts-user-types)

NonDeliveryReasonCode ::= INTEGER {
        transfer-failure (0),
        unable-to-transfer (1),
        conversion-not-performed (2),
        physical-rendition-not-performed (3),
        physical-delivery-not-performed (4),
        restricted-delivery (5),
        directory-operation-unsuccessful (6) } (0. .ub-reason-codes)

NonDeliveryDiagnosticCode ::= INTEGER {
        unrecognised-OR-name (0),
        ambiguous-OR-name (1),
        mts-congestion (2),
        loop-detected (3),
        recipient-unavailable (4),
        maximum-time-expired (5),
        encoded-information-types-unsupported (6),
        content-too-long (7),
        conversion-impratical (8),
        implicit-conversion-prohibited (9),
        implicit-conversion-not-subscribed (10),
        invalid-arguments (11),
        content-syntax-error (12),
        size-constraint-violation (13),
        protocol-violation (14),
        content-type-not-supported (15),
        too-many-recipients (16),
        no-bilateral-agreement (17),
        unsupported-critical-function (18),

        -- continued
```

FIGURE 2/X.411 (Part 18 of 41)

Abstract syntax definition of the MTS abstract service

```
    -- continued

conversion-with-loss-prohibited (19),
line-too-long (20),
page-split (21),
pictorial-symbol-loss (22),
punctuation-symbol-loss (23),
alphabetic-character-loss (24),
multiple-information-loss (25),
recipient-reassignment-prohibited (26),
redirection-loop-detected (27),
dL-expansion-prohibited (28),
no-DL-submit-permission (29),
dl-expansion-failure (30),
physical-rendition-attributes-not-supported (31),
undeliverable-mail-physical-delivery-address-incorrect (32),
undeliverable-mail-physical-delivery-office-incorrect-or-invalid (33),
undeliverable-mail-physical-delivery-address-incomplete (34),
undeliverable-mail-recipient-unknown (35),
undeliverable-mail-recipient-deceased (36),
undeliverable-mail-organization-expired (37),
undeliverable-mail-recipient-refused-to-accept (38),
undeliverable-mail-recipient-did-not-claim (39),
undeliverable-mail-recipient-changed-address-permanently (40),
undeliverable-mail-recipient-changed-address-temporarily (41),
undeliverable-mail-recipient-changed-temporary-address (42),
undeliverable-mail-new-address-unknown (43),
undeliverable-mail-recipient-did-not-want-forwarding (44),
undeliverable-mail-originator-prohibited-forwarding (45),
secure-messaging-error (46),
unable-to-downgrade (47) } (0. .ub-diagnostic-codes)
SupplementaryInformation ::= PrintableString (SIZE (1. .ub-supplementary-info-length))
```

FIGURE 2/X.411 (Part 19 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Extension fields

ExtensionField ::= SEQUENCE {
        type [0] EXTENSION,
        criticality [1] Criticality DEFAULT { },
        value [2] AND DEFINED BY type DEFAULT NULL NULL }

Criticality ::= BIT STRING {
        for-submission (0),
        for-transfer (1),
        for-delivery (2) } (SIZE (0. .ub-bit-options))        -- critical 'one', non-critical 'zero'

EXTENSIONS MACRO ::=
BEGIN

TYPE NOTATION ::= "CHOSEN FROM" "{" ExtensionList "}"
VALUE NOTATION ::= Value (VALUE SET OF ExtensionField      -- each of a different type --)

ExtensionList ::= Extension "," ExtensionList | Extension | empty
Extension ::= value (EXTENSION)

END  -- of EXTENSIONS

EXTENSION MACRO ::=
BEGIN

TYPE NOTATION ::= DataType Critical | empty
VALUE NOTATION ::= value (VALUE ExtensionType)

DataType ::= type (X) Default | empty
Default ::= "DEFAULT" value (X) | empty
Critical ::= "CRITICAL FOR" CriticalityList | empty
CriticalityList ::= Criticality | CriticalityList "," Criticality
Criticality ::= "SUBMISSION" | "TRANSFER" | "DELIVERY"

END  -- of EXTENSION

ExtensionType ::= INTEGER (0. .ub-extension-types)

recipient-reassignment-prohibited EXTENSION
        RecipientReassignmentProhibited DEFAULT recipient-reassignment-allowed
        CRITICAL FOR DELIVERY
        ::= 1
```

FIGURE 2/X.411 (Part 20 of 41)

Abstract syntax definition of the MTS abstract service

```
RecipientReassignmentProhibited ::= ENUMERATED {
        recipient-reassignment-allowed (0),
        recipient-reassignment-prohibited (1) }

originator-requested-alternate-recipient EXTENSION
        OriginatorRequestedAlternateRecipient
        CRITICAL FOR SUBMISSION
        ::= 2

OriginatorRequestedAlternateRecipient ::= ORAddressAndOrDirectoryName

-- OriginatorRequestedAlternateRecipient as defined here differs from the
-- field of the same name in Figure 4/X.411, since, on submission the
-- OR-address need not be present, but on transfer the OR-address
-- must be present.

dl-expansion-prohibited EXTENSION
        DLExpansionProhibited DEFAULT dl-expansion-allowed
        CRITICAL FOR DELIVERY
        ::= 3

DLExpansionProhibited ::= ENUMERATED {
        dl-expansion-allowed (0),
        dl-expansion-prohibited (1) }

conversion-with-loss-prohibited EXTENSION
        ConversionWithLossProhibited DEFAULT conversion-with-loss-allowed
        CRITICAL FOR DELIVERY
        ::= 4

ConversionWithLossProhibited ::= ENUMERATED {
        conversion-with-loss-allowed (0),
        conversion-with-loss-prohibited (1) }

latest-delivery-time EXTENSION
        LatestDeliveryTime
        CRITICAL FOR DELIVERY
        ::= 5

LatestDeliveryTime ::= Time

requested-delivery-method EXTENSION
        RequestedDeliveryMethod DEFAULT any-delivery-method
        CRITICAL FOR DELIVERY
        ::= 6
```

FIGURE 2/X.411 (Part 21 of 41)

Abstract syntax definition of the MTS abstract service

```
RequestedDeliveryMethod ::= SEQUENCE OF INTEGER { -- each different in order of preference,
                                                    most preferred first

        any-delivery-method (0),
        mhs-delivery (1),
        physical-delivery (2),
        telex-delivery (3),
        teletex-delivery (4),
        g3-facsimile-delivery (5),
        g4-facsimile-delivery (6),
        ia5-terminal-delivery (7),
        videotex-delivery (8),
        telephone-delivery (9) }  (0. .ub-integer-options)

physical-forwarding-prohibited EXTENSION
        PhysicalForwardingProhibited DEFAULT physical-forwarding-allowed
        CRITICAL FOR DELIVERY
        ::= 7

PhysicalForwardingProhibited ::= ENUMERATED {
        physical-forwarding-allowed (0),
        physical-forwarding-prohibited (1) }

physical-forwarding-address-request EXTENSION
        PhysicalForwardingAddressRequest DEFAULT physical-forwarding-address-not-requested
        CRITICAL FOR DELIVERY
        ::= 8

PhysicalForwardingAddressRequest ::= ENUMERATED {
        physical-forwarding-address-not-requested (0),
        physical-forwarding-address-requested (1) }

physical-delivery-modes EXTENSION
        PhysicalDeliveryModes DEFAULT ordinary-mail
        CRITICAL FOR DELIVERY
        ::= 9

PhysicalDeliveryModes ::= BIT STRING {
        ordinary-mail (0),
        special-delivery (1),
        express-mail (2),
        counter-collection (3),
        counter-collection-with-telephone-advice (4),
        counter-collection-with-telex-advice (5),
        counter-collection-with-teletex-advice (6),
        bureau-fax-delivery (7)
        -- bits 0 to 6 are mutually exclusive
        -- bit 7 can be set with any of bits 0 to 6 -- } (SIZE (0. .ub-bit-options))
```

FIGURE 2/X.411 (Part 22 of 41)

Abstract syntax definition of the MTS abstract service

```
registered-mail-type EXTENSION
        RegisteredMailType DEFAULT non-registered-mail
        CRITICAL FOR DELIVERY
        ::= 10

RegisteredMailType ::= INTEGER
        non-registered-mail (0),
        registered-mail (1),
        registered-mail-to-addresse-in-person (2) } (0. .ub-integer-options)

recipient-number-for-advice EXTENSION
        RecipientNumberForAdvice
        CRITICAL FOR DELIVERY
        ::= 11

RecipientNumberForAdvice ::= TeletexString (SIZE (1. .ub-recipient-number-for-advice-length)

physical-rendition-attributes EXTENSION
        PhysicalRenditionAttributes DEFAULT id-att-physicalRendition-basic
        CRITICAL FOR DELIVERY
        :: = 12

PhysicalRenditionAttributes ::= OBJECT IDENTIFIER

originator-return-address EXTENSION
        OriginatorReturnAddress
        CRITICAL FOR DELIVERY
        ::= 12

OriginatorReturnAddress ::= ORAddress

physical-delivery-report-request EXTENSION
        PhysicalDeliveryReportRequest DEFAULT return-of-undeliverable-mail-by-PDS
        CRITICAL FOR DELIVERY
        ::= 14

PhysicalDeliveryReportRequest::= INTEGER {
        return-of-undeliverable-mail-by-PDS (0),
        return-of-notification-by-PDS (1),
        return-of-notification-by-MHS (2),
        return-of-notification-by-MHS-and-PDS (3) } (0. .ub-integer-options)
```

FIGURE 2/X.411 (Part 23 of 41)

Abstract syntax definition of the MTS abstract service

```
originator-certificate EXTENSION
        OriginatorCertificate
        CRITICAL FOR DELIVERY
        ::= 15

OriginatorCertificate ::= Certificates

message-token EXTENSION
        MessageToken
        ::= 16

MessageToken ::= Token

content-confidentiality-algorithm-identifier EXTENSION
        ContentConfidentialityAlgorithmIdentifier
        ::= 17

ContentConfidentialityAlgorithmIdentifier ::= AlgorithmIdentifier

content-integrity-check EXTENSION
        ContentIntegrityCheck
        ::= 18

ContentIntegrityCheck ::= SIGNATURE SEQUENCE {
        algorithm-identifier ContentIntegrityAlgorithmIdentifier,
        content Content }

ContentIntegrityAlgorithmIdentifier ::= AlgorithmIdentifier

message-origin-authentication-check EXTENSION
        MessageOriginAuthenticationCheck
        CRITICAL FOR DELIVERY
        ::= 19

MessageOriginAuthenticationCheck ::= SIGNATURE SEQUENCE {
        algorithm-identifier MessageOriginAuthenticationAlgorithmIdentifier,
        content Content
        content-identifier ContentIdentifier OPTIONAL,
        message-security-label MessageSecurityLabel OPTIONAL }

MessageOriginAuthenticationAlgorithmIdentifier ::= AlgorithmIdentifier
```

FIGURE 2/X.411 (Part 24 of 41)

Abstract syntax definition of the MTS abstract service

```
message-security-label EXTENSION
        MessageSecurityLabel
        CRITICAL FOR DELIVERY
        ::= 20

MessageSecurityLabel ::= SecurityLabel

proof-of-submission-request EXTENSION
        ProofOfSubmissionRequest DEFAULT proof-of-submission-not-requested
        CRITICAL FOR SUBMISSION
        ::= 21

ProofOfSubmissionRequest ::= ENUMERATED {
        proof-of-submission-not-requested (0),
        proof-of-submission-requested (1) }

proof-of-delivery-request EXTENSION
        ProofOfDeliveryRequest DEFAULT proof-of-delivery-not-requested
        CRITICAL FOR DELIVERY
        ::= 22

ProofOfDeliveryRequest ::= ENUMERATED {
        proof-of-delivery-not-requested (0),
        proof-of-delivery-requested (1) }

content-correlator EXTENSION
        ContentCorrelator
        ::= 23

ContentCorrelator ::= ANY       -- maximum ub-content-correlator-length octets including all encoding

probe-origin-authentication-check EXTENSION
        ProbeOriginAuthenticationCheck
        CRITICAL FOR DELIVERY
        ::= 24

ProbeOriginAuthenticationCheck ::= SIGNATURE SEQUENCE {
        algorithm-identifier ProbeOriginAuthenticationAlgorithmIdentifier,
        content-identifier ContentIdentifier OPTIONAL,
        message-security-label MessageSecurityLabel OPTIONAL }

ProbeOriginAuthenticationAlgorithmIdentifier ::= AlgorithmIdentifier
```

FIGURE 2/X.411 (Part 25 of 41)

Abstract syntax definition of the MTS abstract service

```
redirection-history EXTENSION
        RedirectionHistory
        ::= 25

RedirectionHistory ::= SEQUENCE SIZE (1. .ub-redirections) OF Redirection

Redirection ::= SEQUENCE {
        intended-recipient-name IntendedRecipientName,
        redirection-reason RedirectionReason }

IntendedRecipientName ::= SEQUENCE {
        OrAddressAndOptionalDirectoryName,
        redirection-time Time }

RedirectionReason ::= ENUMERATED {
        recipient-assigned-alternate-recipient (0),
        originator-requested-alternate-recipient (1),
        recipient-MD-assigned-alternate-recipient (2) }

dl-expansion-history EXTENSION
        DLExpansionHistory
        ::= 26

DLExpansionHistory ::= SEQUENCE SIZE (1. .ub-dl-expansions) OF DLExpansion

DLExpansion ::= SEQUENCE {
        ORAddressAndOptionalDirectoryName,
        dl-expansion-time Time }

physical-forwarding-address EXTENSION
        PhysicalForwardAddress
        ::= 27

PhysicalForwardingAddress ::= ORAddressAndOptionalDirectoryName

recipient-certificate EXTENSION
        RecipientCertificate
        ::= 28

proof-of-delivery EXTENSION
        ProofOfDelivery
        ::= 29

originator-and-DL-expansion-history EXTENSION
        OriginatorAndDLExpansionHistory
        ::= 30

OriginatorAndDLExpansionHistory ::= SEQUENCE SIZE (0. .ub-dl-expansions) OF OriginatorAndDLExpansion

OriginatorAndDLExpansion ::= SEQUENCE {
        originator-or-dl-name ORAddressAndOptionalDirectoryName,
        origination-or-expansion-time TIME }
```

FIGURE 2/X.411 (Part 26 of 41)

Abstract syntax definition of the MTS abstract service

```
reporting-DL-name EXTENSION
        ReportingDLName
        ::= 31

ReportingDLName ::= ORAddressAndOptionalDirectoryName

reporting-MTA-certificate EXTENSION
        ReportingMTACertificate
        CRITICAL FOR DELIVERY
        ::= 32

ReportingMTACertificate ::= Certificates

report-origin-authentication-check EXTENSION
        ReportOriginAuthenticationCheck
        CRITICAL FOR DELIVERY
        ::= 33

ReportOriginAuthenticationCheck ::= SIGNATURE SEQUENCE {
        algorithm-identifier ReportOriginAuthenticationAlgorithmIdentifier,
        content-identifier ContentIdentifier OPTIONAL,
        message-security-label MessageSecurityLabel OPTIONAL,
        per-recipient SEQUENCE SIZE (1..ub-recipients) OF PerRecipientReportFields }

ReportOriginAuthenticationAlgorithmIdentifier ::= AlgorithmIdentifier

PerRecipientReportFields ::= SEQUENCE {
        actual-recipient-name ActualRecipientName,
        originally-intended-recipient-name OriginallyIntendedRecipientName OPTIONAL,
        CHOICE {
                delivery [0] PerRecipientDeliveryReportFields,
                non-delivery [1] PerRecipientNonDeliveryReportFields } }

PerRecipientDeliveryReportFields ::= SEQUENCE {
        message-delivery-time MessageDeliveryTime,
        type-of-MTS-user TypeOfMTSUser,
        recipient-certificate [0] RecipientCertificate OPTIONAL,
        proof-of-delivery [1] ProofOfDelivery OPTIONAL }

PerRecipientNonDeliveryReportFields ::= SEQUENCE {
        non-delivery-reason-code NonDeliveryReasonCode,
        non-delivery-diagnostic-code NonDeliveryDiagnosticCode OPTIONAL }
```

FIGURE 2/X.411 (Part 27 of 41)

Abstract syntax definition of the MTS abstract service

```
originating-MTA-certificate EXTENSION
        OriginatingMTACertificate
        ::= 34

OriginatingMTACertificate ::= Certificates

proof-of-submission EXTENSION
        ProofOfSubmission
        ::= 35

ProofOfSubmission ::= SIGNATURE SEQUENCE{
        algorithm-identifier ProofOfSubmissionAlgorithmIdentifier,
        message-submission-envelope MessageSubmissionEnvelope,
        content Content,
        message-submission-identifier MessageSubmissionIdentifier,
        message-submission-time MessageSubmissionTime }

ProofOfSubmissionAlgorithmIdentifier ::= AlgorithmIdentifier
```

FIGURE 2/X.411 (Part 28 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Common parameter types

Content ::= OCTET STRING        -- when the content-type has the integer value external,
                                -- the value of the content octet string is the ASN.1
                                -- encoding of the external content an external-content
                                -- is a data type EXTERNAL

MTSIdentifier ::= [APPLICATION 4] SEQUENCE{
        global-domain-identifier GlobalDomainIdentifier,
        local-identifier LocalIdentifier }

LocalIdentifier ::= IA5String (SIZE (1. .ub-local-id-length))

GlobalDomainIdentifier ::= [APPLICATION 3] SEQUENCE{
        country-name CountryName,
        administration-domain-name AdministrationDomainName,
        private-domain-identifier PrivateDomainIdentifier OPTIONAL }

PrivateDomainIdentifier ::= CHOICE{
        numeric NumericString (SIZE (1. .ub-domain-name-length)),
        printable PrintableString (SIZE (1. .ub-domain-name-length)) }

MTAName ::= IA5String (SIZE (1. .ub-mta-name-length))

Time ::= UTCTime
```

FIGURE 2/X.411 (Part 29 of 41)

Abstract syntax definition of the MTS abstract service

```
-- O/R names

ORAddressAndOrDirectoryName ::= ORName

ORAddressAndOptionalDirectoryName ::= ORName

ORName ::= [APPLICATION 0] SEQUENCE {
        address COMPONENTS OF ORAddress,
        directory-name [0] Name OPTIONAL }

ORAddress ::= SEQUENCE {
        standard-attributes StandardAttributes,
        domain-defined-attributes DomainDefinedAttributes OPTIONAL,
                -- also see teletex-domain-defined-attributes
        extension-attributes ExtensionAttributes OPTIONAL }

-- Note – The OR-address is semantically absent from the OR-name if the standard-attribute sequence is empty
-- and the domain-defined-attributes and extension-attributes are both omitted.


-- Standard attributes

StandardAttributes ::= SEQUENCE
        country-name CountryName OPTIONAL,
        administration-domain-name AdministrationDomainName OPTIONAL,
        network-address [0] NetworkAddress OPTIONAL, -- also see extended-network-address
        terminal-identifier [1] TerminalIdentifier OPTIONAL,
        private-domain-name [2] PrivateDomainName OPTIONAL,
        organization-name [3] OrganizationName OPTIONAL, --also see teletex-organization-name
        numeric-user-identifier [4] NumericUserIdentifier OPTIONAL,
        personal-name [5] PersonalName OPTIONAL, -- also see teletex-personal-name
        organizational-unit-names [6] OrganizationalUnitNames OPTIONAL
                -- also see teletex-organizational-unit-names -- }

CountryName ::= [APPLICATION 1] CHOICE {
        x121-dcc-code NumericString (SIZE (ub-country-name-numeric-length)),
        iso-3166-alpha2-code PrintableString (SIZE (ub-country-name-alpha-length)) }

AdministrationDomainName ::= [APPLICATION 2] CHOICE {
        numeric NumericString (SIZE (0. .ub-domain-name-length)),
        printable PrintableString (SIZE (0. .ub-domain-name-length)) }

NetworkAddress ::= x121Address

X121Address ::= NumericString (SIZE (1. .ub-x121-address-length))

TerminalIdentifier ::= PrintableString (SIZE (1. .ub-terminal-id-length))
```

FIGURE 2/X.411 (Part 30 of 41)

Abstract syntax definition of the MTS abstract service

```
PrivateDomainName ::= CHOICE {
        numeric NumericString (SIZE (1. .ub-domain-name-length)),
        printable PrintableString (SIZE (1. .ub-domain-name-length)) }

OrganizationName ::= Printable String (SIZE (1. .ub-organization-name-length))

NumericUserIdentifier ::= NumericString (SIZE (1. .ub-numeric-user-id-length))

Personal Name ::= SET {
        surname [0] PrintableString (SIZE (1. .ub-surname-length)),
        given-name [1] PrintableString (SIZE (1. .ub-given-name-length)) OPTIONAL,
        initials [2] PrintableString (SIZE (1. .ub-initials-length)) OPTIONAL,
        generation-qualifier [3] PrintableString (SIZE (1. .ub-generation-qualifier-length)) OPTIONAL }

OrganizationUnitNames ::= SEQUENCE SIZE (1. .ub-organizational-units) OF OrganizationUnitName

OrganizationUnitName ::= PrintableString (SIZE (1. .ub-organizational-unit-name-length))


-- Domain-defined attributes

DomainDefinedAttributes ::= SEQUENCE SIZE (1. .ub-domain-defined-attributes) OF DomainDefinedAttribute

DomainDefinedAttribute ::= SEQUENCE {
        type PrintableString (SIZE (1. .ub-domain-attribute-type-length)),
        value PrintableString (SIZE (1. .ub-domain-defined-attribut-value-length)) }


-- Extension attributes

ExtensionAttributes ::= SET SIZE (1. .ub-extension-attributes) OF ExtensionAttribute

ExtensionAttribute ::= SEQUENCE {
        extension-attribute-type [0] EXTENSION-ATTRIBUTE,
        extension-attribute-value [1] ANY DEFINED BY extension-attribute-type }

EXTENSION-ATTRIBUTE MACRO ::=
BEGIN

TYPE NOTATION::= TYPE | empty
VALUE NOTATION ::= value (VALUE INTEGER (0. .ub-extension-attributes))

END  -- of EXTENTION-ATTRIBUTE
```

FIGURE 2/X.411 (Part 31 of 41)

Abstract syntax definition of the MTS abstract service

```
common-name EXTENSION-ATTRIBUTE
        CommonName
        ::= 1

CommonName ::= PrintableString (SIZE (1. . .ub-common-name-length))

teletex-common-name EXTENSION-ATTRIBUTE
        TeletexCommonName
        ::= 2

TeletexCommonName ::= TeletexString (SIZE (1. . .ub-common-name-length))

teletex-organization-name EXTENSION-ATTRIBUTE
        TeletexOrganizationalName
        ::= 3

TeletexOrganizationalName ::= TeletexString (SIZE (1. . .ub-organization-name-length))

teletex-personal-name EXTENSION-ATTRIBUTE
        TeletexPersonalName
        ::= 4

TeletexPersonalName ::= SET {
        surname [0] TeletexString (SIZE (1. . .ub-surname-length)),
        given-name [1] TeletexString (SIZE (1. . .ub-given-name-length)) OPTIONAL,
        initials [2] TeletexString (SIZE (1. . .ub-initials-length)) OPTIONAL,
        generation-qualifier [3] TeletexString (SIZE (1. . .ub-generation-qualifier-length)) OPTIONAL }

teletex-organizational-unit-names EXTENSION-ATTRIBUTE
        TeletexOrganizationUnitNames
        ::= 5

TeletexOrganizationUnitNames ::= SEQUENCE (SIZE (1. . .ub-organizational-units) OF
        TeletexOrganizationalUnitName

TeletexOrganizationalUnitName ::= TeletexString (SIZE (1. . .ub-organizational-unit-name-length))

teletex-domain-defined-attributes EXTENSION-ATTRIBUTE
        TeletexDomainDefinedAttributes
        ::= 6

TeletexDomainDefinedAttributes ::= SEQUENCE SIZE (1. . .ub-domain-defined-attributes) OF
        TeletexDomainDefinedAttribute
```

FIGURE 2/X.411 (Part 32 of 41)

Abstract syntax definition of the MTS abstract service

```
TeletexDomainDefinedAttribute ::= SEQUENCE {
        typeTeletexString (SIZE (1..ub-domain-defined-attribute-type-length)),
        value TeletexString (SIZE (1..ub-domain-defined-attribute-value-length)) }

pds-name EXTENSION-ATTRIBUTE
        PDSName
        ::= 7

PDSName ::= PrintableString (SIZE (1..ub-pds-name-length))

physical-delivery-country-name EXTENSION-ATTRIBUTE
        PhysicalDeliveryCountryName
        ::= 8

PhysicalDeliveryCountryName ::= CHOICE {
        x121-dcc-code NumericString (SIZE (ub-country-name-numeric-length)),
        iso-3166-alpha2-code PrintableString (SIZE (ub-country-name-alpha-length)) }

postal-code EXTENSION-ATTRIBUTE
        PostalCode
        ::= 9

PostalCode ::= CHOICE {
        numeric-code NumericString (SIZE (1..ub-postal-code-length)),
        printable-code PrintableString (SIZE (1..ub-postal-code-length)) }

physical-delivery-office-name EXTENSION-ATTRIBUTE
        PhysicalDeliveryOfficeName
        ::= 10

PhysicalDeliveryOfficeName ::= PDS Parameter

physical-delivery-office-number EXTENSION-ATTRIBUTE
        PhysicalDeliveryOfficeNumber
        ::= 11

PhysicalDeliveryOfficeNumber ::= PDS Parameter

extension-OR-address-components EXTENSION-ATTRIBUTE
        ExtensionORAddressComponents
        ::= 12
```

FIGURE 2/X.411 (Part 33 of 41)

Abstract syntax definition of the MTS abstract service

```
ExtensionORAddressComponents ::= PDS Parameter

physical-delivery-personal-name EXTENSION-ATTRIBUTE
        PhysicalDeliveryPersonalName
        ::= 13

PhysicalDeliveryPersonalName ::= PDS Parameter

physical-delivery-organization-name EXTENSION-ATTRIBUTE
        PhysicalDeliveryOrganizationName
        ::= 14

PhysicalDeliveryOrganizationName ::= PDS Parameter

extension-physical-delivery-address-components EXTENSION-ATTRIBUTE
        ExtensionPhysicalDeliveryAddressComponents
        ::= 15

ExtensionPhysicalDeliveryAddressComponents ::= PDS Parameter

unformatted-postal-address EXTENSION-ATTRIBUTE
        UnformattedPostalAddress
        ::= 16

UnformattedPostalAddress ::= SET{
        printable-address SEQUENCE SIZE (1. .ub-pds-physical-address-lines) OF
                  PrintableString (SIZE (1. .ub-pds-parameter-length)) OPTIONAL,
        teletex-string TeletexString (SIZE (1. .ub-unformatted-address-length)) OPTIONAL}

street-address EXTENSION-ATTRIBUTE
        StreetAddress
        ::= 17

StreetAddress ::= PDS Parameter
```

FIGURE 2/X.411 (Part 34 of 41)

Abstract syntax definition of the MTS abstract service

```
post-office-box-address EXTENSION-ATTRIBUTE
        PostOfficeBoxAddress
        ::= 18

PostOfficeBoxAddress ::= PDS Parameter

poste-restante-address EXTENSION-ATTRIBUTE
        PosteRestanteAddress
        ::= 19

PosteRestanteAddress ::= PDS Parameter

unique-postal-name EXTENSION-ATTRIBUTE
        UniquePostalName
        ::= 20

UniquePostalName ::= PDS Parameter

local-postal-attributes EXTENSION-ATTRIBUTE
        LocalPostalAttributes
        ::= 21

LocalPostalAttributes ::= PDS Parameter

PDS Parameter ::= SET
        printable-string PrintableString (SIZE (1. . .ub-pds-parameter-length)) OPTIONAL,
        teletex-string TeletexString (SIZE (1. . .ub-pds-parameter-length)) OPTIONAL }

extended-network-address EXTENSION-ATTRIBUTE
        ExtendedNetworkAddress
        ::= 22

ExtendedNetworkAddress ::= CHOICE {
        e163-4-address SEQUENCE {
                number [0] NumericString (SIZE (1. .ub-e163-4-number-length)),
                sub-address [1] NumericString (SIZE (1. .ub-e163-4-sub-address-length)) OPTIONAL },
        psap-address [0] PresentationAddress }

terminal-type EXTENSION-ATTRIBUTE
        TerminalType
        ::= 23
```

FIGURE 2/X.411 (Part 35 of 41)

Abstract syntax definition of the MTS abstract service

```
TerminalType ::= INTEGER {
        telex (3),
        teletex (4),
        g3-facsimile (5),
        g4-facsimile (6),
        ia5-terminal (7),
        videotex (8) } (0. .ub-integer-options)
```

FIGURE 2/X.411 (Part 36 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Encoded information types

EncodedInformationTypes ::= [APPLICATION 5] SET {
        built-in-encoded-information-types [0] BuiltInEncodedInformationTypes,
        non-basic-parameters COMPONENTS OF NonBasicParameters,
        external-encoded-information-types [4] ExternalEncodedInformationTypes OPTIONAL }


-- Built-in encoded information types

Built-inEncodedInformationTypes ::= BIT STRING {
        undefined (0),
        telex (1),
        ia5-text (2),
        g3-facsimile (3),
        g4-class-1 (4),
        teletex (5),
        videotex (6),
        voice (7),
        sfd (8),
        mixed-mode (9) } (SIZE (0..ub-built-in-encoded-information-types))


-- Non-basic parameters

NonBasicParameters ::= SET {
        g3-facsimile [1] G3FacsimileNonBasicParameters DEFAULT { },
        teletex [2] TeletexNonBasicParameters DEFAULT { },
        g4-class-1-and-mixed-mode [3] G4Class1AndMixedModeNonBasicParameters OPTIONAL }

G3FacsimileNonBasicParameters ::= BIT STRING {
        two-dimensional (8),
        fine-resolution (9),
        unlimited-length (20),
        b4-length (21),
        a3-width (22),
        b4-width (23),
        uncompressed (30) }                   -- as defined in Recommendation T.30

TeletexNonBasicParameters ::= SET {
        graphic-character-sets [0] TeletexString OPTIONAL,
        control-character-sets [1] TeletexString OPTIONAL,
        page-formats [2] OCTET STRING OPTIONAL,
        miscellaneous-terminal-capabilities [3] TeletexString OPTIONAL,
        private-use [4] OCTET STRING OPTIONAL     -- maximum ub-teletex-private-use-length octets -- }
        -- as defined in Recommendation T.62
```

FIGURE 2/X.411 (Part 37 of 41)

Abstract syntax definition of the MTS abstract service

```
G4Class1AndMixedModeNonBasicParameters ::= PresentationCapabilities

PresentationCapabilities ::= ANY       -- as defined in Recommendations T.400, T.503 and T.501


-- External encoded information types

ExternalEncodedInformationTypes ::= SET SIZE (1..ub-encoded-information-types) OF
        ExternalEncodedInformationType

ExternalEncodedInformationType ::= OBJECT IDENTIFIER
```

FIGURE 2/X.411 (Part 38 of 41)

Abstract syntax definition of the MTS abstract service

```
-- Token

Token ::= SEQUENCE {
        token-type-identifier [0] TOKEN,
        token [1] ANY DEFINED BY token-type-identifier }

TOKEN MACRO ::=
BEGIN

TYPE NOTATION ::= type | empty
VALUE NOTATION ::= (VALUE OBJECT IDENTIFIER)

END  -- of TOKEN

asymmetric-token TOKEN
                AsymmetricToken
        ::= id-tok-asymmetricToken

AsymmetricToken ::= SIGNED SEQUENCE {
        signature-algorithm-identifier AlgorithmIdentifier,
        recipient-name RecipientName,
        time Time,
        signed-data [0] TokenData OPTIONAL,
        encryption-algorithm-identifier [1] AlgorithmIdentifier OPTIONAL,
        encrypted-data [2] ENCRYPTED TokenData OPTIONAL }

TokenData ::= SEQUENCE {
        type [0] TOKEN-DATA,
        value [1] ANY DEFINED BY type }

TOKEN-DATA MACRO ::=
BEGIN

TYPE NOTATION ::= type | empty
VALUE NOTATION ::= value (VALUE INTEGER)

END  -- TOKEN-DATA

bind-token-signed-data TOKEN-DATA
                BindTokenSignedData
        ::=

BindTokenSignedData ::= RandomNumber
```

FIGURE 2/X.411 (Part 39 of 41)

Abstract syntax definition of the MTS abstract service

```
RandomNumber ::= BIT STRING

message-token-signed-data TOKEN-DATA
                MessageTokenSignedData
    ::= 2

MessageTokenSignedData ::= SEQUENCE {
    content-confidentiality-alogorithm-identifier [0] ContentConfidentialityAlgorithmIdentifier
                OPTIONAL,
    content-integrity-check [1] ContentIntegrityCheck OPTIONAL,
    message-security-label [2] MessageSecurityLabel OPTIONAL,
    proof-of-delivery-request [3] ProofOfDeliveryRequest OPTIONAL,
    message-sequence-number [4] INTEGER OPTIONAL }

message-token-encrypted-data TOKEN-DATA
                MessageTokenEncryptedData
    ::= 3

MessageTokenEncryptedData ::= SEQUENCE {
    content-confidentiality-key [0] EncryptionKey OPTIONAL,
    content-integrity-check [1] ContentIntegrityCheck OPTIONAL,
    message-security-label [2] MessageSecurityLabel OPTIONAL,
    content-integrity-key [3] EncryptionKey OPTIONAL,
    message-sequence-number [4] INTEGER OPTIONAL }

EncryptionKey ::= BIT STRING


-- Security label

Security label ::= SET {
    security-policy-identifier SecurityPolicyIdentifier OPTIONAL,
    security-classification SecurityClassification OPTIONAL,
    privacy-mark PrivacyMark OPTIONAL,
    security-categories SecurityCategories OPTIONAL }

SecurityPolicyIdentifier ::= OBJECT IDENTIFIER

SecurityClassification ::= INTEGER {
    unmarked (0),
    unclassified (1),
    restricted (2),
    confidential (3),
    secret (4),
    top-secret (5) } (0. . .ub-integer-options)
```

FIGURE 2/X.411 (Part 40 of 41)

Abstract syntax definition of the MTS Abstract Service

```
PrivacyMark ::= PrintableString (SIZE (1. .ub-privacy-mark-length))

SecurityCategories ::= SET SIZE (1. .ub-security-categories) OF SecurityCategory

SecurityCategory ::= SEQUENCE {
        type [0] SECURITY-CATEGORY,
        value [1] ANY DEFINED BY type }

SECURITY-CATEGORY MACRO ::=
BEGIN

TYPE NOTATION ::= type | empty
VALUE NOTATION ::= value (VALUE OBJECT IDENTIFIER)

END  -- of SECURITY-CATEGORY

END  -- of MTSAbstractService
```

FIGURE 2/X.411 (Part 41 of 41)

Abstract syntax definition of the MTS abstract service

SECTION 3 – MESSAGE TRANSFER AGENT ABSTRACT SERVICE

10 Refined message transfer system model

Paragraph 6 describes the MTS as an object, without reference to its internal structure. This paragraph refines the MTS model, and exposes its component objects and the ports shared between them.

Figure 3/X.411 models the MTS and reveals its internal structure.

The MTS comprises a collection of message-transfer-agent (MTA) objects, which cooperate together to form the MTS and offer the MTS abstract service to its users. It is the MTAs which perform the active functions of the MTS, i.e. transfer of messages, probes and reports, generation of reports, and content conversion.

MTA objects also have ports, some of which are precisely those which are also visible at the boundary of the MTS object, i.e. submission-ports, delivery-ports and administration ports. However, MTAs also have another type of port – which are concerned with the distribution of the MTS abstract service between the MTAs, and are not visible at the boundary of the MTS object.

A transfer-port enables an MTA to transfer messages, probes and reports to another MTA. In general, a message, probe or report may have to be transferred a number of times between different MTAs to reach its intended destination.

If a message is addressed to multiple recipients served by several different MTAs, the message must be transferred through the MTS along several different paths. From the perspective of an MTA transferring such a message, some recipients may be reached via one path while other recipients may be reached via another. At such an MTA, two copies of the message are created, and each is transferred to the next MTA along its respective path. The copying and branching of the message is repeated until each copy has reached a final destination MTA, where the message can be delivered to one or more recipient MTS-users.

Every MTA along a path taken by a message is responsible for delivering or transferring the message to a particular subset of the originally-specified-recipients. Other MTAs take care of the deliver or transfer to remaining recipients, using copies of the messages created along the way.

Reports on the delivery or non-delivery of a message to one or more recipient MTS-users, are generated by MTAs in accordance with the request of the originator of the message and the originating-MTA. An MTA may generate a delivery-report upon successfully delivering a copy of a message to a recipient MTS-user. It may generate a non-delivery-report upon determining that a copy of a message is undeliverable to one or more recipients, that is, it is unable to deliver the message to the recipient MTS-users, or it is unable to transfer the message to an adjacent MTA that would take responsibility for delivery or transferring the message further.

For efficiency, an MTA may generate a single, combined report that applies to several copies of a single, multiple recipient message for which it is responsible. Both delivery- and non-delivery-reports may be combined together. However, in order for reports to be combined in this manner, the same content conversion, if any, must have been performed on the message for all recipients to whom the report refers.

Reports that pertain to copies of the same multiple recipient message but that were generated by different MTAs are not combined by any intermediate MTAs, but instead remain distinct.

When required, an MTA may perform content conversion. When neither the originating nor the recipient MTS-user requests nor prohibits conversion, implicit conversion of a message's encoded-information-types may be performed by an MTA to suit the encoded-information-types that the recipient MTS-user is able to receive. The originating MTS-user may also explicitly request conversion of specific encoded-information-types for a particular recipient MTS-user.

The submission-, delivery- and administration-ports of an MTA, which are also visible at the boundary of the MTS, are defined in Section 2 of this Recommendation. The remaining paragraphs in this section define the transfer-port of an MTA, and the procedures performed by MTAs to ensure the correct distributed operation of the MTS.

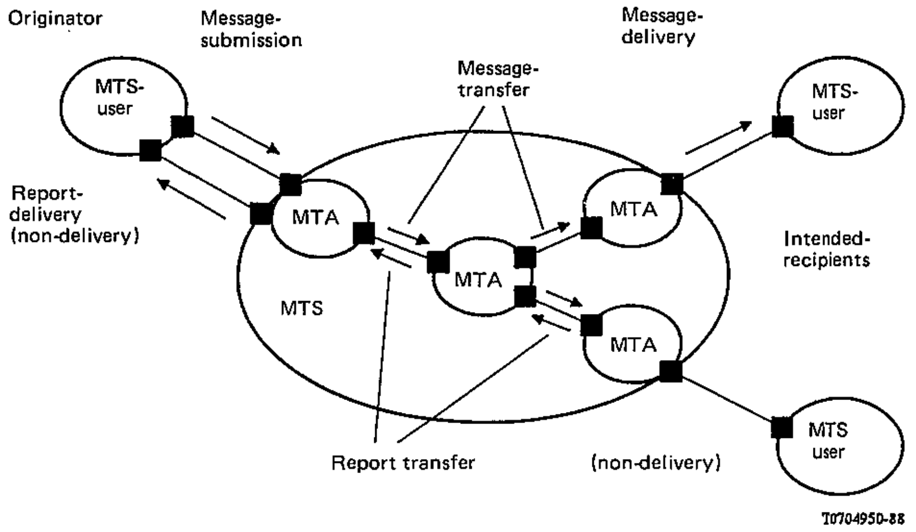


FIGURE 3/X.411

Refined message transfer system model

11 Message transfer agent abstract service overview

Section two defines the MTS abstract service provided by the submission-, delivery- and administration-ports of an MTA. This paragraph defines the following abstract-operations that are provided by the transfer-ports of MTAs:

MTA-bind and MTA-unbind

a) MTA-bind

b) MTA-unbind.

Transfer port abstract-operations

c) message-transfer

d) probe-transfer

e) report-transfer.

11.1 *MTA-bind and MTA-unbind*

The **MTA-bind** enables an MTA to establish an association with another MTA. Abstract-operations other than MTA-bind can only be invoked in the context of an established association.

The **MTA-unbind** enables an MTA to establish an association with another MTA. Abstract-operations other than MTA-bind can only be invoked in the context of an established association.

11.2 *Transfer port abstract-operations*

The **message-transfer** abstract-operation enables an MTA to transfer a message to another MTA.

The **probe-transfer** abstract-operation enables an MTA to transfer a probe to another MTA.

The **report-transfer** abstract-operation enables an MTA to transfer a report to another MTA.

12 Message transfer agent abstract service definition

The MTS abstract service is defined in § 8. This paragraph defines the semantics of the parameters of the abract-service provided by the transfer-port of MTAs.

Paragraph 12.1 defines the MTA-bind and MTA-unbind. Paragraph 12.2 defines the transfer-port. Paragraph 12.3 defines some common parameter types.

The abstract-syntax of the MTA abstract service is defined in § 13.

12.1 *MTA-bind and MTA-unbind*

This paragraph defines the abstract-service used to establish and release associations between MTAs.

12.1.1 *Abstract-bind and abstract-unbind*

This paragraph defines the following abstract-bind and abstract-unbind:

a) MTA-bind

b) MTA-unbind.

12.1.1.1 *MTA-bind*

The MTA-bind enables an MTA to establish an association with another MTA.

The MTA-bind establishes the **credentials** of MTAs to interact, and the **application-context** and **security-context** of the association. An assocition can only be released by the initiator of that association (using MTA-unbind).

Abstract-operations other than MTA-bind can only be invoked in the context of an established association.

The successful completion of the MTA-bind signifies the establishment of an association.

The disruption of the MTA-bind by a bind-error indicates that an association has not been established.

12.1.1.1.1 *Arguments*

Table 27/X.411 lists the arguments of the MTA-bind, and for each argument qualifies its presence and indicates the paragraph in which the argument is defined.

TABLE 27/X.411

MTA-bind arguments

Argument	Presence	Clause
Bind arguments		
Initiator-name	O	12.1.1.1.1.1
Initiator-credentials	O	12.1.1.1.1.2
Security-context	O	12.1.1.1.1.3

12.1.1.1.1.1 *Initiator-name*

This argument contains a name for the initiator of the association. It may be generated by the initiator of the association.

The name of an **MTA-name**.

12.1.1.1.1.2 *Initiator-credentials*

This argument contains the **credentials** of the initiator of the association. It may be generated by the initiator of the association.

The **initiator-credentials** may be used by the responder to authenticate the identity of the initiator (see Recommendation X.509).

If only simple-authentication is proposed, the **initiator-credentials** comprise a simple **password** associated with the **initiator-name**.

If strong-authentication is used, the **initiator-credentials** comprise an **initiator-bind-token** and, optionally, an **initiator-certificate**.

The **initiator-bind-token** is a **token** generated by the initiator of the association. If the **initiator-bind-token** is an **asymmetric-token**, the **signed-data** comprises a **random-number**. The **encrypted-data** of an **asymmetric-token** may be used to convey secret security-relevant information (e.g., one or more symmetric-encryption-keys) used to secure the association, or may be absent from the **initiator-bind-token**.

The **initiator-certificate** is a **certificate** of the initiator of the association, generated by a trusted source (e.g., a certification-authority). It may be supplied by the initiator of the association, if the **initiator-bind-token** is an **asymmetric-token**. The **initiator-certificate** may be used to convey a verified copy of the public-asymmetric-encryption-key (**subject-key**) of the initiator of the association. The initiator's public-asymmetric-encryption-key may be used by the responder to compute the **responder-bind-token**. If the responder is known to have, or have access to, the initiator's certificate (e.g., via the Directory), the **initiator-certificate** may be omitted.

12.1.1.1.1.3 *Security-context*

This argument indicates the **security-context** that the initiator of the association proposes to operate at. It may be generated by the initiator of the association.

The **security-context** comprises one or more **security-labels** that defines the sensitivity of interactions that may occur between the MTAs for the duration of the association, in line with the security-policy in force. The **security-context** shall be one that is allowed by the **security-labels** associated with the MDs (MTAs).

If **security-contexts** are not established between the MTAs, the sensitivity of interactions that may occur between the MTAs may be at the discretion of the invoker of an abstract-operation.

12.1.1.1.2 *Results*

Table 28/X.411 lists the results of the MTA-bind, and for each result qualifies its presence and indicates the paragraph in which the result is defined.

TABLE 28/X.411

MTA-bind results

Result	Presence	Clause
Bind results		
Responder-name	O	12.1.1.1.2.1
Responder-credentials	O	12.1.1.1.2.2

12.1.1.1.2.1 *Responder-name*

This argument contains a name for the responder of the association. It may be generated by the responder of the association.

The name is an **MTA-name**.

12.1.1.1.2.2 *Responder-credentials*

This argument contains the **credentials** of the responder of the association. It may be generated by the responder of the association.

The **responder-credentials** may be used by the initiator to authenticate the identity of the responder (see Recommendation X.509).

If only simple-authentication is used, the **responder-credentials** comprise a simple **password** associated with the **responder-name**.

If strong-authentication is used, the **responder-credentials** comprise a **responder-bind-token**. The **responder-bind-token** is a **token** generated by the responder of the association. The **responder-bind-token** shall be the same type of **token** as the **initiator-bind-token**. If the **responder-bind-token** is an **asymmetric-token**, the **signed-data** comprises a **random-number** (which may be related to the **random-number** supplied in the **initiator-bind-token**). The **encrypted-data** of an **asymmetric-token** may be used to convey security-relevant information (e.g., one or more symmetric-encryption-keys) used to secure the association, or may be absent from the **responder-bind-token**.

12.1.1.1.3 *Bind-errors*

The bind-errors that may disrupt the MTA-bind are defined in § 12.1.2.

12.1.1.2 *MTA-unbind*

The MTA-unbind enables the release of an established association by the initiator of the association.

12.1.1.2.1 *Arguments*

The MTA-unbind service has no arguments.

12.1.1.2.2 *Results*

The MTA-unbind service returns an empty result as indication of release of the association.

12.1.1.2.3 *Unbind-errors*

There are no unbind-errors that may disrupt the MTA-unbind.

12.1.2 *Bind-errors*

This paragraph defines the following bind-errors:

a) authentication-error,

b) busy,

c) unacceptable-dialogue-mode,

d) unacceptable-security-context.

12.1.2.1 *Authentication-error*

The authentication-error bind-error reports that an association cannot be established due to an authentication error; the initiator's **credentials** are not acceptable or are improperly specified.

The authentication-error bind-error has no parameters.

12.1.2.2 *Busy*

The busy bind-error reports that an association cannot be established because the responder is busy.

The busy bind-error has no parameters.

12.1.2.3 *Unacceptable-dialogue-mode*

The unacceptable-dialogue-mode bind-error reports that the dialogue-mode proposed by the initiator of the association is unacceptable to the responder (see § 12 of Recommendation X.419).

The unacceptable-dialogue-mode bind-error has no parameters.

12.1.2.4 *Unacceptable-security-context*

The unacceptable-security-context-bind-error reports that the **security-context** proposed by the initiator of the association is unacceptable to the responder.

The Unacceptable-security-context bind-error has no parameters.

12.2 *Transfer port*

This paragraph defines the abstract-operations and abstract-errors which occur at a transfer-port.

12.2.1 *Abstract-operations*

This paragraph defines the following transfer-port abstract-operations:

a) message-transfer,

b) probe-transfer,

c) report-transfer.

12.2.1.1 *Message-transfer*

The message-transfer abstract-operation enables the MTA to transfer a message to another MTA.

12.2.1.1.1 *Arguments*

Table 29/X.411 lists the arguments of the message-transfer abstract-operation, and for each argument qualifies its presence and identifies the paragraph in which the argument is defined.

12.2.1.1.1.1 *Message-identifier*

This argument contains an **MTS-identifier** that distinguishes the message from all other messages, probes and reports within the MTS. It shall be generated by the originating-MTA of the message, and shall have the same value as the **message-submission-identifier** supplied to the originator of the message when the message was submitted, and the **message-delivery-identifier** supplied to the recipient of the message when the message is delivered.

When a message is copied for routing to multiple recipients via different MTAs, each copy of the message bears the **message-identifier** of the original. The copies can be distinguished from one another by the **originally-specified-recipient-number** and the corresponding **responsibility** arguments, which specify to which recipient(s) each copy is to be delivered.

12.2.1.1.1.2 *Per-domain-bilateral-information*

This argument contains information intended for MDs which the message will encounter as it is transferred through the MTS. It may be generated by the originating-MD of the message.

This argument may contain zero or more elements, each of which comprises:

– the **bilateral-information** intended for an MD;

– the **country-name**, the **administration-domain-name** and, optionally, the **private-domain-identifier** of the MD for which the **bilateral-information** is intended.

12.2.1.1.1.3 *Trace-information*

This argument documents the actions taken on the message (or probe or report) by each MD through which the message (or probe or report) passes as it is transferred through the MTS (see § 12.3.1). It shall be generated by each MD through which the message (or probe or report) passes.

12.2.1.1.1.4 *Internal-trace-information*

This argument documents the actions taken on the message (or probe or report) by each MTA through which the message (or probe or report) passes as it is transferred within an MD (see § 12.3.1). It shall be generated by each MTA through which the message (or probe or report) passes within an MD.

This argument shall not be supplied by the invoker of the message-transfer abstract-operation when transferring a message to another MD, unless by bilateral agreement between MDs.

12.2.1.1.1.5 *Originally-specified-recipient-number*

This argument, combined with the **message-identifier**, unambiguously identifies the copy of the message delivered to each recipient. It shall be generated by the originating-MTA of the message. A different value of this argument is specified for each recipient of the message.

The **originally-specified-recipient-number** is an integer value in the range that begins with one and ends with the number of originally-specified-recipients.

TABLE 29/X.411

Message-transfer arguments

Argument	Presence	Clause
Relaying arguments		
Message-identifier	M	12.2.1.1.1.1
Per-domain-bilateral-information	C	12.2.1.1.1.2
Trace-information	M	12.2.1.1.1.3
Internal-trace-information	C	12.2.1.1.1.4
DL-expansion-history	C	8.3.1.1.1.7
Originator argument		
Originator-name	M	8.2.1.1.1.1
Recipient arguments		
Recipient-name	M	8.2.1.1.1.2
Originally-specified-recipient-number	M	12.2.1.1.1.5
Responsibility	M	12.2.1.1.1.6
DL-expansion-prohibited	C	8.2.1.1.1.6
Disclosure-of-recipients	C	8.2.1.1.1.7
Redirection arguments		
Alternate-recipient-allowed	C	8.2.1.1.1.3
Recipient-reassignment-prohibited	C	8.2.1.1.1.4
Originator-requested-alternate-recipient	C	8.2.1.1.1.5
Intended-recipient-name	C	8.3.1.1.1.4
Redirection-reason	C	8.3.1.1.1.5
Priority argument		
Priority	C	8.2.1.1.1.8
Conversion arguments		
Implicit-conversion-prohibited	C	8.2.1.1.1.9
Conversion-with-loss-prohibited	C	8.2.1.1.1.10
Explicit-conversion	C	8.2.1.1.1.11
Delivery time arguments		
Deferred-delivery-time	C	12.2.1.1.1.7
Latest-delivery-time	C	8.2.1.1.1.13
Delivery method argument		
Requested-delivery-method	C	8.2.1.1.1.14
Physical delivery arguments		
Physical-forwarding-prohibited	C	8.2.1.1.1.15
Physical-forwarding-address-request	C	8.2.1.1.1.16
Physical-delivery-modes	C	8.2.1.1.1.17
Registred-mail-type	C	8.2.1.1.1.18
Recipient-number-for-advice	C	8.2.1.1.1.19
Physical-rendition-attributes	C	8.2.1.1.1.20
Originator-return-address	C	8.2.1.1.1.21
Delivery report request arguments		
Originator-report-request	M	8.2.1.1.1.22
Originating-MTA-report-request	M	12.2.1.1.1.8
Content-return-request	C	8.2.1.1.1.23
Physical-delivery-report-request	C	8.2.1.1.1.24
Security arguments		
Originator-certificate	C	8.2.1.1.1.25
Message-token	C	8.2.1.1.1.26
Content-confidentiality-algorithm-identifier	C	8.2.1.1.1.27
Content-integrity-check	C	8.2.1.1.1.28
Message-origin-authentication-check	C	8.2.1.1.1.29
Message-security-label	C	8.2.1.1.1.30
Proof-of-delivery-request	C	8.2.1.1.1.32
Content arguments		
Original-encoded-information-types	C	8.2.1.1.1.33
Content-type	M	8.2.1.1.1.34
Content-identifier	C	8.2.1.1.1.35
Content-correlator	C	8.2.1.1.1.36
Content	M	8.2.1.1.1.37

There is a one-to-one relationship between a particular **originally-specified-recipient-number** value and a particular **recipient-name** at the time of message-submission; it should not be assumed that this is a singular relationship at the time of message-delivery. That is, an **originally-specified-recipient-number** value can be used to distinguish an originally specified **recipient-name**, but not an actual recipient that will receive the message.

12.2.1.1.1.6 *Responsibility*

This argument indicates whether the receiving-MTA shall have the responsibility to either deliver the message to a recipient or to transfer it to another MTA for subsequent delivery to the recipient. It shall be generated by the sending-MTA. A different value of this argument may be specified for each recipient of the message.

This argument may have one of the following values: **responsible** or **not-responsible**.

12.2.1.1.1.7 *Deferred-delivery-time*

This argument is defined in § 8.2.1.1.1.12. It may appear in a message at a transfer-port if there is a bilateral agreement that an MTA other than the originating-MTA of the message will defer the delivery of the message.

12.2.1.1.1.8 *Originating-MTA-report-request*

This argument indicates the kind of report requested by the originating-MTA. It shall be generated by the originating-MTA of the message. A different value of this argument may be specified for each recipient of the message.

This argument may have one of the following values:

– **non-delivery-report**: a report is returned only in case of non-delivery, and it contains only the **last-trace-information**;

– **report**: a report is returned in case of delivery or non-delivery, and it contains only the **last-trace-information**;

– **audited-report**: a report is returned in case of delivery or non-delivery, and it contains all of the **trace-information**.

The **originating-MTA-report-request** argument shall specify at least the report level specified in the **originator-report-request** argument, where the increasing order or report levels is **no-report, non-delivery-report, report, audited-report**.

12.2.1.1.2 *Results*

The message-transfer abstract-operation does not return a result.

12.2.1.1.3 *Abstract-errors*

There are no abstract-errors that may disrupt the message-transfer abstract-operation.

12.2.1.2 *Probe-transfer*

The probe-transfer abstract-operation enables an MTA to transfer a probe to another MTA.

12.2.1.2.1 *Arguments*

Table 30/X.411 lists the arguments of the probe-transfer abstract-operation, and for each argument qualifies its presence and identifies the paragraph in which the argument is defined.

12.2.1.2.1.1 *Probe-identifier*

This argument contains an **MTS-identifier** that distinguishes the probe from all other message, probes and reports within the MTS. It shall be generated by the originating-MTA of the probe, and shall have the same value as the **probe-submission-identifier** supplied to the originator of the probe when the probe was submitted.

12.2.1.2.2 *Results*

The probe-transfer abstract-operation does not return a result.

12.2.1.2.3 *Abstract-errors*

There are no abstract-errors that may disrupt the probe-transfer abstract-operation.

TABLE 30/X.411

Probe-transfer arguments

Argument	Presence	Clause
Relaying arguments		
Probe-identifier	M	12.2.1.2.1.1
Per-domain-bilateral-information	C	12.2.1.1.1.2
Trace-information	M	12.2.1.1.1.3
Internal-trace-information	C	12.2.1.1.1.4
DL-expansion-history	C	8.3.1.1.1.7
Originator argument		
Originator-name	M	8.2.1.1.1.1
Recipient arguments		
Recipient-name	M	8.2.1.1.1.2
Originally-specified-recipient-number	M	12.2.1.1.1.5
Responsibility	M	12.2.1.1.1.6
DL-expansion-prohibited	C	8.2.1.1.1.6
Redirection arguments		
Alternate-recipient-allowed	C	8.2.1.1.1.3
Recipient-reassignment-prohibited	C	8.2.1.1.1.4
Originator-requested-alternate-recipient	C	8.2.1.1.1.5
Intended-recipient-name	C	8.3.1.1.1.4
Redirection-reason	C	8.3.1.1.1.5
Conversion arguments		
Implicit-conversion-prohibited	C	8.2.1.1.1.9
Conversion-with-loss-prohibited	C	8.2.1.1.1.10
Explicite-conversion	C	8.2.1.1.1.11
Delivery method argument		
Request-delivery-method	C	8.2.1.1.1.14
Physical delivery argument		
Physical-rendition-attributes	C	8.2.1.1.1.20
Report request arguments		
Originator-report-request	M	8.2.1.1.1.22
Originating-MTA-report-request	M	12.2.1.1.1.8
Security arguments		
Originator-certificate	C	8.2.1.1.1.25
Probe-origin-authentication-check	C	8.2.1.2.1.1
Message-security-label	C	8.2.1.1.1.30
Content arguments		
Original-encoded-information-types	C	8.2.1.1.1.33
Content-type	M	8.2.1.1.1.34
Content-identifier	C	8.2.1.1.1.35
Content-correlator	C	8.2.1.1.1.36
Content-length	C	8.2.1.2.1.2

12.2.1.3 *Report-transfer*

The report-transfer abstract-operation enables an MTA to transfer a report to another MTA.

12.2.1.3.1 *Arguments*

Table 31/X.411 lists the arguments of the report-transfer abstract-operation, and for each argument qualifies its presence and identifies the paragraph in which the argument is defined.

12.2.1.3.1.1 *Report-identifier*

This argument contains an **MTS-identifier** that distinguishes the report from all other messages, probes and reports within the MTS. It shall be generated by the originating-MTA of the report.

12.2.1.3.1.2 *Report-destination-name*

This argument contains the **OR-name** of the immediate destination of the report. It shall be generated by the originating-MTA of the report, and subsequently modified by the DL expansion-points if any DLs had been expanded to add recipients to the subject.

The originating-MTA of the report shall set this argument to be the **originator-name** of the subject if the subject does not have a **DL-expansion-history**, or to the last **OR-name** in the **DL-expasion-history** if this is present in the subject.

A DL expansion-point may replace its own **OR-name** in this argument by the **OR-name** which immediately preceeds its own **OR-name** in the report's **originator-and-DL-expansion-history**, or some other **OR-name** according to the reporting-policy of the DL.

12.2.1.3.1.3 *Subject-identifier*

This argument contains the **message-identifier** (or **probe-identifier**) of the subject (an **MTS-identifier**). It shall be generated by the originating-MTA of the subject.

12.2.1.3.1.4 *Subject-intermediate-trace-information*

The argument contains the **trace-information** present in the subject when it was transferred into the reporting-MD. It shall be present if, and only if, an audit-and-confirmed report was requested by the originating-MTA of the subject. It may be generated by the reporting-MTA.

Note – The inclusion in the **subject-intermediate-trace-information** of the **internal-trace-information** present in the subject when it was transferred to the reporting-MTA is for further study.

12.2.1.3.1.5 *Arrival-time*

This argument contains the **time** at which the subject entered the MD making the report. It shall be generated by the originating-MD of the report. A different value of this argument may be specified for each recipient of the subject to which the report relates.

12.2.1.3.1.6 *Additional-information*

The specification of the contents of this argument is by bilateral agreement between MDs.

12.2.1.3.2 *Results*

The report-transfer abstract-operation does not return a result.

12.2.1.3.3 *Abstract-errors*

There are no abstract-errors that may disrupt the report-transfer abstract-operation.

12.2.2 *Abstract-errors*

The transfer-port has not abstract-errors.

TABLE 31/X.411

Report-transfer arguments

Argument	Presence	Clause
Relaying arguments		
Report-identifier	M	12.2.1.3.1.1
Trace-information	M	12.2.1.1.1.3
Internal-trace-information	C	12.2.1.1.1.4
Report destination argument		
Report-destination-name	M	12.2.1.3.1.2
Report request argument		
Originator-report-request	M	8.2.1.1.1.22
Subject trace arguments		
Subject-identifier	M	12.2.1.3.1.3
Originally-specified-recipient-number	M	12.2.1.1.1.5
Subject-intermediate-trace-information	C	12.2.1.3.1.4
Arrival-time	M	12.2.1.3.1.5
Originator-and-DL-expansion-history	C	8.3.1.2.1.3
Reporting-DL-name	C	8.3.1.2.1.4
Conversion argument		
Converted-encoded-information types	C	8.3.1.2.1.5
Supplementary information arguments		
Supplementary-information	C	8.3.1.2.1.6
Physical-forwarding-address	C	8.3.1.2.1.7
Subject redirection arguments		
Actual-recipient-name	M	8.3.1.2.1.2
Intended-recipient-name	C	8.3.1.1.1.4
Redirection-reason	C	8.3.1.1.1.5
Content arguments		
Original-encoded-information-types	C	8.2.1.1.1.33
Content-type	C	8.2.1.1.1.34
Content-identifier	C	8.2.1.1.1.35
Content-correlator	C	8.2.1.1.1.36
Returned-content	C	8.3.1.2.1.14
Delivery arguments		
Message-delivery-time	C	8.2.1.2.1.8
Type-of-MTS-user	C	8.3.1.2.1.9
Non-delivery arguments		
Non-delivery-reason-code	C	8.3.1.2.1.10
Non-delivery-diagnostic-code	C	8.3.1.2.1.11
Security arguments		
Recipient-certificate	C	8.3.1.1.2.1
Proof-of-delivery	C	8.3.1.1.2.2
Reporting-MTA-certificate	C	8.3.1.2.1.12
Report-origin-authentication-check	C	8.3.1.2.1.13
Message-security-label	C	8.2.1.1.1.30
Additional information argument		
Additional-information	C	12.2.1.3.1.6

12.3 *Common parameter types*

This paragraph defines a number of common parameter types of the MTA abstract service.

12.3.1 *Trace-information and internal-trace-information*

Trace-information documents the actions taken on a message, probe or report by each MD through which it passes as it is transferred through the MTS.

Internal-trace-information documents the action taken on a message, probe or report by each TMA through which it passes as it is transferred through an MD. **Internal-trace-information** shall be removed from a message, probe or report before it is transferred out of an MD, unless by bilateral agreement between MDs.

Trace-information (or **internal-trace-information**) comprises a sequence of **trace-information-elements** (or **internal-trace-information-elements**). The first **trace-information-element** (or **internal-trace-information-element**) is that supplied by the originating-MD (or -MTA) of the message, probe or report. The second **trace-information-element** (or **internal-trace-information-element**) is that supplied by the next MD (or MTA) encountered by the message, probe or report, and so on. Each MD (or MTA) adds its **trace-information-element** (or **internal-trace-information-element**) to the end of the existing sequence. **Trace-information** is added by the first MTA encountered by the message, probe or report in each MD it passes through.

Each **trace-information-element** includes the **global-domain-identifier** of the MD supplying the **trace-information-element**.

Each **internal-trace-information-element** includes the **MTA-name** of the MTA supplying the **internal-trace-information-element** and the **global-domain-identifier** of the MD to which the MTA belongs.

Each **trace-information-element** (or **in ternal-trace-information-element**) includes the **arrival-time** at which the message, probe or report entered the MD (or MTA). In the case of the originating-MD (or -MTA) of the message, probe or report, the **arrival-time** is the time of message-submission, probe-submission or report generation, respectively.

Each **trace-information-element** (or **internal-trace-information-element**) specifies the **routing-action** the MD (or MTA) supplying the **trace-information-element** (or **internal-trace-information-element**) took with respect to the message, probe or report. **Relayed** is the normal **routing-action** of transferring the message, probe or report to another MD (or MTA). **Rerouted** indicates that an attempt had previously been made to route the message, probe or report to an **attempted-domain** (or **attempted-MTA**); the **global-domain-identifier** of the **attempted-domain** is included in the **trace-information-element**; if the rerouting attempt was to another MTA within the same MD, then the MTA-name of the **attempted-MTA** is included in the **internal-trace-information-element**; if the rerouting attempt was to another MD, then the **global-domain-identifier** of the **attempted-domain** is included in the **internal-trace-information-element** instead of an **MTA-name**.

Each **trace-information-element** (or **internal-trace-information-element**) also specifies any **additional-actions** the MD (or MTA) supplying the **trace-information-element** (or **internal-trace-information-element**) took with respect to the message, probe or report. Indications of any such **additional-actions** which appear in the **internal-trace-information-elements** during a traversal of an MD shall also be reflected in the corresponding **trace-information-element(s)** for the traversal of the MD.

If the deferred-delivery caused the MD (or MTA) supplying the **trace-information-element** (or **internal-trace-information-element**) to hold the message for a period of time, the **deferred-time** when it started to process the message for delivery or transfer is also included in the **trace-information-element** (or **internal-trace-information-element**). This parameter is not present in **trace-information-elements** (or **internal-trace-information-elements**) on probes and reports.

If the MD (or MTA) supplying the **trace-information-element** (or **internal-trace-information-element**) subjects a message to conversion, the **converted-encoded-information-types** resulting from the conversion is also included in the **trace-information-element** (or **internal-trace-information-element**). For a probe, an MD that would have converted the subject-message indicates the **encoded-information-types** the subject-message would contain after conversion in its **trace-information-element** (or **internal-trace-information-element**). This parameter is not present in **trace-information** (or **internal-trace-information-element**) on reports.

If the MD (or MTA) redirects a message or a probe (for any, but not necessarily all, of a message's or probe's recipients), **redirected** is indicated in the **trace-information-element** (or **internal-trace-information-element**). This parameter is not present in **trace-information** (or **internal-trace-information**) on reports.

If the MD (or MTA) expands a DL of a message or a probe, **dl-operation** is indicated in **trace-information-element** (or **internal-trace-information-element**). If the MD (or MTA) is a DL expasion-point and replaces its own **OR-name** in the **report-destination-name** of a report with another **OR-name** (see § 12.2.1.3.1.2), **dl-operation** is indicated in the **trace-information-element** (or **internal-trace-information-element**) of the report.

Loop detection and suppression is done by an MD (or MTA) when it receives a message, probe or report from another MD (or MTA). Messages, probes and reports may legitimately re-enter an MD (or MTA) for several reasons (**rerouted**, etc) and consequently a message, probe or report may have several disjoint **trace-information-elements** (or **internal-trace-information-elements**) from the same MD (or MTA). Each time a message, probe or report is transferred through an MD (or MTA) the generation of **trace-information-elements** (or **internal-trace- information-elements**) is performed as follows:

i) one trace-information-element (or internal-trace-information-element) is added, marked as relayed;

ii) if a rerouting attempt is to occur, then the trace-information-element (or internal-trace-information-element) added in i) is modified to rerouted (and the number of trace-information-element (or internal-trace-information-elements) added by the MD (or MTA) for this traversal of the MD (or MTA) remains at one);

iii) if subsequent attempts to reroute occur, then a new trace-information-element (or internal-trace-information-element) is added (marked as rerouted) to reflect each new rerouting attempt.

Several rerouting attempts to the same MD (or MTA) may occur.

Each **trace-information-element** (or **internal-trace-information-element**) added by an MD (or MTA) may contain indications of **additional-actions** performed by the MD (or MTA) on the message or probe (i.e., **deferred- time** (not present in **trace-information** (or **internal-trace-information**) on probes), **converted-encoded- information-types**, **redirected** or **dl-operation**).

13 Message transfer agent abstract syntax definition

The abstract-syntax of the MTA abstract service is defined in Figure 4/X.411.

The abstract-syntax of the MTA abstract service is defined using the abstract syntax notation (ASN.1) defined in Recommendation X.208, and the absract service definition conventions defined in Recommendation X.407.

The abstract-syntax definition of the MTA abstract service has the following major parts:

– *Prologue*: declaration of the exports from, and imports to, the MTA abstract service module (Figure 4/X.411, Part 1).

– *MTS refinement, objects and ports*: refinement of the MTS object, and definitions of the MTA object and the transfer-port (Figure 4/X.411, Part 2).

– *MTA-bind and MTA-unbind*: definitions of the MTA-bind and MTA-unbind used to establish and release associations between MTAs (Figure 4/X.411, Part 3).

– *Transfer ports*: definitions of the transfer-port abstract-operations: message-transfer, probe-transfer and report-transfer (Figure 4/X.411, Part 4).

– *Message transfer envelope*: definition of the message-transfer-envelope (Figure 4/X.411, Parts 5 and 6).

– *Probe transfer envelope*: definition of the probe-transfer-envelope (Figure 4/X.411, Part 7).

– *Report transfer envelope and content*: definitions of the report-transfer-envelope and report-transfer-content (Figure 4/X.411, Part 8).

– *Envelope and report content fields*: definitions of envelope and report content fields (Figure 4/X.411, Parts 9 and 10).

– *Extension fields*: definitions of extension-fields (Figure 4/X.411, Parts 11 and 12).

– *Common parameters types*: definitions of common parameter types (Figure 4/X.411, Part 13).

Note – The module implies a number of changes to the P1 protocol defined in Recommendation X.411 (1984). These changes are highlighted by means of underlining.

Each **extension-field** defined in Figure 4/X.411 (Parts 12 and 13) carries with it an indication of its **criticality** for submission, transfer and delivery. The criticality mechanism is described in § 9.1, and the procedures related to **extension-fields** and their **criticality** indications are further defined in § 14.

```
MTAAbstractService { joint-iso-ccitt mhs-motis(6) mts(3) modules(0) mta-abstract-service(2) }
DEFINITIONS IMPLICIT TAGS ::=
BEGIN


-- Prologue
-- Exports everything


IMPORTS
        -- Abstract service macros
        REFINE, OBJECT, PORT, ABSTRACT-BIND, ABSTRACT-UNBIND, ABSTRACT-OPERATION
                  FROM AbstractServiceNotation { joint-iso-ccitt mhs-motis(6) asdc(2) modules(0)
                      notation(1) }
        -- MTS abstract service parameters
        mTS, submission, delivery, administration, InitiatorCredentials, SecurityContext,
        ResponderCredentials, OriginalEncodedInformationTypes, ContentTypes, ContentIdentifier,
        Priority, PerMessageIndicators, DeferredDeliveryTime, CountryName, AdministrationDomainName,
        PrivateDomainIdentifier, ExplicitConversion, ContentLength, ConvertedEncodedInformationTypes,
        ReportType, SupplementaryInformation, EXTENSION, EXTENSIONS, recipient-reassignment-prohibited,
        dl-expansion-prohibited, conversion-with-loss-prohibited, latest-delivery-time,
        requested-delivery-method, physical-forwarding-prohibited, physical-forwarding-address-request,
        physical-delivery-modes, registered-mail-type, recipient-number-for-advice, physical-rendition-attributes,
        originator-return-address, physical-delivery-report-request, originator-certificate, message-token,
        content-confidentiality-algorithm-identifier, content-integrity-check, message-origin-authentication-check,
        message-security-label, proof-of-delivery-request, content-correlator, probe-origin-authentication-check,
        redirection-history, dl-expansion-history, originator-and-dl-expansion-history, reporting-dl-name,
        physical-forwarding-address, recipient-certificate, proof-of-delivery, reporting-MTA-certificate,
        report-origin-authentication-check, Content, MTSIdentifier, GlobalDomainIdentifier, MTAName, Time,
        ORAddressAndOptionalDirectoryName
                  FROM MTSAbstractService { joint-iso-ccitt mhs-motis(6) mts(3) modules(0)
                      mts-abstract-service(1) }
        -- Object identifiers
        id-ot-mta, id-pt-transfer
                  FROM MTSObjectIdentifiers { joint-iso-ccitt mhs-motis(6) mts(3) modules(0)
                      object-identifiers(0) }
        -- Upper bounds
        ub-bit-options, ub-dl-expansions, ub-integer-options, ub-recipients, ub-redirections, ub-transfers
                  FROM MTSUpperBounds { joint-iso-ccitt mhs-motis(6) mts(3) modules(0)
                      upper-bounds(3) };
```

FIGURE 4/X.411 (Part 1 of 13)

Abstract syntax definition of the MTA abstract service

```
-- MTS refinement

MTSRefinement ::= REFINE mTS AS
        mTA RECURRING
                submission      [S] VISIBLE
                delivery        [S] VISIBLE
                administration  [S] VISIBLE
                transfer            PAIRED WITH mTA


-- Objects

mTA OBJECT
        PORTS { submission [S], delivery [S], administration [S], transfer }
        ::= id-ot-mta


-- Ports

transfer PORT
        ABSTRACT OPERATIONS { MessageTransfer, ProbeTransfer, ReportTransfer }
        ::= id-pt-transfer
```

FIGURE 4/X.411 (Part 2 of 13)

Abstract syntax definition of the MTA abstract service

```
-- MTA-bind and MTA-unbind

MTABind ::= ABSTRACT-BIND
        TO { transfer }
        BIND
        ARGUMENT CHOICE {
                NULL,           -- if no authentication is required
                [1] SET {       -- if authentication is required
                    initiator-name [0] MTAName,
                    initiator-credentials [1] InitiatorCredentials,
                    security-context [2] SecurityContext OPTIONAL } }
        RESULT CHOICE {
                NULL,           -- if no authentication is required
                [1] SET {       -- if authentication is required
                    responder-name [0] MTAName,
                    responder-credentials [1] ResponderCredentials } }
        BIND-ERROR INTEGER {
                busy (0),
                authentication-error (2),
                unacceptable-dialogue-mode (3)
                unacceptable-security-context (4) } (0 . . ub-integer-options)

MTAUnbind ::= ABSTRACT-UNBIND
        FROM { transfer }
```

FIGURE 4/X.411 (Part 3 of 13)

Abstract syntax definition of the MTA abstract service

```
-- Transfer port

MessageTransfer ::= ABSTRACT-OPERATION
        ARGUMENT Message

ProbeTransfer ::= ABSTRACT-OPERATION
        ARGUMENT Probe

ReportTransfer ::= ABSTRACT-OPERATION
        ARGUMENT Report

Message ::= SEQUENCE {
        envelope MessageTransferEnvelope,
        content Content }

Probe ::= ProbeTransferEnvelope

Report ::= SEQUENCE {
        envelope ReportTransferEnvelope,
        content ReportTransferContent }
```

FIGURE 4/X.411 (Part 4 of 13)

Abstract syntax definition of the MTA abstract service

```
-- Message transfer envelope

MessageTransferEnvelope ::= SET {
        COMPONENTS OF PerMessageTransferFields,
        per-recipient-fields [2] SEQUENCE SIZE (1 . . ub-recipients) OF
                        PerRecipientMessageTransferFields }

PerMessageTransferFields ::= SET {
        message-identifier MessageIdentifier,
        originator-name OriginatorName,
        original-encoded-information-types OriginalEncodedInformationTypes OPTIONAL,
        content-type ContentType,
        content-identifier ContentIdentifier OPTIONAL,
        priority Priority DEFAULT normal,
        per-message-indicators PerMessageIndicators DEFAULT { },
        deferred-delivery-time [0] DeferredDeliveryTime OPTIONAL,
        per-domain-bilateral-information [1] SEQUENCE OF PerDomainBilateralInformation OPTIONAL,
        trace-information TraceInformation,
        extension [3] EXTENSIONS CHOSEN FROM {
                recipient-reassignment-prohibited,
                dl-expansion-prohibited,
                conversion-with-loss-prohibited,
                latest-delivery-time,
                originator-return-address,
                originator-certificate,
                content-confidentiality-algorithm-identifier,
                message-origin-authentication-check,
                message-security-label,
                content-correlator,
                dl-expansion-history,
                internal-trace-information} DEFAULT { } }
```

FIGURE 4/X.411 (Part 5 of 13)

Abstract syntax definition of the MTA abstract service

```
PerRecipientMessageTransferFields ::= SET {
        recipient-name RecipientName,
        originally-specified-recipient-number [0] OriginallySpecifiedRecipientNumber,
        per-recipient-indicators [1] PerRecipientIndicators,
        explicit-conversion [2] ExplicitConversion OPTIONAL,
        extension [3] EXTENSIONS CHOSEN FROM {
                    originator-requested-alternate-recipient,
                    requested-delivery-method,
                    physical-forwarding-prohibited,
                    physical-forwarding-address-request,
                    physical-delivery-modes,
                    registered-mail-type,
                    recipient-number-for-advice,
                    physical-rendition-attributes,
                    physical-delivery-report-request,
                    message-token,
                    content-integrity-check,
                    proof-of-delivery-request,
                    redirection-history } DEFAULT { } }
```

FIGURE 4/X.411 (Part 6 of 13)

Abstract syntax definition of the MTA abstract service

```
-- Probe transfer envelope
ProbeTransferEnvelope ::= SET {
        COMPONENT OF PerProbeTransferFields,
        per-recipient-field [2] SEQUENCE SIZE (1 . . ub-recipient) OF PerRecipientProbeTransferFields }
PerProbeTransferFields ::= SET {
        probe-identifier ProbeIdentifier,
        originator-name OriginatorName,
        original-encoded-information-types OriginalEncodedInformationTypes OPTIONAL,
        content-type-ContentType,
        content-identifier ContentIdentifier OPTIONAL,
        content-length [0] ContentLength OPTIONAL,
        per-message-indicators PerMessageIndicators DEFAULT { },
        per-domain-bilateral-information [1] SEQUENCE SIZE (1 . . ub-transfers) OF
                    PerDomainBilateralInformation OPTIONAL,
        trace-information TraceInformation,
        extensions [3] EXTENSIONS CHOSEN FROM {
                    recipient-reassignment-prohibited,
                    dl-expansion-prohibited,
                    conversion-with-loss-prohibited,
                    originator-certificate,
                    message-security-label,
                    content-correlator,
                    probe-origin-authentication-check,
                    dl-expansion-history,
                    internal-trace-information} DEFAULT { } }
PerRecipientProbeTransferFields ::= SET {
        recipient-name RecipientName,
        originally-specified-recipient-number [0] OriginallySpecifiedRecipientNumber,
        per-recipient-indicators [1] PerRecipientIndicators,
        explicit-conversion [2] ExplicitConversion OPTIONAL,
        extensions [3] EXTENSIONS CHOSEN FROM {
                    originator-requested-alternate-recipient,
                    requested-delivery-method,
                    physical-rendition-attributes,
                    redirection-history } DEFAULT { } }
```

FIGURE 4/X.411 (Part 7 of 13)

Abstract syntax definition of the MTA abstract service

```
-- Report transfer envelope

ReportTransferEnvelope ::= SET {
        report-identifier ReportIdentifier,
        report-destination-name ReportDestinationName,
        trace-information TraceInformation,
        extensions [1] EXTENSIONS CHOSEN FROM {
                message-security-label,
                originator-and-DL-expansion-history,
                reporting-DL-name,
                reporting-MTA-certificate,
                report-origin-authentication-check,
                internal-trace-information} DEFAULT { } }


-- Report transfer content

ReportTransferContent ::= SET {
        COMPONENT OF PerReportTransferFields,
        per-recipient-fields [0] SEQUENCE SIZE (1 .. ub-recipients) OF
                PerRecipientReportTransferFields }

PerReportTransferFields ::= SET {
        subject-identifier SubjectIdentifier,
        subject-intermediate-trace-information SubjectIntermediateTraceInformation OPTIONAL,
        original-encoded-information-types OriginalEncodedInformationTypes OPTIONAL,
        content-type ContentType OPTIONAL,
        content-identifier ContentIdentifier OPTIONAL,
        returned-content [1] Content OPTIONAL,
        additional-information [2] AdditionalInformation OPTIONAL,
        extensions [3] EXTENSIONS CHOSEN FROM {
                content-correlator } DEFAULT { } }

PerRecipientReportTransferFields ::= SET {
        actual-recipient-name [0] ActualRecipientName,
        originally-specified-recipient-number [1] OriginallySpecifiedRecipientNumber,
        per-recipient-indicator [2] PerRecipientIndicators,
        last-trace-information [3] LastTraceInformation,
        originally-intended-recipient-name [4] OriginallyIntendedRecipientName OPTIONAL,
        supplementary-information [5] SupplementaryInformation OPTIONAL,
        extensions [6] EXTENSIONS CHOSEN FROM {
                redirection-history,
                physical-forwarding-address,
                recipient-certificate,
                proof-of-delivery } DEFAULT { } }
```

FIGURE 4/X.411 (Part 8 of 13)

Abstract syntax definition of the MTA abstract service

```
-- Envelope and report content fields
MessageIdentifier ::= MTSIdentifier
OriginatorName ::= ORAddressAndOptionalDirectoryName
PerDomainBilateralInformation ::= SEQUENCE {
        country-name CountryName,
        CHOICE {
                    administration-domain-name AdministrationDomainName,
                    SEQUENCE {
                        administration-domain-name [0] AdministrationDomainName,
                        private-domain-identifier [1] PrivateDomainIdentifier OPTIONAL } },
        bilateral-information BilateralInformation }
BilateralInformation ::= ANY        --maximum ub-bilateral-info octets including all encoding
RecipientName ::= ORAddressAndOptionalDirectoryName
OriginallySpecifiedRecipientNumber ::= INTEGER (SIZE (1 . . ub-recipients))
PerRecipientIndicators ::= BIT STRING {
        responsibility (0),
        -- reponsible 'one', not-responsible 'zero'
        originating-MTA-report (1),
        originating-MTA-non-delivery-report (2),
        -- either originating-MTA-report, or originating-MTA-non-delivery-report, or both, shall be 'one':
        -- originating-MTA-report bit 'one' requests a 'report';
        -- originating-MTA-non-delivery-report bit 'one' requests a 'non-delivery-report';
        -- both bits 'one' requests and 'audited-report';
        -- bits 0-2 'don't care' for Report Transfer Content
        originator-report (3),
        originator-non-delivery-report (4),
        -- at most one bit shall be 'one':
        -- originator-report bit 'one' requests a 'report';
        -- originator-non-delivery-report bit 'one' requests a 'non-delivery-report';
        -- both bits 'zero' requests 'no-report'
        reserved-5 (5),
        reserved-6 (6),
        reserved-7 (7),
        -- reserved-bits 5-7 shall be 'zero' -- } (SIZE (8 . . ub-bit-options))
ProbeIdentifier ::= MTSIdentifier
```

FIGURE 4/X.411 (Part 9 fo 13)

Abstract syntax definition of the MTA abstract service

```
ReportIdentifier ::= MTSIdentifier
ReportDestinationName ::= ORAddressAndOptionalDirectoryName
SubjectIdentifier ::= MessageOrProbeIdentifier
MessageOrProbeIdentifier ::= MTSIdentifier
SubjectIntermediateTraceInformation ::= TraceInformation
AdditionalInformation ::= ANY        -- maximum ub-additional-info octets including all enconding
ActualRecipientName ::= ORAddressAndOptionalDirectoryName
LastTraceInformation ::= SET {
        arrival-time [0] ArrivalTime,
        converted-encoded-information-type ConvertedEncodedInformationTypes OPTIONAL,
        report-type [1] ReporType }
OriginallyIntendedRecipientName ::= ORAddressAndOptionalDirectoryName
```

FIGURE 4/X.411 (Part 10 of 13)

Abstract syntax definition of the MTA abstract service

```
-- Extension fields
originator-requested-alternate-recipient EXTENSION
        OriginatorRequestedAlternateRecipient
        ::= 2
OriginatorRequestedAlternateRecipient ::= ORAddressAndOptionalDirectoryName
internal-trace-information EXTENSION
        InternalTraceInformation
        ::= 38
```

FIGURE 4/X.411 (Part 11 of 13)

Abstract syntax definition of the MTA abstract service

```
InternalTraceInformation ::= SEQUENCE SIZE (1 .. ub-transfers) OF InternalTraceInformationElement
InternalTraceInformationElement ::= SEQUENCE {
        global-domain-identifier GlobalDomainIdentifier,
        mta-name MTAName,
        mta-supplied-information MTASuppliedInformation }
MTASuppliedInformation ::= SET {
        arrival-time [0] ArrivalTime,
        routing-action [2] RoutingAction,
        attempted CHOICE {
                mta MTAName,
                domain GlobalDomainIdentifier } OPTIONAL,
        -- additional-actions-- COMPONENTS OF InternalAdditionalActions }
InternalAdditionalActions ::= AdditionalActions
```

FIGURE 4/X.411 (Part 12 of 13)

Abstract syntax definition of the MTA abstract service

```
-- Common parameter types
TraceInformation ::= [APPLICATION 9] SEQUENCE (SIZE (1 .. ub-transfers) OF TraceInformationElement
TraceInformationElement ::= SEQUENCE {
        global-domain-identifier GlobalDomainIdentifier,
        domain-supplied-information DomainSuppliedInformation }
DomainSuppliedInformation ::= SET {
        arrival-time [0] ArrivalTime,
        routing-action [2] RoutingAction,
        attempted-domain GlobalDomainIdentifier OPTIONAL,
        -- additional-actions-- COMPONENT OF AdditionalActions }
AdditionalActions ::= SET {
        deferred-time [1] DeferredTime OPTIONAL,
        converted-encoded-information-types ConvertedEncodedInformationTypes OPTIONAL,
        other-actions [3] OtherActions DEFAULT {} }
RoutingAction ::= ENUMERATED {
        relayed (0),
        rerouted (1) }
DeferredTime ::= Time
ArrivalTime ::= Time
OtherActions ::= BIT STRING {
        redirected (0),
        dl-operation (1) } (SIZE (0 .. ub-bit-options))
END  -- of MTA abstract service
```

FIGURE 4/X.411 (Part 13 of 13)

Abstract syntax definition of the MTA abstract service

SECTION 4 – PROCEDURES FOR DISTRIBUTED OPERATION OF THE MTS

14 Procedures for distributed operation of the MTS

This paragraph specifies the procedures for distributed operation of the MTS, which are performed by MTAs. Each MTA individually performs the procedures described below; the collective action of all MTAs provides the MTS Abstract Service to the users of the MTS.

Although the procedures include most of the important actions required of an MTA, considerable detail has been omitted for clarity of exposition and to avoid unnecessary redundancy. The abstract-service definitions should be consulted for a definitive treatment of MTA actions.

14.1 *Overview of the MTA model*

14.1.1 *Organization and modelling technique*

The description of procedures for a single MTA is based on the model shown in Figures 5/X.411 through 11/X.411 and described below. It should be noted that the model is included for expositional purposes only and is not intended to constrain in any way the implementation of an MTA.

Neither the procedures shown nor the order of processing steps in them necessarily imply specific characteristics of an actual MTA.

The model distinguishes between *modules* and *procedures*. *Modules*, in the sense used here, are autonomous processing entities which can be invoked by other modules or by events external to the MTA, and which can in turn invoke other modules or generate external events. Modules are not bound together by an explicitly described control structure; rather the control structure among modules arises from the pattern of cross invocations. Modules correspond to *objects* in the sense of object-oriented programming.

Procedures are used here in the conventional programming sense. Procedures are task or function oriented. Procedures can call other procedures, subroutine fashion, with control returning to the calling procedure when the called procedure has completed. Such calls can be nested to arbitrary depth, and a procedure can call itself recursively. Procedures are bound together by explicitly defined control structures built from procedure calls and such conventional programming devices as iteration and conditional execution.

In the model procedures exist within modules. Each module contains at least one procedure and can contain several. In the latter case, the procedures and governing control structure are described explicitly. In the former case the existence of a module's single procedure is usually treated as implicit.

Using these modelling techniques, an MTA application process can be refined as follows: for each abstract-operation (whether consumer or supplier) that can exist between an MTA and the MTS-users it serves, or between an MTA and the other MTAs with which it cooperates there is a single module called an *external module*. The set of external module is responsible for the input and output of messages, probes, and reports into and out of the MTA and for the support of such operations as MTS-bind, MTS-unbind, Register, Submission-control and Delivery-control. The external modules are shown in Figure 5/X.411 and described in §§ 14.5 through 14.10, grouped by port.

In order to perform the various abstract-operations for which it is responsible, an MTA must perform certain processing operations on each message, probe, or report that enters, or originates within it. In the model these are the province of *internal modules*, shown in Figure 6/X.411 and described in §§ 14.2 through 14.4.

The external and internal modules relate to one another as follows: an external module comunicates only with an internal module, and not with another external module or directly with a procedure within an internal module. Thus, the internal modules not only support the bulk of processing within an MTA, but also serve as links between its external modules. In addition to the internal modules Figure 6/X.411 also shows the external modules with which they communicate.

The MTA is event driven in that it remains quiescent until an event is detected on one of its ports. Many events, such as the invocation of a MTS-bind, Submission-control, Delivery-control or Register abstract-operation by an MTS-user or another MTA, are dealt with directly and completely by the module assigned to that abstract-operation. However other events trigger processing that can reverberate through the MTA, endure over time and ultimately trigger one or more output events. Is is these events that engage the internal processing modules. They are:

a) a message or probe originated by a locally supported MTS-user enters via the submission-port;

b) a message, probe or report relayed from another MTA enters via the transfer-port.

Because the processing within an MTA can become rather complex, especially for messages with multiple recipients, the model assumes, as an internal bookkeeping device, that each message carries with it a set of instructions, one for the message as a whole, and one for each recipient. These instructions help guide a message through the processing steps and convey information between the modules and procedures internal to the MTA.

Note 1 – The procedures described herein focus on the processing of a single message. This is adequate in all but one respect: the queuing of messages and the relative prioriry of procedure invocation are driven explicitly by the argument **priority** in case of a message which enters via the submission - or the transfer-port, or implicitly (of urgent priority) in the case of a report or a probe which is generated internally or enters via the transfer-port.

Note 2 – An MTA can specify several default delivery time windows for each message priority e.g. those values defined in the F.400 series Recommendations. The MTS and therefore each MTA involved should take such values into account during message processing. For example, the MTA can apply a maximum delivery deadline. If that time period expires prior to delivery, the MTA generates a non-delivery-report and discards the message. The required actions in this case are identical to the actions required when **latest-delivery-time** is reached.

Note 3 – The discussion of trace-information is incomplete due to its complex nature. Some important details are highlighted but the complete and definitive treatment of trace-information appears in § 12.3.1.

14.2 *Deferred delivery module*

This module provides the Deferred Delivery element-of-service. It is invoked by the Message-submission and Message-in modules which pass a message to be checked for deferred delivery request and held if necessary. It invokes the Main module, passing on the message upon completion of its single internal procedure.

14.2.1 *Deferred delivery procedure*

14.2.1.1 *Arguments*

A message to be checked for deferred delivery request and held if necessary.

14.2.1.2 *Results*

The message is returned after expiration of the **deferred-delivery-time**. If deferred occurred, an arrival timestamp accompanies the message.

14.2.1.3 *Errors*

None.

14.2.1.4 *Procedure description*

The message is checked for presence of the **deferred-delivery-time** field. If absent the procedure returns the message and terminates. If present the **deferred-delivery-time** is checked against current time. If the **deferred-delivery-time** has expired, the procedure returns the message and terminates.

Otherwise, in the case of a relayed message, the MTA checks for a bilateral agreement obligating it to provide deferred delivery for this message. If absent the procedure returns the message and terminates.

Otherwise depending on bilateral agreement or intra-domain policy the current time is noted as the message arrival time and the message is held until expiration of the **deferred-delivery-time**. The message and timestamp are then returned as result. The procedure then terminates.

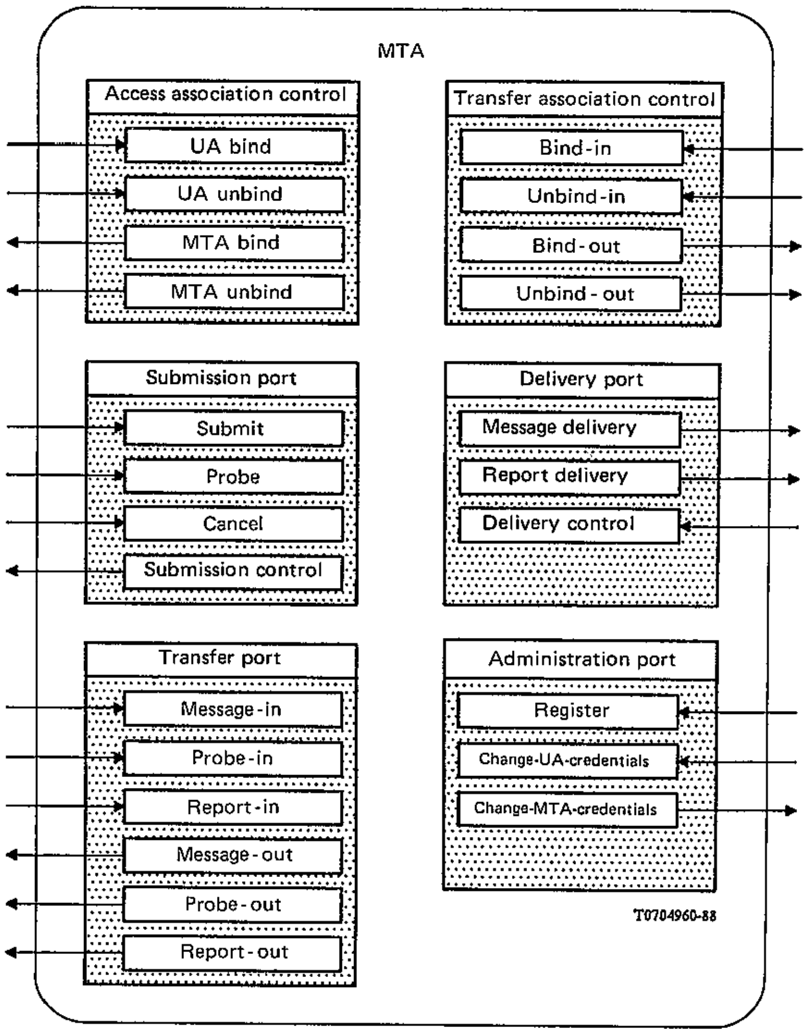


FIGURE 5/X.411

Ports and modules of an MTA

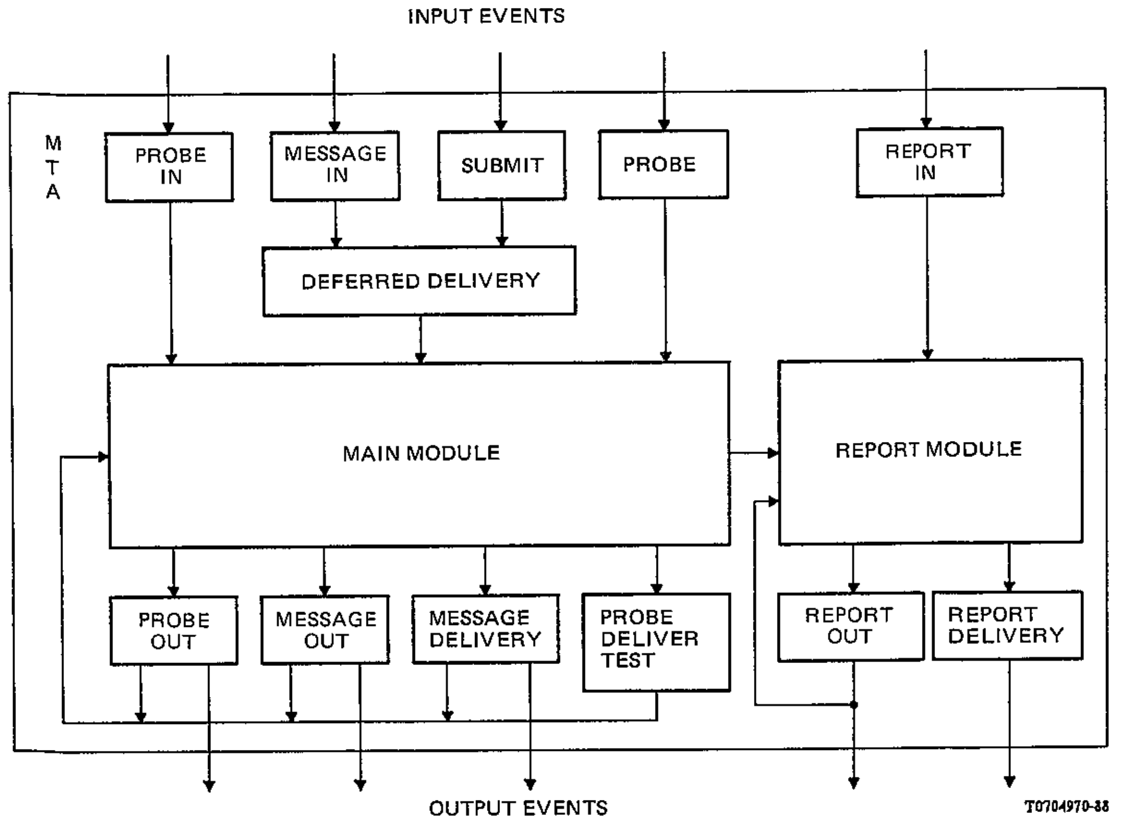


FIGURE 6/X.411

Relationship of internal and external modules

14.3 *Main module*

The Main module performs the bulk of processing on messages and probes entering the MTA. Figure 6/X.411 shows the relationships between the main module and the modules which it can invoke or be invoked by. The main module is subject to invocation by:

1) the Probe-in module, which passes a probe;
2) the Deferred-delivery module, which passes a message;
3) the Probe module, which passes a probe.

In the case of an error condition or the need for a positive delivery report, the main module can also be invoked by:

4) the Message-out module, which passes a message with per-message instruction indicating the problem encountered;
5) the Probe-out module, which passes a probe with per-message instruction indicating the problem encountered;
6) the Message-delivery module, which passes a message with per-recipient instruction indicating the problem(s) and/or success(es) encountered;
7) the Probe-delivery-test module, which passes a probe with per-recipient instructions indicating the problem(s) or success(es) encountered.

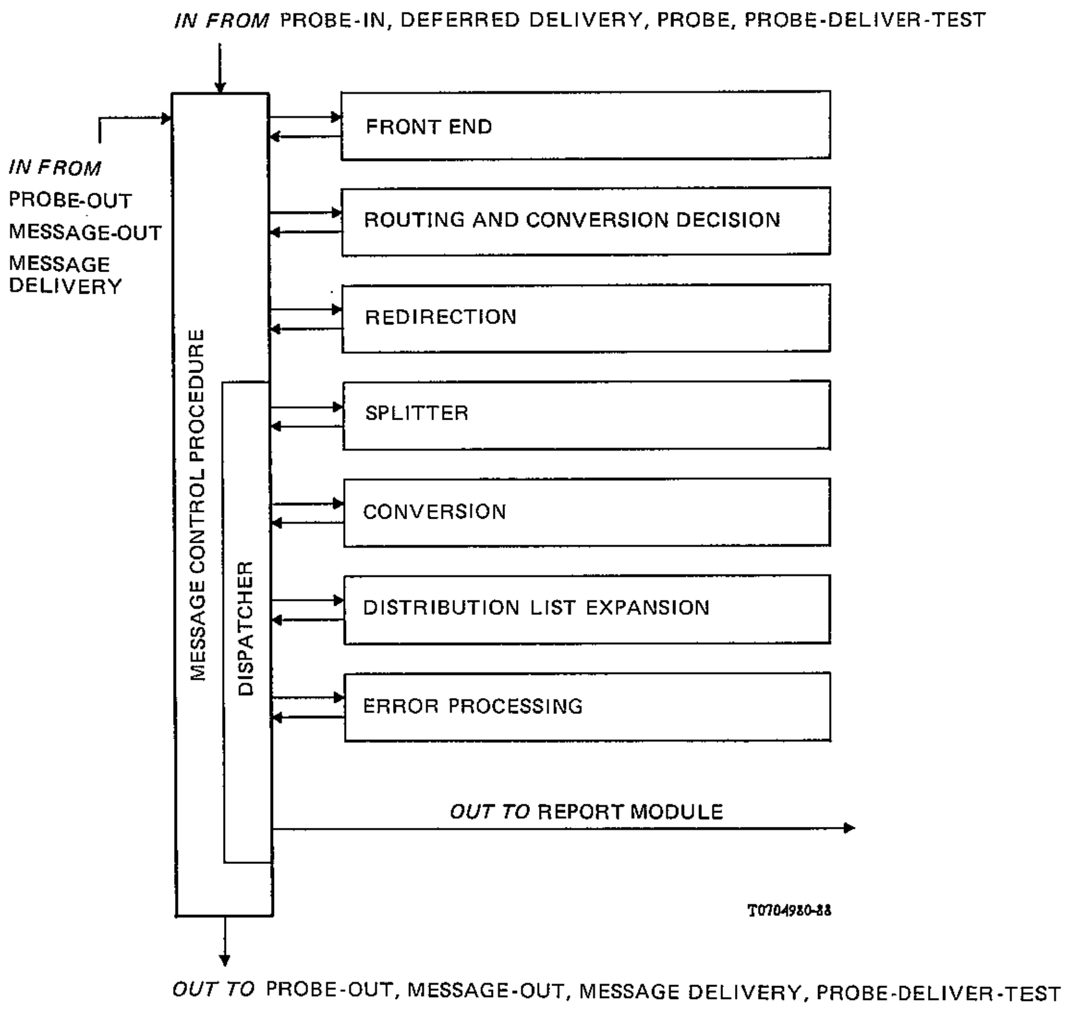
The Main module contains procedures which collectively, support the following functions:

Trace processing

– Loop detection
– Routing and rerouting

– Recipient redirection
– Content conversion
– Distribution list expansion
– Message replication
– Origin authentication of messages and probes
– Name resolution.

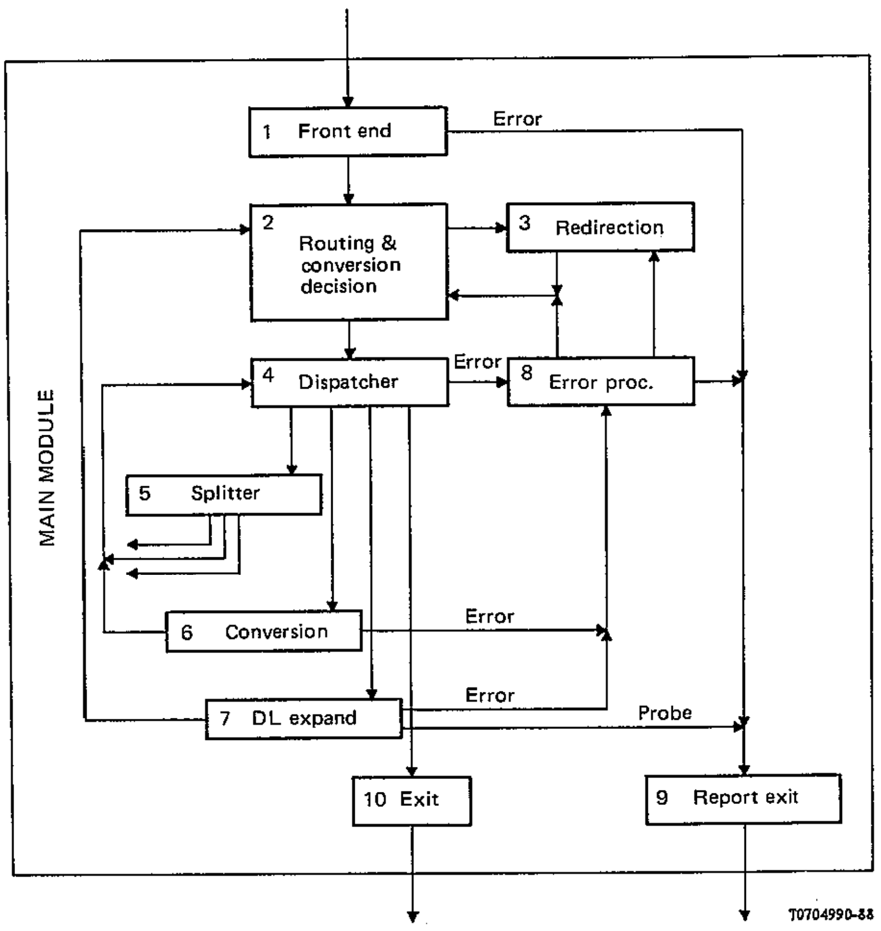
The procedures that perform these functions are called by a single Control procedure that guides the processing of each message or probe received by the Main module. Figure 7/X.411 shows the organization of the Control and subsidiary procedures within the main module; Figure 8/X.411 shows the flow of information through these procedures.



IN FROM PROBE-IN, DEFERRED DELIVERY, PROBE, PROBE-DELIVER-TEST
IN FROM PROBE-OUT MESSAGE-OUT MESSAGE DELIVERY
MESSAGE CONTROL PROCEDURE
DISPATCHER
FRONT END
ROUTING AND CONVERSION DECISION
REDIRECTION
SPLITTER
CONVERSION
DISTRIBUTION LIST EXPANSION
ERROR PROCESSING
OUT TO REPORT MODULE
T0704980-88
OUT TO PROBE-OUT, MESSAGE-OUT, MESSAGE DELIVERY, PROBE-DELIVER-TEST

FIGURE 7/X.411

Organization of procedures within the main module



Note – Numbers in this Figure refer to the numbered steps in the control procedure's logic (§ 14.3.1.4).

FIGURE 8/X.411

Information flow within the main module

For each message or probe received, the Main module calls the Control procedure with that message or probe as argument. As result, the Control procedure returns one or more replicas of the message or probe with appropriate instructions attached. Depending on the nature of these instructions the Main module then invokes:

1) the message-out module, to which it passes each message with a per-message transfer instruction;
2) the probe-out module, to which it passes each probe with a per-message transfer instruction;
3) the message-delivery module, to which it passes each message with one or more per-recipient delivery instructions;
4) the probe-delivery-test module, to which it passes each probe with one or more per-recipient delivery instructions;
5) the report module, to which it passes each message or probe with a per-message instructions and/or one or more per-recipient instructions indicating report generation.

14.3.1 *Control procedure*

This procedure directs each incoming message or probe through the remaining procedures of the Main module. The overall flow of information is shown in Figure 8/X.411.

14.3.1.1 *Arguments*

One of the following (these arguments correspond to the messages and probes that can be passed to the Main module upon invocation):

1) a message or probe without instructions (from the probe-in or probe module);
2) a message without instructions but with optional arrival timestamp (from the deferred-delivery module);
3) a message or probe with per-message instruction describing a transfer problem (from the message-out or probe-out module);
4) a message or probe with per-recipient instructions describing delivery problems or successes (from the message-delivery or probe-delivery-test module).

14.3.1.2 *Results*

1) One or more replicas of the message or probe argument each accompanied by a per-message instruction indicating transfer, and/or
2) one or more replicas of the message or probe argument each accompanied by one or more per-recipient instructions indicating delivery or delivery test, and/or
3) one or more replicas of the message or probe argument each accompanied by one or more per-recipient instructions indicating report generation.

14.3.1.3 *Errors*

None. Error conditions are accounted for in the results described above.

14.3.1.4 *Procedure description*

1) A message or probe without instructions:

   The Front-end procedure is first called to perform trace initialization and several per message checks such as message expiration and routing loop detection.

   Upon a return with report instruction indicating a problem with the message processing continues at step 9.

   On all other returns processing continues below.

2) Routing-and-conversion-decision procedure is called to compute per-recipient routing and conversion instructions. (These are complete instructions that will direct the message or probe through the remainder of the procedures.)

   If a redirection instruction is indicated (e.g., recipient-requested-alternate-recipient), processing continues at step 3.

   Otherwise, processing continues at step 4 (dispatcher).

3) Redirection is called. Upon successful return, processing continues at step 2.

   In the case of an unsuccessful return, processing continues at step 8 (error-handler).

4) Dispatcher. The dispatcher acts on the generated instructions and passes control to the first of the following procedures that is applicable:

   – splitting (step 5);
   – conversion (step 6);
   – distribution-list-expansion (step 7);
   – error-processing (step 8) in case the decision process encountered a problem, e.g., routing error;
   – exit (step 10).

5) Splitter is called for replication as required by the per-recipient instructions generated in routing-and-conversion-decision procedure. For each replica processing continues individually at step 4 (dispatcher).

6) Conversion is called for each message or probe needing conversion.

   Upon successful return of the message or probe, processing continues at step 4 (dispatcher).

   Upon return with report instruction indicating a conversion error, processing continues at step 8 (error-handler).

7) The DL-expansion procedure is called.

   Upon successful return of a message, processing continues at step 2 so that the recipients resulting from DL expansion can be properly dealt with.

If a copy of the message with delivery report instructions is returned, in place of or in addition to the above return, its processing continues at step 9.

A probe returning successfully will have report instructions; processing continues at step 9 (report-generation).

Upon return of a message or probe with report instruction indicating DL expansion error-processing continues at step 8.

8) This is the collection point that processing reaches upon detection that a message or probe cannot be handled by the main line procedures. The error-processing procedure is called to seek another delivery method or an alternate-recipient. Upon successful return the error-processing procedure indicates the new recipient in an instruction to the Routing-and-conversion-decision procedure (step 2), where processing continues.

   If redirection is possible, the message or probe is passed to the report generator (step 9).

9) The control procedure terminates at this point and returns a message or probe with report generation instructions.

10) When a message or probe reaches this point the control procedure terminates.

14.3.2 *Front-end procedure*

This procedure performs trace initialization, detection of message expiration, initial security check, loop detection, and criticality check.

14.3.2.1 *Arguments*

A message or probe and an optional arrival timestamp.

14.3.2.2 *Results*

The message, or probe with initialized trace information for this MTA.

14.3.2.3 *Errors*

The message or probe with report generation instructions detailing the problem encountered.

14.3.2.4 *Procedure description*

1) If the message has crossed a domain boundary, a **trace-information-element** for this domain is added with **relay** as action. If an arrival time accompanies the message, then delivery deferral has occurred and **deferred-time** is set to the current time and **arrival-time** is set to the accompanying timestamp value. Otherwirse no deferral has occurred and the **arrival-time** is set to the current time. An **internal-trace-information-element** is also added whether or not the message has crossed a domain boundary.

2) If required by the security policy in force and/or if the message-origin-authentication-check is incorrect, the procedure returns a report generation instruction. The values of the non-delivery-reason- code and non-delivery-diagnostic-code are set to unable-to-transfer, and secure-messaging-error, respectively.

3) If any of the extension fields is marked critical for relaying but is not semantically understood by the MTA, the procedure returns a report generation instruction. The non-delivery-reason-code is set to transfer-failure and the non-delivery-diagnostic-code to unsupported-critical-function. The procedure then terminates.

4) If the latest-delivery-time has passed, or the system's maximum transit time has elapsed for the message's priority, the procedure returns a report generation instruction. The non-delivery-reason-code is set to unable-to-transfer and the non-delivery-diagnostic-code is set to maximum-time-expired. The procedure then terminates.

5) Loop detection is performed. The loop detection algorithm is beyond the scope of this Recommendation. However, an example of a combined routing and loop detection algorithm is given in § 14.3.11. If a loop is detected, the procedure returns a report generation instruction. 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.

14.3.3 *Routing-and-conversion-decision procedure*

For each of a message or probe's recipients for which the MTA is responsible, this procedure determines the routing and conversion actions, if any, to be taken by this MTA. The actions are recorded as per-recipient instructions associated with the message. The actions are subsequently carried out by other sub-procedures within the internal procedure, or elsewhere in the MTA.

Note – This procedure may be called multiple times for any particular message. In such cases, the procedure ignores per recipient instructions generated by previous calls to this procedure which have not yet been acted upon elsewhere.

14.3.3.1 *Arguments*

1) A message or probe with **responsibility** true for those recipients of concern to this MTA.

14.3.3.2 *Results*

The message or probe that formed the procedure's argument plus new or revised per-recipient instructions indicating what routing and possible conversion action should be taken by this MTA.

14.3.3.3 *Errors*

None. Error conditions, if any, are noted in the per-recipient instructions.

14.3.3.4 *Procedure description*

Each recipient is considered in turn. If **responsibility** is false, the recipient is ignored. Otherwise, the Routing-decision and Conversion-decision procedures are called in turn for this recipient. When all recipients have been considered in this way the procedure terminates. See Figure 9/X.411.

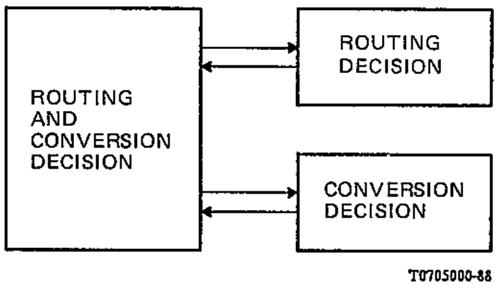


FIGURE 9/X.411

Organization of procedures within routing and conversion decision procedure

14.3.4 *Routing-decision procedure*

This procedure generates a routing instruction for a single message recipient.

14.3.4.1 *Arguments*

1) A message recipient plus the per-recipient instruction, if any, applicable to this recipient.
2) The per-message instruction, if any, applicable to this message. Other message fields are also accessible to the procedure as required.

14.3.4.2 *Results*

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

a) relay to another MTA;
b) deliver to a local recipient;
c) expand the distribution list represented by this recipient;
d) generate a report indicating delivery failure. The **non-delivery-reason-code** and **non-delivery-diagnostic- code** are included in the instruction;
e) redirect to a recipient specified alternate recipient.

14.3.4.3 *Errors*

None. Error conditions are recorded in the routing instruction.

14.3.4.4 *Procedure description*

The procedure is described in the following steps.

Note – To ensure the security-policy is not violated during routing, the **message-security-label** should be checked as appropriate against the **security-context**.

1) If there is a per-message instruction indicating a previous relay failure, then the procedure attempts to compute an alternate next hop destination for this recipient. The choice of routing algorithm is beyond the scope of this Recommendation. However, an example of an applicable algorithm is contained in clause 14.3.11. If successful, then the message's **internal-trace-information** is updated with a **rerouted** routing-action to reflect the fact that the message has been re-routed (see § 12.3.1). If the message was to have crossed a domain boundary then the **trace-information** is also updated accordingly. The procedure returns a relay instruction to the alternate destination and terminates.

   If no alternate next hop is available or all available next hops have already been tried unsuccessfully or prohibited, then the procedure returns a report generation instruction for this recipient. The **non-delivery-reason-code** is set to **transfer-failure** and the **non-delivery-diagnostic-code** is set as appropriate to the realy failure encountered. The procedure then terminates.

2) 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 Report-delivery procedure. The procedure then terminates.

3) If the recipient is a distribution list for which this MTA serves as expansion point, then the message's **DL-expansion-prohibited** argument is examined. If the value is **DL-expansion-allowed** then the procedure returns a routing instruction (subject to the security-policy in force) to expand the distribution list and terminates.

   If the value is **DL-expansion-prohibited**, or the security prohibits the use of a DL, then the procedure returns a report generation instruction for this recipient. The **non-delivery-reason-code** is set to **unable-to-transfer** and **non-delivery-diagnostic-code** to **DL-expansion-prohibited**. The procedure then terminates.

   In all cases other than the above, the following steps are taken.

4) If the recipient appears to be local, that is, an MTS-user directly supported by this MTA, then the following steps are taken.

   a) The **OR-address** is checked to ensure that it unambiguously specifies an actual local recipient. Otherwise 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 **unrecognized-OR-name** or **ambiguous-OR-name** as appropriate. The procedure then terminates.

   b) If the **OR-address** unambiguously specifies an actual local recipient, then the recipient registration parameters are checked for recipient-requested-alternate-recipient. In the determination of an alternate-recipient the **user-security-label** should be checked against the **message-security-label** to ensure no violation of the security-policy occurs.

      If **recipient-assigned-alternate-recipient** is in effect, allowed by the **recipient-reassignment-prohibited** field, and permitted by the security-policy, then a redirection instruction is generated and the procedure terminates.

      Otherwise the procedure returns a report instruction for this recipient and terminates. The **non-delivery-reason-code** is set to **unable-to-transfer** and the **non-delivery-diagnostic-code** is set as appropriate.

   c) If **recipient-requested-alternate-recipient** is not in effect, then the message is checked against the recipient's remaining registration parameters. For example the message's content length is compared to the recipient's **deliverable-maximum-content-length**, the message's **content-type** to the recipient's **deliverable-content-types**, etc. If no problem is encountered, then the Routing-decision procedure returns a delivery instruction for this recipient and terminates.

      If there is a problem between message and registration parameters, 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 as appropriate to the message problem encountered. The procedure then terminates.

5) If the recipient is not local to this MTA then the Routing-decision procedure attempts to determine a next hop instruction (subject to the security-policy in force) for this recipient. If successful, then a relay instruction to the next hop is returned and the procedure terminates.

   If a next hop cannot be determined, 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 as appropriate to the problem encountered. The procedure then terminates.

14.3.5 *Conversion-decision procedure*

This procedure generates a conversion instruction for a single message recipient.

14.3.5.1 *Arguments*

1) A message or probe recipient plus the per-recipient instruction, if any, applicable to this recipient.
2) Other message fields are also considered by the procedure:
   a) **original-encoded-information-types,**
   b) **implicit-conversion-prohibited,**
   c) **conversion-with-loss-prohibited,**
   d) **explicit-conversion.**

14.3.5.2 *Results*

1) A content conversion instruction applicable to this recipient, and possibly,
2) a revised routing instruction indicating Relay-out or Probe-out to an MTA able to perform the required conversion, or, in lieu of 1 and 2 above,
3) an instruction to generate a report indicating delivery failure. The **non-delivery-reason-code** and **non-delivery-diagnostic-code** are included in the instruction.

14.3.5.3 *Errors*

None. Error conditions are recorded in the routing instruction.

14.3.5.4 *Procedure description*

Note – As the circumstances under which a particular MTA stages conversion are left for further study, it is impractical to describe a procedure to decide what EITs are required for conversion output. For example, if an intermediate MTA stages the conversion, there is no standardized way to know the EITs that the MTS-user can handle. Consequently the following clauses assume that the EITs for conversion are known to the MTA.

1) If explicit conversion is required for this recipient, the procedure starts at step 6.
2) If implicit conversion is required but the recipient has not subscribed to the implicit conversion facility, the procedure returns a negative report instruction with the **non-delivery-reason-code conversion-not-performed** and the non-delivery-diagnostic-code implicit-conversion-not-subscribed. The procedure then terminates.
3) If the required conversion is impractical, the procedure generates a negative report instruction with the **non-delivery-reason-code conversion-not-performed** and the **non-delivery-diagnostic-code conversion-impractical**. The procedure then terminates.
4) If conversion would be required but is prohibited for the message, the procedure generates a negative report instruction with the **non-delivery-reason-code conversion-not-performed** and the **non-delivery-diagnostic-code conversion-prohibited**. The procedure then terminates.
5) If the required conversion would cause a loss of information and the **conversion-with-loss-prohibited** field has the value **with-loss-prohibited**, the procedure generates a negative report instruction with the **non-delivery-reason-code conversion-not-performed** and one of the following **non-delivery-diagnostic-codes**, as appropriate:
   – **line-too-long,**
   – **page-split,**
   – **pictorial-symbol-loss,**
   – **punctuation-symbol-loss,**
   – **alphabetical-character-loss, or**
   – **multiple-information-loss.**

The procedure then terminates.

6) If the required conversion is allowable, cannot be performed by this MTA, but can be performed by an MTA known to this MTA, then no conversion instruction is generated. The routing instruction previously generated is changed to Transfer-out or Probe-out, with a next hop destination appropriate to the MTA in question. The procedure then terminates.

7) If the required conversion can be performed by this MTA, the procedure returns an instruction to perform the conversion and terminates.

14.3.6 *Error-processing procedure*

When another procedure encounters a deliverability or routing error, this procedure is called to determine whether delivery or routing can be achieved by reassignment of the recipient or by choosing a different **OR-address** for the same recipient. If not, non-delivery must be signalled to the Report module. Errors provoking a call on this procedure include:

– **recipient-name** does not identify an MTS-user;

– delivery failure;

– MTA is unable to perform necessary conversion;

– transfer path problems;

– DL-expansion problems;

– security violations;

– conflict with Registration parameters.

Note – The action taken on error-processing shall be subject to the security-policy in force.

14.3.6.1 *Arguments*

1) A message or probe with the per-recipient fields that caused the problem.

2) Report instructions indicating the error.

14.3.6.2 *Results*

The message or probe in question with an updated **recipient-name** field, or

1) the message or probe in question;

2) report instructions.

14.3.6.3 *Errors*

None.

14.3.6.4 *Procedure description*

Note – This procedure may be called multiple times for a given recipient. Eventually all alternatives will be exhausted and step 5 executed to report failure.

1) The arguments are checked for inclusion of a **directory-name**. If present, the procedure performs a Directoy loo-up to determine a new **OR-address. The OR-address**, if any, thus extracted from the Directory is checked for satisfaction of the **requested-delivery-method** argument, if present. If the check succeeds, the new **OR-address** is substituted for the old and the procedure terminates.

Note – Following the substitution of the new **OR-address** for the original, the message may legitimately be routed to an MD/MTA that it has already visited. The technique used to prevent premature detection of a routing loop is for further study.

2) Otherwise the procedure determinates whether an **originator-requested-alternate-recipient** was specified for the recipient of concern. If so, the Redirection procedure is called with the message, relevant fields indicated, as argument. Upon successful return from Redirection, the procedure terminates, returning the now redirected message as result.

3) Otherwise the procedure checks for a delivery error, and if present checks the error's cause by examination of the **non-delivery-reason-code** and **non-delivery-diagnostic-code**. If the recipient **OR-address** does not identify an MTS-user, then the **per-message-indicators** are checked for **alternate-recipient-allowed**. If the value found is **alternate-recipient-allowed**, and the MTA has been configured with the address of an alternate-recipient for this class of recipient, then Redirection is called to redirect the message to the alternate-recipient. Upon successful return from Redirection, the procedure terminates, returning the now redirected message as result.

4) The handling of errors which can be resolved but are due to other than addressing problems is a local matter, for example routing to another MTA within the domain because of conversion problems.

5) If the delivery error is of a type other than those cited above, or if the value of **alternate-recipient-allowed** is **alternate-recipient-prohibited**, or if no suitable MD-specified alternate-recipient exists, then the procedure returns a report instruction and terminates.

14.3.7 *Redirection procedure*

This procedure redirects a message to an alternate-recipient.

Note – The use of redirection facilities shall be subject to the security-policy in force.

14.3.7.1 *Arguments*

1) The **OR-name** of the alternate-recipient to whom the message is to be redirected.
2) The per-recipient message fields for the recipient to be replaced by an alternate.
3) The message or probe which is to be redirected.
4) The redirection reason.

14.3.7.2 *Results*

The message or probe supplied in the third argument with the recipient identified in the second argument replaced by the alternate-recipient in the first argument.

14.3.7.3 *Errors*

An indication that a redirection loop has been detected.

14.3.7.4 *Procedure description*

1) The procedure first ensures that redirection to the specified alternate recipient would not result in a redirection loop. The **OR-name** of the alternate-recipient supplied in argument 1 is compared with each **intended-recipient-name** from the sequence of **redirection-history** from the per-recipient fields identified in argument 2. Upon a match the procedure terminates indicating that a redirection loop has been detected.
2) An element is appended to the **redirection-history** (which is created if not present), using the **recipient-name** from argument 2 to form the **intented-recipient-name**, obtaining the **redirection-reason** from argument 4 and containing the Time at which this redirection is performed. The **OR-name** supplied in the first argument is then substituted for that **recipient-name**.
3) In the **other-actions** field of the current **trace-information**, the value **redirected** is set to true.
4) The message transfer envelope is updated as follows:

recipient-name:
replaced

trace-information:
indicate **redirected**

redirection-history:
append previous **recipient-name** and **redirection-reason**

originator-requested-alternate-recipient:
deleted if, and only if the **redirection-reason** indicates **originator-requested-alternate-recipient**

14.3.8 *Splitter procedure*

The splitter replicates messages and probes as required for further processing. The replicas are modified as appropriate to correctly indicate the distribution of responsibility for the various recipients from the original. Each replica is accompanied by a per-message instruction indicating its further disposition within the MTA.

Note – The use of Splitter facilities shall be subject to the security-policy in force.

14.3.8.1 *Arguments*

A message or probe. For each recipient with **responsibility** true a per-recipient routing/conversion instruction accompanies the message.

14.3.8.2 *Results*

One or more replicas of the original message or probe with responsibility appropriately indicated, and a per-message instruction indicating the replica's further disposition within the MTA.

14.3.8.3 *Errors*

None.

14.3.8.4 *Procedure description*

The splitter examines the instructions generated by the Routing-and-conversion-decision procedure to (conceptually) segregate the recipients with **responsibility** true into groups. A replica is created for each group. Further processing for that replica (in other procedures) is dependent on the routing and conversion instructions applicable to the group it represents.

Note 1 – Message replication is required in an MTA because of the potentially differing treatment required for a message's various recipients. These differences arise from the need for more than one relaying path outward from an MTA, from the need for more than one conversion to be carried out on the message's content and from the need to expand distribution lists. For example when more than one relay path exists, a separate copy of the message must be created for each such path, with **responsibility** values as appropriate for the recipients lying along that path.

Note 2 – The determination of what replicas are needed is a local matter, undertaken to minimize the total number of such replicas created. The following paragraphs suggest one approach but are not intended to constrain in any way the approach followed in an actual implementation.

Note 3 – For simplicity of exposition, the Splitter is described as a single-pass algorithm. That is, all necessary replicas are created prior to any further processing. An important optimization would be to minimally split the message for conversion, and then to complete the splitting of the converted copies.

1) The procedure considers first those recipients for which content conversion instructions exist. These recipients are grouped such that the members of each group are subject to identical conversion instructions. A replica is created for each such group with **responsibility** true for the recipients in that group, false for all others.
2) The recipients are then examined for those for which DL-expansion instructions exist. A replica is created for each such DL recipient with **responsibility** false for all recipients but the single DL that yielded the replica.
3) The groups are further subdivided based on per-recipient routing instruction calls for Transfer-out or Probe-out. These recipients are grouped such that each group shares a common next hop destination. A replica is created for each such group with **responsibility** true for recipients in the group, false for all others. For all recipients in each such group, this will be either the first relay attempt of a rerouting attempt. In the latter case the trace-information for the message or probe is modified to indicate that this is a first or subsequent rerouting.
4) Finally, the routing instructions for some recipients will call for Message-delivery or Report-generation. A replica is created for each such subgroup with **responsibility** true for the recipients in the group, false for all others.
5) The procedure now terminates.

14.3.9 *Conversion-procedure*

This procedure performs conversions on messages and indicates those conversions that would have been performed on probes.

14.3.9.1 *Arguments*

A message or probe with the required conversion(s) indicated.

14.3.9.2 *Results*

The message or probe with conversions performed and indicated (just indicated in the case of a probe).

14.3.9.3 *Errors*

The message or probe with report instructions detailing the conversion problem encountered.

14.3.9.4 *Procedure description*

1) For a message, the conversion procedures for built in EITs are performed as defined in Recommendation X,408. The conversion procedures between externally defined EITs and between built in and externally defined EITs are outside the scope of this Recommendation.
2) Upon conversion the message or probe's **trace-information** for this domain is updated to show the converted EITs. The procedure now terminates.

14.3.10 *Distribution-list-expansion procedure*

This procedure takes a message with a single DL recipient and returns a message whose recipient list includes the members of the DL. For a probe it verifies whether DL-expansion would occur, if requested.

Note – The use of DL-expansion shall be subject to the security-policy in force.

14.3.10.1 *Arguments*

1) A message with information indicating the recipient DL which is to be expanded, or
2) a probe with information indicating the recipient DL whose expansion is to be verified.

14.3.10.2 *Results*

1) The message with zero or more recipients representing the DL's membership. Other fields can be updated as indicated in the procedure description below;
2) optionally, the message with report generation instructions to indicate successful delivery,
3) the probe with a report generation instruction.

14.3.10.3 *Errors*

1) A report instruction indicating delivery failure. Values for the **non-delivery-reason-code** and **non-delivery-diagnostic-code** are as indicated in the procedure description below.
2) In the case of DL recursion the procedure terminates without returning errors or results.

14.3.10.4 *Procedure description*

1) For a message (not a probe), do Recursion Detection: The components of the **DL-expansion-history** field are examined for an occurrence of the DL recipient's name. Note that a distinguished **OR-name** of the DL is used for recursion detection, and each expansion point is responsible for ensuring that only that **OR-name** is placed in the **DL-expansion-history**.

   If the DL recipients name is present in the **DL-expansion-history**, then the DL is recursively defined and shall not be expanded further. The message is discarded and no reports or other results are returned. The expansion procedure terminates.
2) DL acquisition: The expansion procedure attempts to acquire the DL attributes.

   If unsuccessful the procedure returns a report instruction with the **non-delivery-reason-code-unable-to-transfer** and **non-delivery-diagnostic-code** as appropriate. The procedure then terminates.
3) Submit permission verification: If it is a message (not a Probe), the last element of the **DL-expansion-history** field (if present) else the **originator-name** is considered to be the sender of the message. For a probe the originator is the sender of the message.

   The sender's name is compared against the components of the DL-submit-permission. If no match, return a report instruction with the **non-delivery-reason-code unable-to-transfer** and **non-delivery-diagnostic-code no-DL-submit-permission**. The procedure then terminates.
4) For a probe: If no other local policy would prevent an attempted delivery, then return a report instruction for successful delivery indication. Procedure then terminates.
5) For a message: The DL recipient's **responsibility** flag is set to false and the DL's members are added as new recipients of the message. The per-recipient fields for each new recipient are copied from that of the DL recipient, except as follows:

   – **recipient-name**: member of the DL.

   The following per-recipient fields are copied or changed according to local DL policy:

   – **DL-expansion-prohibited,**

   – **originating-MTA-report-request** (see Note 1),

   – **originator-report-request** (see Note 1),

– **originator-requested-alternate-recipient** (see Note 2),

– **explicit-conversion.**

Note 1 – Copy only if DL-policy requires and the originator would not receive unrequested reports.

Note 2 – The **originator-requested-alternate-recipient** can be removed, or replaced, according to local DL policy, or copied, but only if explicity required by DL policy.

Note 3 – Any DL-members that identify DLs that are already present in the **DL-expansion-history** may be excluded from the DL expansion and not included in the new recipients of the message.

6) In the **other-actions** field of the current **trace-information**, the value **dl-operation** is set to true.

7) The distinguished value of the DL's **OR-name** (including its **OR-address**) and the Time at which this expansion occurred are appended to the **DL-expansion-history** field of the message.

   Note – The use of a distinguished value of the DL's **OR-name** here refers not to distinguished **directory-name** but to a specific **OR-name** of the DL which the expansion point chooses to use for comparison purposes.

8) If the new report request values (determined in step 5) or the DL's local policy will prevent the originator from receiving a requested delivery report from the DL's members, then a copy of the message, with delivery report request instructions for the expanded DL, is constructed and returned along with the message.

9) The procedure returns the revised message and the optional report request and then terminates.

14.3.11 *Loop detection and routing algorithm*

The routing and loop detection algorithms for inter or intra domain use are beyond the scope of this Recommendation. In order to expose the issues that must be considered, the remainder of this clause describes one approach toward routing and loop detection. This material is not part of the Recommendation.

The paragraphs that follow describe a simple method of loop detection together with a minimal routing algorithm. The algorithm is minimal in the sense that it presupposes only minimal knowledge from each MD and performs transfer steps that avoid loops (in the sense indicated below). Of course, this algorithm can be improved any time an MD knows more about the topology of the network of MDs.

The algorithm recognizes the fact that it is in general legitimate (i.e. no loop should be detected) to re-enter an MD if a specific operation has been performed by another MD since the last passage through the MD about to be re-entered. Legitimate operations are: conversion, DL-expansion, and redirection.

1) Notation: The Trace Information sequence is made of **trace-information-elements** denoted in a simplified way as [MD, routing-action, operation], where MD is the name of an MD; routing-action is “relayed” or “rerouted”, operation is “conversion”, “DL-operation”, “redirection” or “nil”. M denotes the message to transfer. MD(o) denotes the current MD (the one currently doing loop detection). Neighbours is the set of selected adjacent MDs (neighbours of MD(o)), which are possible relay-MDs for M. Trace-Info* is the suffix of Trace-Info obtained by considering the tail of the trace info sequence beginning with the last [MD, r, op] trace info element where op is not nil (nil indicates that no operation has been performed by an MD).

2) Loop Detection: Examine Trace-Info for loops. A loop is detected if the trace info sequence contains a suffix, [MD(o), relayed, op(o)] . . . [MD(p), relayed, op(p)] where for all j of which o < j £ p the associated trace info element is [MD(j), relayed, op(j)] and op(j) = nil. That is, a loop is detected if M arrives at an MD which has already relayed it and each MD afterwards has also relayed it without performing any operation other than routing. If a loop is detected, then the algorithm returns an error indicating the problem, and terminates.

3) Routing Setup: If no loop is detected, the set, Neighbours, is adjusted, if necessary, for loop-avoiding transfer steps in the context of the current message. (The adjustment affects other message.)

   a) If there is no loop and no occurrence of [MD(o), r, op], in Trace-Info*, then Neighbours is unchanged.

   b) If there is no loop but there is an occurrence of [MD(o), r, op] in Trace-Info*, then remove from Neighbours all MDs which appear in that suffix of Trace-Info* which begins with [MD(o), r, op]. Modify the trace info element added by the current domain to show rerouted as routing action. Add a previous-MD parameter determined as follows: The last [MD(o), r, op] trace info element in Trace Info is located. The previous-MD is the MD appearing in the first trace info element after this last [MD(o), r, op] trace info element.

c) In cases a and b, if Neighbours is empty, the algorithm returns an error indicating the problem and terminates.

4) Routing action. A next hop is selected from Neighbours for each recipient to be relayed.

14.4 *Report module*

The Report module can be invoked by:

1) the Report-in module, which passes a report, or

2) the Main module, which passes a message or probe with report instructions;

3) the Report-out module, which passes a report with failure description.

If an error is encountered by the procedures internal to this module, no output is generated. Otherwise the Report module invokes the Report-out or Report-delivery module, passing a report with transfer or delivery instructions, respectively. See Figure 10/X.411.

Note – The use of reports shall be subject to the security-policy in force.

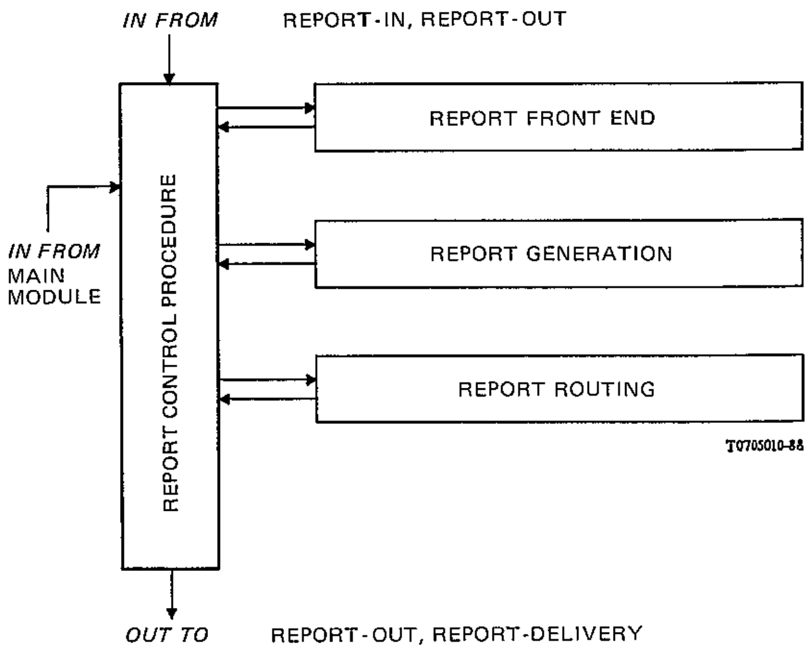


FIGURE 10/X.411

Organization of procedures within the report module

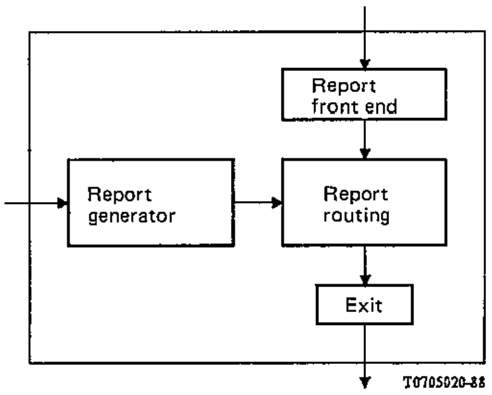


FIGURE 11/X.411

Information flow within the report module

14.4.1 *Control procedure*

14.4.1.1 *Arguments*

1) A report, or
2) a message or probe with report instructions.

14.4.1.2 *Results*

1) A report with relaying or delivery instructions, or
2) no result in case an error is encountered.

14.4.1.3 *Errors*

None. The report, message, or probe is discaded if an error is encountered.

14.4.1.4 *Procedure description*

1) For a report from report-in the report-front-end procedure is first called to perform trace initialization and several initial verification steps. A null return indicates an error; the report is discarded and processing terminates. Otherwise processing continues as step 3 below.
2) For a message or probe the Report-generation procedure is first called to create a report. A null return indicates an error; the message or probe is discarded and processing terminates. If a report is returned, processing continues at step 3, below.
3) The Report-routing procedure is called to generate a routing instruction for the report. A null return indicates an error; the report is discarded and processing terminates. In the case of a positive return the trace update procedure is now called to indicate passage through this MTA. The Control procedure returns the completed report together with routing instruction and terminates, subject to the security-policy.

14.4.2 *Report-front-end procedure*

This procedure performs trace initialization detection of message-expiration violations, initial security check, loop detection and criticality check.

14.4.2.1 *Arguments*

A report.

14.4.2.2 *Results*

The report with initialized **trace-information** for this MTA.

14.4.2.3 *Errors*

None. The report is discarded if an error is detected.

14.4.2.4 *Procedure description*

1) If the report has crossed a domain boundary, a **trace-information-element** for this domain is added with current time as the **arrival-time** and **relay** as **action**. An **internal-trace-information-element** is also added whether or not the report has crossed a domain boundary.
2) If required by the security-policy in force and/or if the **report-origin-authentication-check** is incorrect, the report is discarded and processing terminates.
3) If any of the extension fields is marked critical for transfer but is not semantically understood by the MTA, the report is discarded. The procedure then terminates.
4) Loop detection is performed. The loop detection algorithm is beyond the scope of this Recommendation. However, an example of a combined routing and loop detection algorithm is given in § 14.3.11. If a loop is detected, the report is discarded and the procedure terminates.

14.4.3 *Report-generation procedure*

This procedure generates a report describing the success and/or failure of operations attempted by this MTA.

14.4.3.1 *Arguments*

A message or probe. For each recipient with **responsibility** true, a per-recipient instruction is included indicating the success or problem to be reported.

14.4.3.2 *Results*

A report describing the successes or failures to be reported.

14.4.3.3 *Errors*

None.

14.4.3.4 *Procedure description*

If the subject's **originating-MTA-report-request** field so indicates, the report is constructed with arguments as described in Table 31/X.411, and further amplified by the following:

The delivery arguments (**message-delivery-time**, **type-of-MTS-user**) or Non-delivery arguments (**non-delivery-reason-code**, **non-delivery-diagnostic-code**) for each recipient are taken from the per-recipient instructions that accompanied the subject message. **Message-delivery-time** is taken from the message or probe trace information in case of a delivery report. If failure is reported for a DL recipient, then the **type-of-MTS-user** is set to **DL**. The **report-destination-name** is the last element from **DL-expansion-history**, if that element exists. For messages with no **DL-expansion-history** and for all probes, the **report-destination-name** is the subject's **originator-name**. The **originator-and-DL-expansion** will contain the **originator-name** and the subject's **Message-submission-time** followed by the content of **DL-expansion-history**.

Note – Reporting-DL-name is not generated under any of these conditions.

In the case where the instructions reflect multiple failures, the report should reflect the original problem rather than the failure of subsequent recovery actions.

Note – That the MTA nominates **critically** values for fields copied from the subject. These new values reflect criticality with regard to the report, not the subject. The MTA will not copy into the report any critical functions which it does not support.

14.4.4 *Report-routing procedure*

This procedure determines the routing action, if any, to be taken on a report. Report-routing reflects special conditions that require a routing procedure different from that applicable to messages or probes:

1) A report has just one recipient - the originator of the message that forms the subject of the report, a DL expansion-point, or, if local policy allows, a DL owner.
2) Insurmountable failures encountered in routing a report result in the discarding of the report. No attempt is made to generate a further report on the difficulty encountered.

The processing actions necessitated by these conditions are described in the following clauses. It should be noted that the routing of reports is subject to the security-policy.

14.4.4.1 *Arguments*

One of the following:

1) a report transferred to this MTA from another MTA and successfully processed by the report-front- end procedure;
2) a report created by the Report-generation procedure internal to this MTA;
3) a report received back from the Report-out procedure together with a description of the transfer failure encountered.

14.4.4.2 *Results*

One of the following:

1) the report, together with relaying instructions to the next hop MTA;
2) the report, together with an indication of the locally supported MTS-user who is to receive Report-delivery.

14.4.4.3 *Errors*

None. If no local recipient or next hop can be determined, the report is discarded.

14.4.4.4 *Procedure description*

1) Reports relayed to this MTA or generated locally receive normal routing attention as follows:
   a) If the Report-destination is not local to this MTA then relaying is required. Report-routing attempts to determine the next hop address. In this determination the **message-security-label** of the report is checked against the **security-context** to ensure no violation of the security-policy occurs. If successful, then the report, together with this information is returned as the procedure's result. The procedure then terminates. The report is subsequently passed to the Report-out procedure.

      If the next hop address cannot be determined, then the report is discarded and the procedure terminates without returning a result.
   b) If the Report-destination is an MTS-user local to this MTA, and the **originator-report-request** field indicates, then Report-delivery is required (subject to the security-policy in force). Report-routing attempts to determine the OR-address of the report destination. If successful, then the report, together with this information is returned as the procedure's result. The procedure then terminates. The report is subsequently passed to the Report-delivery procedures.

      If the report was not requested or the report destination address cannot be determined, the report is discarded and the procedure terminates without returning a result.
   c) If the **report-destination-name** is of a DL local to this MTA, then this report is in process of routing back along a path of successive DL expansion-points. In the **other-actions** field of the current **trace-information-element**, the value **dl-expansion** is set to true.

      Any processing based on local DL policy would occur here; e.g. a copy of the report can be constructed and sent to the DL owner. In this case the **report-destination-name** will be that of the DL owner and the **reporting-DL-name** will be constructed to contain the subject DL name. This copy of the report shall not contain the **returned-content**. In addition, suppression of reports can be done here.

      Note – The possibility that a DL owner is itself a DL is for further study.

      If the report is not to be suppressed, the MTA then replaces the **OR-name** currently in the **report-destination-name** field by the OR-name immediately preceding that one in the **originator-and-DL-expansion-history** field. Thus the report acquires, as a new destination, the next entry back along the chain of entries in the **originator-and-DL-expansion-history** field:

      report-destination-name:

      Copy previous DL **OR-name** from originator-and-**DL-expansion-history**.

      reporting-DL-name:

      Generated only in case of reports to DL owner.

      In order to route the report to this new destination, the Report-routing procedure now calls itself recursively. The result returned, if any, from this recursive call is returned, and the procedure terminates.

2) A report received back from the Report-out procedure has encountered a transfer failure in the process of relaying to another MTA. The Report-routing procedure attempts to reroute such a report, i.e. compute an alternative next hop address (subject to the security-policy in force). If an alternative next hop address is found then the report, together with this information and suitably modified trace information is returned as the procedure's result. The procedure then terminates. The report is subsequently passed to the report-out procedures.

   If an alternative next hop address cannot be determined, then the report is discarded and the procedure terminates without returning a result.

14.5 *MTS-bind and MTS-unbind*

14.5.1 *MTS-user initiated MTS-bind procedure*

This paragraph describes the behaviour of the MTA when an MTS-bind is invoked by an MTS-user.

14.5.1.1 *Arguments*

The MTS-bind arguments are defined in § 8.1.1.1.1.

14.5.1.2 *Results*

The MTS-bind results are defined in § 8.1.1.1.2.

14.5.1.3 *Errors*

The bind-errors are defined in § 8.1.2.

14.5.1.4 *Procedure description*

1) If the MTAs resources cannot currently support the establishment of a new association, the procedure returns a Busy bind-error and terminates.
2) Otherwise, if authentication is required by the security-policy, the MTA attempts to both authenticate the MTS-user via the **initiator-credentials** supplied and check the acceptability of the **security-context**. If the **initiator-credentials** cannot be authenticated, the procedure returns an authentication-error and terminates. If the **security-context** is not acceptable, the procedure returns an unacceptable-security-context bind-error and terminates.
3) If authentication is successful and the **security-context** is acceptable then the MTA accepts the requested association. The procedure returns the **MTA-name** and **responder-credentials**. Messages-waiting is also returned if the MTS-user subscribes to the Hold for Delivery element-of-service. The procedure then terminates.
4) If authentication is not required, Messages-waiting is returned if the **MTS-user** subscribes to the Hold for Delivery element-of-service and the procedure terminates.

14.5.2 *MTS-user initiated MTS-unbind procedure*

This paragraph describes the behaviour of the MTA when an MTS-unbind is invoked by an MTS-user in order to release an existing association established by the MTS-user.

14.5.2.1 *Arguments*

None.

14.5.2.2 *Results*

The MTS-unbind procedure returns an empty result as an indication of release of the association.

14.5.2.3 *Errors*

None.

14.5.2.4 *Procedure description*

The procedure releases the association, returns an empty result, and terminates.

14.5.3 *MTA initiaed MTS-bind procedure*

This paragraph describes the steps taken by an MTA when tasked to establish an association with an MTS-user.

14.5.3.1 *Arguments*

The MTS-bind arguments are defined in § 8.1.1.1.1.

14.5.3.2 *Results*

An internal identifier for the association established.

14.5.3.3 *Errors*

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

14.5.3.4 *Procedure description*

1) The procedure establishes values for the arguments defined in § 8.1.1.1.1. Messages-waiting may be supplied if the MTS-user subscribes to the hold for delivery element-of-service. Values for **initiator-name**, **security-context**, and **initiator-credentials** are taken from internal information.
2) The procedure determines the **user-address** of the MTS-user and attempts to establish an association with the arguments of § 8.1.1.1.1. If unsuccessful a failure indication is returned and the procedure terminates.
3) If successful, the results returned from the MTS-user (defined in § 8.1.1.1.2) are examined. The **responder-name** is checked for correctness and an attempt is made to authenticate the MTS-user via the **responder-credentials** returned. If either check fails, the procedure closes the connection, returns a failure indication, and terminates.
4) If both checks are successful the procedure returns the association identifier and terminates.

14.5.4 *MTA initiated MTS-unbind procedure*

This procedure is called to release an association with an MTS-user.

14.5.4.1 *Arguments*

This internal identifier for the association to be released.

14.5.4.2 *Results*

The MTS-unbind procedure returns an empty result as an indication of release of the association.

14.5.4.3 *Errors*

None.

14.5.4.4 *Procedure description*

The procedure releases the association, returns an empty result, and terminates.

14.6 *Submission port*

14.6.1 *Message-submission procedure*

This paragraph describes the behaviour of the MTA when the Message-submission abstract-operation is invoked by the MTS-user on a submission port.

14.6.1.1 *Arguments*

The Message-submission arguments listed in Table 3/X.411 and described in paragraphs indicated in that table.

14.6.1.2 *Results*

1) The Message-submission results listed in Table 5/X.411 and described in paragraphs indicated in that table are passed back to the MTS-user.
2) The Deferred Delivery module is invoked and passed the submitted message.

14.6.1.3 *Errors*

See § 8.2.1.1.3 for description of the relevant abstract-errors.

14.6.1.4 *Procedure description*

1) Error Checking

   The message-submission procedure checks for error conditions. If any is found, the indicated abstract-error is returned. All further processing is terminated. Responsibility for the intended message is not accepted by the MTA.

   Errors of particular interest:

   a) Security errors. If the message-security-label is not compatible with the security-context or, if required, the message-origin-authentication-check is incorrect, a security-error is generated.

   b) Criticality errors. If any of the extension fields is marked **critical-for-submission**, but not semantically understood by the MTA, an unsupported-critical-function-error is returned.

   If no errors are encountered at this stage, processing continues at step 2. Additional errors may be encountered in these later processing stages, in which case the MTA takes action as described above.

2) Name Processing

   The following procedure applies to originator-name, recipient-name and originator-requested-alternate-recipient, unless otherwise noted.

   a) If the **OR-name** contains only a **directory-name**, the MTA attempts to obtain the **OR-address**.

      The MTA may use the **requested-delivery-method**, if present, as an indication of which form of **OR-address** the **directory-name** should be mapped to. If a form of **OR-address** appropriate to the **requested-delivery-method**, cannot be found, the recipient-improperly-specified abstract-error is returned by the MTA.

   b) If the **OR-name** contains both the **directory-name** and the **OR-address**, their association need not be validated. If the **OR-address** is later found to be invalid, the MTA proceeds as if the **OR-address** was not supplied in the **OR-name**. The procedure described in (a) above is used to obtain the **OR-address**, which, if valid, replaces the supplied **OR-address** in the **OR-name**.

      If the obtained **OR-address** is invalid, an abstract-error is returned as described in (a) above.

   c) If a **recipient-name** contains an **OR-address** of a form not appropriate to the **requested-delivery-method**, if present, the **recipient-improperly-specified** abstract-error is returned by the MTA.

   d) The validation of the **OR-address**, whether passed in the Message-submission argument or obtained by resolving the **directory-name**, has two steps. The first step validates that the purported **OR-address** has the combination of attributes needed for a valid **OR-address** (see § 8.5.5). The second step, which applies only to the **originator-name**, validates that the **OR-address** is, in fact, the **OR-address** of the MTS-user submitting the message.

3) Transfer or Responsibility, Return of Results

   If no errors are detected in the above processing, the MTA accepts responsibility for the message and so signifies by returning the Message-submission results to the MTS-user. The Message-submission results are described in § 8.2.1.1.2. The **message-submission-identifier** and **message-submission-time** arguments are constructed as appropriate by the MTA. The **content-identifier** is identical to the corresponding Message-submission argument. If requested by the originator, the originating-MTA generates the **proof-of-submission** using the algorithm identified by the **proof-of-submission-algorithm-identifier** and the arguments defined in § 8.2.1.1.2.4. In addition the **originating-MTA-certificate** is returned.

4) Message Construction

   A Message is constructed from the Message-submission arguments, as possibly modified in the above processing steps, plus additional arguments supplied by the MTA, as specified in § 12.2.1.1.

   When complete, the Message-submission procedure terminates and the message is passed to the Deferred Delivery module for further processing.

14.6.2 *Probe-submission procedure*

This paragraph describes the behaviour of the MTA when the Probe-submission abstract-operation is invoked by the MTS-user on a submission-part.

14.6.2.1 *Arguments*

The Probe-submission arguments listed in Table 7/X.411 and described in paragraphs indicated in that table.

14.6.2.2 *Results*

1) The Probe-submission results listed in Table 8/X.411 and described in paragraphs indicated in that table are passed back to the MTS-user.

2) The Main module is invoked and passed the submitted probe.

14.6.2.3 *Errors*

See § 8.2.1.2.3 for descriptions of the relevant abstract-errors.

14.6.2.4 *Procedure description*

1) Error Checking

The Probe-submission procedure checks for error conditions. If any is found, the indicated abstract-error is returned. Responsibility for the intended probe is not accepted by the MTA.

Errors of particular interest:

a) Security errors. If the **message-security-label** is not compatible with the **security-context**, or if the **probe-origin-authentication-check** is incorrect, a security-error is generated.

b) Criticality errors. If any of the extension-fields is **critical-for-submission**, but not semantically understood by the MTA, an unsupported-critical-function-error is returned.

If no errors are encountered at this stage, processing continues at step 2. Additional errors may be encountered in these later processing stages, in which case the MTA takes action as described above.

2) Name Processing

The following procedure applies to originator-name, recipient-name and originator-requested-alternate-recipient, unless otherwise noted.

a) If the **OR-name** contains only a **directory-name**, the MTA attempts to obtain the **OR-address**.

In the case of **recipient-name**, the MTA may use the **requested-delivery-method**, if present, to indicate which form of **OR-address** the **directory-name** should be mapped to. If a form of **OR-address** appropriate to the **requested-delivery-method** cannot be found, the recipient-improperly-specified abstract-error is returned to the MTA.

b) If the **OR-name** contains both the **directory-name** and the **OR-address**, their association need not be validated. If the **OR-address** is later found to be invalid, the MTA proceeds as if the **OR-address** was not supplied in the **OR-name**. The procedure described in a) above is used to obtain the **OR-address**, which, if valid, replaces the supplied **OR-address** in the **OR-name**.

If the obtained **OR-address** is invalid, an abstract-error is returned as described in b) above.

c) If a **recipient-name** contains an **OR-address** of a form not appropriate to the **requested-delivery-method**, if present, the recipient-improperly-specified abstract-error is returned by the MTA.

d) The validation of the **OR-address**, whether passed in the Probe-submission argument or obtained by resolving the **directory-name**, has two steps. The first step validates that the purported **OR-address** has the combination of attributes needed for a valid **OR-address** (see § 8.5.5). The second step, which applies only to the **originator-name**, validates that the **OR-address** is, in fact, the **OR-address** of the MTS-user submitting the message.

3) Transfer of Responsibility, Return of Results

If no errors are detected in the above steps, the MTA accepts responsibility for the probe and so signifies by returning the Probe-submission results to the MTS-user. The Probe-submission results are described in § 8.2.1.2.2. The **probe-submission-identifier** and **probe-submission-time** arguments are constructed as appropriate by the MTA. The **content-identifier** is identical to the corresponding Probe-submission argument.

4) Probe Construction

A Probe is constructed from the Probe-submission arguments, as possibly modified in the above processing steps, plus additional arguments supplied by the MTA.

When complete, the Probe-submission procedure terminates and the probe is passed to the main module for further processing.

14.6.3 *Cancel-deferred-delivery procedure*

This paragraph describes the behaviour of the MTA when the Cancel-deferred-delivery abstract-operation is invoked by the MTA-user on a submission-port in order to cancel the deferred delivery message previously submitted to the MTA.

14.6.3.1 *Arguments*

The Cancel-deferred-delivery arguments listed in Table 10/X.411 and described in paragraphs indicated in that table.

14.6.3.2 *Results*

An empty result is passed back to the MTS-user as an indication of successful cancellation.

14.6.3.3 *Errors*

See § 8.2.1.3.3 for descriptions of the relevant abstract-errors.

14.6.3.4 *Procedure description*

1) If a **proof-of-submission** has already been provided, the Too-late-to-cancel abstract-error is returned by the MTA. The deferred delivery of the message is not cancelled.
2) If the value of the **message-submission-identifier** argument is recognized by the MTA as being valid and associated with a message being held by the MTA for deferred-delivery, the MTA discards this message as being cancelled, and assumes no further responsibility for it.
3) If the value of the **message-submission-identifier** argument is recognized by the MTA as being valid but refers to a message already delivered or transferred to another MTA, the Too-late-to-cancel abstract-error is invoked by the MTA. The deferred delivery of the message is not cancelled.
4) If the value of the **message-submission-identifier** argument is not recognized as being valid (either because the MTA never assigned such a value or because the MTA no longer holds the historical record of a deferred delivery message that has been transferred or delivered), then the Message-submission-identifier-invalid or Too-late-to-cancel abstract-error is returned by the MTA, the choice of which being a local matter.

14.6.4 *Submission-control procedure*

This paragraph describes the behaviour of the MTA when invoking the Submission-control abstract-operation on a submission-port in order to temporarily limit the submission-port abstract-operations that the MTS-user can invoke. These controls remain in force for the duration of the current association unless overridden by a subsequent Submission-control abstract-operation.

Note – The use of Submission-control shall be subject to the security-policy in force. The **permissible-security-context** Submission-control argument limits the **security-context** established during the MTS-bind.

14.6.4.1 *Arguments*

The Submission-control arguments listed in Table 12/X.411 and described in paragraphs indicated in that table.

14.6.4.2 *Results*

The Submission-control results listed in Table 13/X.411 and described in paragraphs indicated in that table are passed back to the MTA by the MTS-user.

14.6.4.3 *Errors*

A Security-error can be passed back by the MTS-user. See § 8.2.1.4.3 for a description of this abstract-error.

14.6.4.4 *Procedure description*

The circumstances causing an MTA to invoke the Submission-control abstract-operation are a local matter, as are the actions taken during and subsequent to its completion.

14.7 *Delivery port*

14.7.1 *Message-delivery procedure*

This paragraph describes the steps taken by an MTA when tasked to deliver a message to one or more MTS-users.

Most provisions of this clause also apply to the case where the MTA has received a probe with one or more local recipients. Unless noted otherwise, all procedure steps save physical delivery apply to the handling of probes.

Note – The generation of reports shall be subject to the security-policy.

14.7.1.1 *Arguments*

1) A message from the main module with per-recipient instructions to deliver to one or more local MTS-users.
2) The message-delivery arguments listed in Table 15/X.411 and described in paragraphs indicated in that table are passed to the recipient MTS-user.

14.7.1.2 *Results*

1) An empty or, if requested, a **proof-of-delivery** and optional **recipient-certificate** result passed back from the MTS-user as an indication of successful delivery with no reporting requirements.
2) The Main module is invoked and passed the message with per-recipient instructions describing any delivery problems encountered and/or indicating successful deliveries to be reported on.

14.7.1.3 *Errors*

Message-delivery abstract-errors that can be returned from the MTS-user to the MTA are described in § 8.3.1.1.3. These error conditions are reported to the Main module in the results described above.

14.7.1.4 *Procedure description*

1) If the message expiration is reached, a report instruction is generated for each local recipient. The values of **non-delivery-reason-code** and **non-delivery-diagnostic-code** are **unable-to-transfer** and **maximum-time-expired**, respectively. The procedure then terminates.
2) If any of the per-message **extension-fields** is set to **critical-for-delivery** but not semantically understood by the MTA, a report instruction for each local recipient is generated. The values of **non-delivery-reason-code** and**non-delivery-diagnostic-code** are set to **unable-to-transfer** and **unsupported-critical-function** respectively.
3) Otherwise, values are established for those arguments to the Message-delivery abstract-operation that apply to all recipients (arguments to message-delivery are described in § 8.3.1.1.1).
4) Steps 4-15 are executed for each recipient with **responsibility** true. The procedure then terminates.
5) To ensure the security-policy is not violated during delivery, the **message-security-label** is checked against the **security-context**. If delivery is barred by the security-policy then, subject to the security policy, a report instruction for this recipient is generated. The values of **non-delivery-reason-code** and **non-delivery-diagnostic-code** are **unable-to-transfer** and **secure-messaging-error**, respectively.
6) If delivery barred by restrictions imposed in a previously invoked Register or Delivery-control-abstract-operation, then, subject to the security-policy in force, the MTA will hold the message pending the lifting of the applicable restriction(s).
7) If the maximum holding time for a held message (the value of this maximum time being a local matter) expires with the applicable restrictions still in effect, then a report instruction is generated for this recipient. The values of **non-delivery-reason-code** and **non-delivery-diagnostic-code** are **unable-to-transfer** and **recipient-unavailable**, respectively. Processing then terminates for this recipient.

   Note – The processing steps (5 and 6 above) associated with control restrictions do not apply in the case of Probe.
8) If restricted delivery is enforced and the recipient falls in the category of unauthorized senders, then a report instruction is generated for this recipient. The value of **non-delivery-reason-code** is set to **restricted-delivery**. Processing then terminates for this recipient.
9) The MTA establishes those arguments for the Message-delivery abstract-operation that apply only to the individual recipient: **message-delivery-identifier** and **message-delivery-time** are given values as described in §§ 8.3.1.1.1.1 and 8.3.1.1.1.2. All other arguments are taken directly from corresponding fields of the message to be delivered. With the exceptions noted below, all arguments shown in Table 11/X.411 are included in each invocation of Message-delivery.
10) If **disclosure-of-recipients** has the value **disclosure-of recipients-allowed**, the MTA includes all recipients, which were specified by the originator, save the current one, in the **other-recipient-name argument**.

Note that if the recipient is a member of a distribution list, other members of this distribution list must not be included in the **other-recipient-name** argument. The recipient is a member of a distribution list if the **DL-expansion-history** field is non-empty.

11) If any of the per-recipient **extension-fields** is set to **critical-for-delivery**, but not semantically understood by the MTA, a report instruction for this recipient is generated. The values of the **non-delivery-reason-code** and **non-delivery-diagnostic-code** are set to **unable-to-transfer** and **unsupported-critical-function** respectively.

12) In the case of delivery to a Physical Delivery Access Unit, the Physical Delivery Arguments are included in the Message-delivery. These arguments are described in §§ 8.2.1.1.1.14-8.2.1.1.1.23.

13) Once all conditions have been met for succesful delivery, the MTA will physically deliver the message. The accomplishment of delivery to a collocated recipient MTS-user is a local matter. In the case of a remotely located recipient MTS-user, the MTA establishes an association with that MTS-user (or uses an existing one) and invokes the Message-delivery abstract-operation across that association. With successful delivery, either remote or local, responsibility for the message passes from the MTA to the recipient MTS-user.

14) Upon a successful delivery, if the **originating-MTA-delivery-report-request** has the value of **report** or **audited-report**, then a report instruction is generated noting the successful delivery. Processing then terminates for this recipient.

15) In the case of a remotely located recipient MTS-user, if an association neither exists nor can be established initially, or there is a transfer failure across an association, the MTA can repeat the attempt at association establishment and/or transfer, the maximum number and/or time duration of repeats being a local matter. If, after repeated attempts transfer has not been accomplished, the message is deemed undeliverable and, subject to the security-policy in force, a report instruction is generated. The values of **non-delivery-reason-code** and **non-delivery-diagnostic-code** are **transfer-failure** and **recipient-unavailable**, respectively. Processing then terminates for this **recipient**.

    Note – The processing steps associated with physical transfer of a message to the recipient MTS-user do not apply in the case of Probe.

16) Return of results and errors by the MTS-user.

    If the Message-delivery abstract-operation is successful, then the MTS-user returns, as an indication of success either an empty result or, if requested, a **proof-of-delivery** and optional recipient-certificate.

    If the Message-delivery abstract-operation violates one or more controls imposed by a previous Delivery-control or Register abstract-operation, then the MTS-user returns a Delivery-control-violated error. If the **security-context** dictates that the MTS-user cannot support the requested abstract-operation because it would violate the security-policy, then the MTS-user returns a Security-error. In this event the Message-delivery invocation has failed and the MTA retains responsibility for the message with respect to this recipient. The message is held for subsequent retry or is passed to the Main module for report generation. Processing then terminates for this recipient.

14.7.2 *Probe-delivery-test procedure*

This paragraph describes the steps taken by an MTA when tasked to test the deliverability of probe.

Note – The use of Reports shall be subject to the security-policy.

14.7.2.1 *Arguments*

1) A probe from the internal procedure with per-recipient instructions to Probe-delivery-test to one or more local MTS-users.

14.7.2.2 *Results*

The Main module is invoked and passed the probe with per-recipient instructions describing whether or not the hypothetical delivery would have occurred and if not why not.

14.7.2.3 *Errors*

None.

14.7.2.4 *Procedure description*

The logic for Message-delivery is described in § 14.7.1. All steps in the paragraph except those specifically noted as inapplicable to Probe are executed.

14.7.3 *Report-delivery procedure*

This paragraph describes the steps taken by an MTA when tasked to deliver a report to an MTS-user. Report-delivery is called for when an MTA receives a report, from Report-in or upon generation within this MTA, whose **originator-name** field specifies an MTS-user served by this MTA.

14.7.3.1 *Arguments*

1) A report from the Report module with per-recipient instructions to deliver to a local recipient.
2) The Report-delivery arguments listed in Table 18/X.411 and described in paragraphs indicated in that table are passed to the recipient MTS-user.

14.7.3.2 *Results*

An empty result passed back from the MTS-user as an indication of successful delivery.

14.7.3.3 *Errors*

Report-delivery errors that can be returned from the MTS-user to the MTA are described in § 8.3.1.2.3.

14.7.3.4 *Procedure description*

1) To ensure the security-policy is not voilated during Report-delivery the **message-security-label** is checked against the security-context. If Report-delivery is barred by the security-policy, then the report is descarded.
2) If Report delivery is barred by restrictions imposed in a previously invoked Register or Delivery-control abstract-operation, then, subject to the security-policy in force, the MTA will hold the report pending the lifting of the applicable restriction(s). Restrictions are established by arguments of the Delivery-control or Register abstract-operation as described in § 8.3.1.3.1.

   If the maximum holding time for a held report (the value of this maximum time being a local matter) expires with the applicable restrictions still in effect, then the report is discarded.
3) Arguments for the Report-delivery abstract-operation are taken from corresponding fields of the report.
4) If any of the per-message or per-recipient **extension-fields** are set to **critical-for-delivery**, but not semantically understood by the MTA, the report is discarded.
5) The accomplishment of Report-delivery to a collocated MTS-user is a local matter. In the case of a remotely located MTS-user, the MTA establishes an association with that MTS-user (or uses an existing one) and invokes the Report-delivery abstract-operation across that association. With successful Report-delivery, either remote or local, responsibility for the report passes from the MTA to the MTS-user.
6) In the case of a remotely located MTS-user, if an association cannot be established initially, the MTA can repeat the attempt, the maximum number and/or time duration of repeats being a local matter. If, after repeated attempts no association has been established, the report is deemed undeliverable and is discarded.
7) Return of Results and Errors by the MTS-user.

   If the Report-delivery abstract-operation is successful, then the MTS-user returns an empty result as an indication of success.

   If the Report-delivery abstract-operation violates one or more controls imposed by a previous Delivery-control or Register abstract-operation, then the MTS-user returns a Delivery-control-violated error. In this event the Report-delivery invocation has failed and the MTA retains responsibility for the report.

14.7.4 *Delivery-control procedure*

This paragraph describes the behavior of the MTA when the Delivery-control abstract-operation is invoked by an MTS-user served by this MTA. Delivery-control imposes and lifts restrictions on the Message-delivery and Report-delivery abstract-operations. These controls remain in force for the duration of the current association unless overridden by a subsequent Delivery-control. Delivery-controls temporarily limit the **security-context** but cannot cause a violation of the security-policy.

These controls do not apply to the processing of probes by the MTA.

14.7.4.1 *Arguments*

The Delivery-control arguments listed in Table 20/X.411 and described in § 8.3.1.3.1.

14.7.4.2 *Results*

1) The Delivery-control results listed in Table 21/X.411 and described in § 8.3.1.3.2 are passed back to the MTS-user by the MTA.
2) Various control parameters of the MTS-user held by this MTA are replaced by values carried in the Delivery-control arguments.

14.7.4.3 *Errors*

See § 8.3.1.3.3 for a description of the relevant abstract-errors.

14.7.4.4 *Procedure description*

1) If the value of the **restrict** argument is **remove**, then all controls established by any previous Delivery-control are removed; the abstract-operation is complete, and the Result is returned to the MTS-user.
2) If the value of the restrict argument is **update**, and no other arguments are present, the request is considered to be valid and the Result returned to the MTS-user.

   In such cases all currently in force control values remain unchanged.
3) If the value of the **restrict** argument is **update**, and other arguments are present, those arguments are checked for compatibility with long term conditions specified by the most recent invocation of the Register abstract-operation on the administration-port (see § 14.4.1). If no incompatibility is detected, and the update is permitted within the security-policy, the indicated updates are carried out, the abstract-operation is complete, annd the Result is returned to the MTS-user.
4) If any of the following incompatibilities is detected with long term conditions, a Control-violates-registration abstract-error is returned by the MTA;
   a) The **permissible-encoded-information-types** has a type not specified among those allowed long term.
   b) The **permissible-content-types** has a content not specified among those allowed long term.
   c) The **permissible-maximum-content-length** exceeds the length allowed long term.
   d) The **permissible-security-context** is violated.

   In any of the error cases, the Delivery-control is discarded and not carried out.

14.8 *Administration port*

14.8.1 *Register procedure*

This paragraph describes the behaviour of the MTA when the Register abstract-operation is invoked by an MTS-user served by this MTA.

14.8.1.1 *Arguments*

The Register arguments listed in Table 23/X.411 and described in paragraphs indicated in that table.

14.8.1.2 *Results*

1) The Register procedure returns an empty result to the MTS-user as an indication of success.
2) Various parameters of the MTS-user held by this MTA are replaced by values carried in the Register arguments.

14.8.1.3 *Errors*

A Register-rejected error returned to the MTS-user as described in § 8.4.1.1.3.

14.8.1.4 *Procedure description*

1) The Register arguments are checked for correct specification. If any is incorrectly specified, the Register procedure returns a Register-rejected error and terminates.
2) If the Register arguments are correctly specified, the values of MTS-user parameters are replaced by those of the Register arguments, and the procedure terminates.

14.8.2 *MTS-user initiated change-credentials procedure*

This paragraph describes the behavior of the MTA when a change-credentials abstract-operation is invoked by the MTS-user.

Note – All changes of credentials shall be subject to the security-policy in force.

14.8.2.1 *Arguments*

The Change-credentials arguments listed in Table 25/X.411 and described in § 8.4.1.2.1.

14.8.2.2 *Results*

1) The Change-credentials procedure returns an empty result to the MTS-user as an indication of success.
2) The MTS-user's credentials held by this MTA are changed in accordance with the new-credentials argument.

14.8.2.3 *Errors*

A New-credentials-unacceptable or Old-credentials-incorrectly-specified abstract-error, as described in § 8.4.1.2.3 and listed in Table 26/X.411.

14.8.2.4 *Procedure description*

Note – All changes of credentials shall be subject to the security-policy in force.

1) If the value of the **old-credentials** argument is not the same as the credentials held by the MTA for the MTS-user invoking the abstract-operation, an Old-credentials-incorrectly-specified error is returned to the MTS-user and the Change-credentials procedure terminates.
2) Otherwise, the **new-credentials** argument is checked for validity. If found invalid (a local matter dictated by the security-policy) a New-credentials-unacceptable error is returned to the MTS-user and the Change-credentials procedure terminates.
3) Otherwise, the MTS-user's credentials held by this MTA are changed to the value of the **new-credentials** argument, an empty result is returned to the MTS-user as an indication of success, and the Change-credentials procedure terminates.

14.8.3 *MTA initiated change-credentials procedure*

This paragraph describes the behaviour of an MTA when changing its credentials held by a locally supported MTS-user.

Note – All changes of credentials shall be subject to the security-policy in force.

14.8.3.1 *Arguments*

The Change-credentials arguments listed in Table 25/X.411 and described in § 8.4.1.2.1.

14.8.3.2 *Results*

The MTS-user returns an empty result to the Change-credentials procedure as an indication of success.

14.8.3.3 *Errors*

The MTS-user can return a New-credentials-unacceptable or Old-credentials-incorrectly-specified error, as described in § 8.4.1.2.3 and listed in Table 26/X.411.

14.8.3.4 *Procedure description*

Note – All changes of credentials shall be subject to the security-policy in force.

1) The procedure invokes the Change-credentials abstract-operation to change the MTA's credentials held by a locally supported MTS-user. The conditions causing an MTA to change its credentials are a local matter.
2) If either the New-credentials-unacceptable or Old-credentials-incorrectly-specified error is received back from the MTS-user, then the MTA must assume its credentials have not been changed. Further action can be undertaken as a local matter, after which the procedure terminates.
3) If an emply result is received back from the MTS-user, the MTA may assume the procedure has been successful and its credentials changed. The procedure terminates.

14.9 *MTA-bind and MTA-unbind*

14.9.1 *MTA-bind-in procedure*

This paragraph describes the behaviour of the MTA when an MTA-bind is invoked by another MTA.

14.9.1.1 *Arguments*

The MTA-bind results are defined in § 12.1.1.1.1 and listed in Table 27/X.411.

14.9.1.2 *Results*

The MTA-bind results are defined in § 12.1.1.1.2 and listed in Table 28/X.411.

14.9.1.3 *Errors*

The bind-errors are defined in § 12.1.2.

14.9.1.4 *Procedure description*

1) If the MTA's resources cannot currently support the establishment of a new association, the procedure returns a Busy bind-error and terminates.
2) Otherwise, if authentication is required by the security-policy, the MTA attempts to both authenticate the calling MTA via the **initiator-credentials** supplied and check the acceptability of the **security-context**. If the **initiator-credentials** cannot be authenticated, the procedure returns an authentication-error and terminates. If the **security-context** is not acceptable, the procedure returns an unacceptable-security-context error and terminates.
3) If authentication is successful and the **security-context** is acceptable, then the MTA establishes the requested association. The procedure returns the **MTA-name** and **responder-credentials**. The procedure then terminates.
4) If authentication is not required, there are no results to return and the procedure terminates.

14.9.2 *MTA-unbind-in procedure*

This paragraph describes the behaviour of the MTA when an MTA-unbind is invoked by another MTA in order to release an existing association.

14.9.2.1 *Arguments*

None.

14.9.2.2 *Results*

The MTA-unbind-in procedure returns an empty result as an indication of release of the association.

14.9.2.3 *Errors*

None.

14.9.2.4 *Procedure description*

The procedure releases the association, returns an empty result, and terminates.

14.9.3 *MTA-bind-out procedure*

This paragraph describes the steps taken by an MTA when tasked to establish an association with another MTA.

14.9.3.1 *Arguments*

1) The **MTA-name** of the MTA with which the association is to be established.
2) The **security-context** for the association.

14.9.3.2 *Results*

An internal identifier for the association established.

14.9.3.3 *Errors*

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

14.9.3.4 *Procedure description*

1) The procedure establishes values for the **arguments** defined in § 12.1.1.1.1. Values for **initiator-name**, **security-context**, and **initiator-credentials** are taken from internal information.
2) The procedure determines the address of the MTA and attempts to establish an association with the arguments of § 12.1.1.1.1. If unsuccessful a failure indication is returned and the procedure terminates.

3) If successful, the results returned from the called MTA (defined in § 12.1.1.1.2) are examined. The **responder-name** is checked for correctness, an attempt is made to authenticate the MTA via the **responder-credentials** returned. If any of the checks fail, the procedure returns a failure indication to the caller, terminates the association, and terminates.

4) If all checks are successful the procedure returns the association identifier and terminates.

14.9.4 *MTA-unbind-out procedure*

This procedure is called to release an association with another MTA.

14.9.4.1 *Arguments*

The internal identifier for the association to be released.

14.9.4.2 *Results*

The MTA-unbind-out procedure returns an empty result as an indication of release of the association.

14.9.4.3 *Errors*

None.

14.9.4.4 *Procedure description*

The procedure releases the association, returns an empty result, and terminates.

14.10 *Transfer port*

Note – The actions taken on the transfer-port are subject to the security-policy in force.

14.10.1 *Message-in procedure*

This paragraph describes the behaviour of the MTA when a Message-transfer abstract-operation is invoked by another MTA on a transfer-port.

14.10.1.1 *Arguments*

The Message-transfer arguments listed in Table 29/X.411 and described in paragraphs indicated in that table.

14.10.1.2 *Results*

1) The Deferred Delivery module is invoked and passed the message transferred in.

14.10.1.3 *Errors*

None.

14.10.1.4 *Procedure description*

On receipt of a message through the occurrence of a Message-transfer abstract-operation (invoked from a neighbour MTA), the Message-in procedure is invoked. This procedure simply passes the message to the Deferred Delivery module to determine the actions to be taken by this MTA.

Responsibility for the message passes to the receiving-MTA with the successful transfer.

14.10.2 *Probe-in procedure*

This paragraph describes the behavior of the MTA when a Probe-transfer abstract-operation is invoked by another MTA on a transfer-port.

14.10.2.1 *Arguments*

The Probe-transfer arguments listed in Table 30/X.411 and described in paragraphs indicated in that table.

14.10.2.2 *Results*

1) The Report module is invoked and passed the report transferred in.

14.10.2.3 *Errors*

None.

14.10.2.4 *Procedure description*

On receipt of a probe through the occurrence of a Probe-transfert abstract-operation (invoked from a neighbour MTA), the Probe-in procedure is invoked. This procedure simply passes the probe to the Main module to determine the actions to be taken by this MTA.

Responsibility for the probe passes to the receiving-MTA with the successful transfer.

14.10.3 *Report-in procedure*

This paragraph describes the behavior of the MTA when it receives a Report on a transfer-port through thje occurrence of a Report-transfer abstract-operation invoked by another MTA, or when it receives an indication for the generation of a report from an access unit such as a PDAU.

14.10.3.1 *Arguments*

The Report arguments listed in Table 31/X.411 and described in paragraphs indicated in that table.

14.10.3.2 *Results*

1) The Report module is invoked and passed the report transferred in.

14.10.3.3 *Errors*

None.

14.10.3.4 *Procedure description*

On receipt of a report through the occurrence of a Report-transfer abstract-operation (invoked from a neighbour MTA), or on receipt of an indication for a report generation from an access unit such as a PDAU, the Report-in procedure is invoked. This procedure simply passes the report to the Report module to determine the actions to be taken by this MTA.

Responsibility for the report passes to the receiving-MTA with the successful transfer.

14.10.4 *Message-out procedure*

This paragraph describes the steps taken by an MTA when tasked to transfer a message to another MTA.

14.10.4.1 *Arguments*

A message from the internal procedure with routing instructions to transfer to another MTA. The fields of this message form the arguments of the Message-transfer abstract-operation as listed in Table 29/X.411.

14.10.4.2 *Results*

None.

14.10.4.3 *Errors*

In case of transfer failure the Main module is invoked and passed the message with a per-message instruction indicating the failure reason.

14.10.4.4 *Procedure description*

The message to be transferred provides the arguments for the Message-transfer abstract-operation. It should be noted that the message may reflect processing (e.g., content conversion, redirection, distribution list expansion) carried out in this or previous MTAs.

1) To ensure the security-policy is not violated during transfer, the **message-security-label** is checked against the **security-context**. If the transfer is barred by either the security-policy or temporary restrictions, then processing continues at step 3, below.
2) Otherwise, the MTA establishes an association with the receiving-MTA (or uses an existing one) and invokes the Message-transfer abstract-operation across that association. The completion of Message-out indicates that the transfer has successful and that the receiving-MTA now accepts responsibility for the message. The Message-out procedure now terminates.

   If an association neither exists nor can be established initially, or there is a transfer failure across an association, the MTA can repeat the attempt at association establishment and/or transfer, the maximum number and/or time duration of repeats being a local matter.

3) If, after repeated attempts transfer has not been accomplished, or a security violation has been detected in step 1, the message is deemed non transferable and is returned, with failure reason indicated, to the Main module for possible rerouting or redirection. Responsibility for the message remains with the sending MTA. The Message-out procedure now terminates.

14.10.5 *Probe-out procedure*

This paragraph describes the steps taken by an MTA when tasked to transfer a probe to another MTA.

14.10.5.1 *Arguments*

A probe from the internal procedure with routing instructions to transfer to another MTA. The fields of this probe form the arguments of the probe-transfer abstract-operation as listed in Table 30/X.411.

14.10.5.2 *Results*

None.

14.10.5.3 *Errors*

In case of transfer failure the Main module is invoked and passed the probe with a per-message instruction indicating the failure reason.

14.10.5.4 *Procedure description*

The probe to be transferred provides the arguments for the Probe-transfer abstract-operation. It should be noted that the probe may reflect processing (e.g., redirection) carried out in this or previous MTAs.

1) To ensure the security-policy is not violated during transfer, the **message-security-label** is checked against the **security-context**. If the transfer is barred by either the security-policy or temporary restrictions, then processing continues at step 3, below.

2) The MTA establishes an association with the receiving MTA (or uses an existing one) and invokes the Probe-transfer abstract-operation across that association. The completion of Probe-out indicates that the transfer has been successful and that the receiving-MTA now accepts responsibility for the probe. The Probe-out procedure now terminates.

   If an association neither exists nor can be established initially, or there is a transfer failure across an association, the MTA can repeat the attempt at association establishement and/or transfer, the maximum number and/or time duration of repeats being a local matter.

3) If, after repeated attempts transfer has not been accomplished, or a security violation has been detected in step 1 above, then the probe is deemed non transferrable and is returned, with failure reason indicated, to the Main module for possible rerouting or redirection. Responsibility for the probe remains with the sending MTA. The Probe-out procedure now terminates.

14.10.6 *Report-out procedure*

This paragraph describes the steps taken by an MTA when tasked to transfer a report to another MTA.

14.10.6.1 *Arguments*

A report from the internal procedure with routing instructions to transfer to another MTA. The fields of this report form the arguments of the Report-transfer abstract-operation as listed in Table 31/X.411.

14.10.6.2 *Results*

None.

14.10.6.3 *Errors*

The report, together with the reason for transfer failure, to be passed back to the Report module.

14.10.6.4 *Procedure description*

The report to be transferred provides the arguments for the Report-transfer abstract-operation. It should be noted that the report may reflect processing (e.g., redirection) carried out in this or previous MTAs.

1) To ensure the security-policy is not violated during transfer, the **message-security-label** is checked against the **security-context**. If the transfer is barred by either the security-policy or temporary restrictions, then processing continues at step 3, below.

2) The MTA establishes an association with the receiving MTA (or uses an existing one) and invokes the Report-transfer abstract-operation across that association. The completion of Report-out indicates that the transfer has been successful and the receiving-MTA now accepts responsibility for the report. The Report-out procedure now terminates.

   If an association neither exists nor can be established initially, or there is a transfer failure across an association, the MTA can repeat the attempt at association establishment and/or transfer, the maximum number and/or time duration of repeats being a local matter.

3) If, after repeat attempts transfer has not been accomplished, or a security violation has been detected in step 1 above, then the report is deemed non transferrable and is returned, with failure reason indicated, to the report module for possible rerouting. Responsibility for the report remains with the sending MTA. The Report-out procedure now terminates.

ANNEX A

(to Recommendation X.411)

Reference definition of MTS object identifiers

This Annex defines for reference purposes various object identifiers cited in the ASN.1 modules in the body of this Recommendation. The object identifiers are assigned in Figure A-1/X.411.

All object identifiers this Recommendation assigns are assigned in this annex. The annex is definitive for all but those ASN.1 modules and the Message Transfer System itself. The definitive assignments for the former occur in the modules themselves; other references to them appear in IMPORT clauses. The latter is fixed.

```
MTSObjectIdentifiers { joint-iso-ccitt mhs-motis(6) mts(3) modules(0) object-identifiers(0) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- Prologue
-- Exports everything


IMPORTS -- nothing --;


-- Message transfer system

id-mts OBJECT IDENTIFIER ::= { joint-iso-ccitt mhs-motis(6) mts(3) }      -- not definitive


-- Categories of object identifiers

id-mod OBJECT IDENTIFIER ::= { id-mts 0 }                                 -- modules
id-ot OBJECT IDENTIFIER ::= { id-mts 1 }                                  -- object types
id-pt OBJECT IDENTIFIER ::= { id-mts 2 }                                  -- port types
id-cont OBJECT IDENTIFIER ::= { id-mts 3 }                                -- content types
id-eit OBJECT IDENTIFIER ::= { id-mts 4 }                                 -- encoded information types
id-att OBJECT IDENTIFIER ::= { id-mts 5 }                                 -- attributes
id-tok OBJECT IDENTIFIER ::= { id-mts 6 }                                 -- token types
id-sa OBJECT IDENTIFIER ::= { id-mts 7 }                                  -- secure agent types


-- Modules

id-mod-object-identifiers OBJECT IDENTIFIER ::= { id-mod 0 }              -- not definitive
id-mod-mts-abstract-service OBJECT IDENTIFIER ::= { id-mod 1 }            -- not definitive
id-mod-mta-abstract-service OBJECT IDENTIFIER ::= { id-mod 2 }            -- not definitive
```

FIGURE A-1/X.411 (Part 1 of 3)

Abstract syntax definition of the MTS object identifiers

```
id-mod-upper-bounds OBJECT IDENTIFIER ::= { id-mod 3 }                -- not definitive


-- Object types

id-ot-mts OBJECT IDENTIFIER ::= { id-ot 0 }

id-ot-mts-user OBJECT IDENTIFIER ::= { id-ot 1 }

id-ot-mta OBJECT IDENTIFIER ::= { id-ot 2 }


-- Port types

id-pt-submission OBJECT IDENTIFIER ::= { id-pt 0 }

id-pt-delivery OBJECT IDENTIFIER ::= { id-pt 1 }

id-pt-administration OBJECT IDENTIFIER ::= { id-pt 2 }

id-pt-transfer OBJECT IDENTIFIER ::= { id-pt 3 }


-- Content types

id-cont-undefined OBJECT IDENTIFIER ::= { id-cont 0 }

id-cont-inner-envelope OBJECT IDENTIFIER ::= { id-cont 1 }


-- Encoded information types

id-eit-undefined OBJECT IDENTIFIER ::= { id-eit 0 }

id-eit-telex OBJECT IDENTIFIER ::= { id-eit 1 }

id-eit-ia5-text OBJECT IDENTIFIER ::= { id-eit 2 }

id-eit-g3-facsimile OBJECT IDENTIFIER ::= { id-eit 3 }

id-eit-g4-class-1 OBJECT IDENTIFIER ::= { id-eit 4 }

id-eit-teletex OBJECT IDENTIFIER ::= { id-eit 5 }

id-eit-videotex OBJECT IDENTIFIER ::= { id-eit 6 }
```

FIGURE A-1/X.411 (Part 2 of 3)

Abstract syntax definition of the MTS object identifiers

```
id-eit-voice OBJECT IDENTIFIER ::= { id-eit 7 }

id-eit-sfd OBJECT IDENTIFIER ::= { id-eit 8 }

id-eit-mixed-mode OBJECT IDENTIFIER ::= { id-eit 9 }


-- Attributes

id-att-physicalRendition-basic OBJECT IDENTIFIER ::= { id-att 0 }


-- Token types

id-tok-asymmetricToken OBJECT IDENTIFIER ::= { id-tok 0 }


-- Secure agent types

id-sa-ua OBJECT IDENTIFIER ::= { id-sa 0 }

id-sa-ms OBJECT IDENTIFIER ::= { id-sa 1 }


END  -- of MTSObjectIdentifiers
```

FIGURE A-1/X.411 (Part 3 of 3)

Abstract syntax definition of the MTS object identifiers

ANNEX B

(to Recommendation X.411)

Reference definition of MTS parameter upper bounds

This annex defines for reference purposes the upper bounds of various variable length data types whose abstract syntaxes are defined in the ASN.1 modules in the body of this Recommendation. The upper bounds are defined in Figure B-1/X.411.

```
MTSUpperBounds { joint-iso-ccitt mhs-motis(6) mts(3) modules(0) upper-bounds(3) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN


-- Prologue
-- Exports everything


IMPORTS -- nothing --;


-- Upper bounds

ub-integer-options INTEGER ::= 256

ub-queue-size INTEGER ::= 2147483647                     -- the largest integer in 32 bits

ub-content-length INTEGER ::= 2147483647                 -- the largest integer in 32 bits

ub-password-length INTEGER ::= 62

ub-bit-options INTEGER ::= 16

ub-content-types INTEGER ::= 1024

ub-tsap-id-length INTEGER ::= 16

ub-recipients INTEGER ::= 32767

ub-content-id-length INTEGER ::= 16

ub-x121-address-length INTEGER ::= 15

ub-mts-user-types INTEGER ::= 256

ub-reason-codes INTEGER ::= 32767

ub-diagnostic-codes INTEGER ::= 32767

ub-supplementary-info-length INTEGER ::= 256

ub-extension-types INTEGER ::= 256
```

FIGURE B-1/X.411 (Part 1 of 3)

Abstract syntax definition of MTS upper bounds

```
ub-recipient-number-for-advice-length INTEGER ::= 32

ub-content-correlator-length INTEGER ::= 512

ub-redirections INTEGER ::= 512

ub-dl-expansions INTEGER ::= 512

ub-built-in-content-type INTEGER ::= 32767

ub-local-id-length INTEGER ::= 32

ub-mta-name-length INTEGER ::= 32

ub-country-name-numeric-length INTEGER ::= 3

ub-country-name-alpha-length INTEGER ::= 2

ub-domain-name-length INTEGER ::= 16

ub-terminal-id-length INTEGER ::= 24

ub-organization-name-length INTEGER ::= 64

ub-numeric-user-id-length INTEGER ::= 32

ub-surname-length INTEGER ::= 40

ub-given-name-length INTEGER ::= 16

ub-initials-length INTEGER ::= 5

ub-generation-qualifier-length INTEGER ::= 3

ub-organizational-units INTEGER ::= 4

ub-organizational-unit-name-length INTEGER ::= 32

ub-domain-defined-attributes INTEGER ::= 4

ub-domain-defined-attribute-type-length INTEGER ::= 8

ub-domain-defined-attribute-value-length INTEGER ::= 128
```

FIGURE B-1/X.411 (Part 2 of 3)

Abstract syntax definition of MTS upper bounds

```
ub-extension-attributes INTEGER ::= 256

ub-common-name-length INTEGER ::= 64

ub-pds-name-length INTEGER ::= 16

ub-postal-code-length INTEGER ::= 16

ub-pds-parameter-length INTEGER ::= 30

ub-physical-address-lines INTEGER ::= 6

ub-unformatted-address-length INTEGER ::= 180

ub-e163-4-number-length INTEGER ::= 15

ub-e163-4-sub-address-length INTEGER ::= 40

ub-built-in-encoded-information-types INTEGER ::= 32

ub-teletex-private-use-length INTEGER ::= 128

ub-encoded-information-types INTEGER ::= 1024

ub-security-labels INTEGER ::= 256

ub-labels-and-redirections INTEGER ::= 256

ub-security-problems INTEGER ::= 256

ub-privacy-mark-length INTEGER ::= 128

ub-security-categories INTEGER ::= 64

ub-transfers INTEGER ::= 512

ub-bilateral-info INTEGER ::= 1024

ub-additional-info INTEGER ::= 1024


END  -- of MTSUpperBounds
```

FIGURE B-1/X.411 (Part 3 of 3)

Abstract syntax definition of MTS upper bounds

ANNEX C

(to Recommendation X.411)

Differences between ISO/IEC and CCITT versions

This annex identifies the technical differences between the ISO/IEC and CCITT versions of CCITT Recommendation X.411 and ISO/IEC 10021-4.

They are:

1) In CCITT Recommendation X.411, extension fields are identified by integers. ISO/IEC 10021-4 allows, in addition, the use of object identifiers for extensions within and/or between PRMDs.

2) In CCITT Recommendation X.411, size contraints are applied to a number of protocol fields (see Annex B). In ISO/IEC 10021-4, the actual values of the constraints are not an integral part of the Standard.

ITU-T RECOMMENDATIONS SERIES

Printed in Switzerland

Geneva, 2008