INTERNATIONAL STANDARD

ISO/IEC 10021-5

Third edition
1996-06-15

Information technology — Message Handling Systems (MHS): Message store: Abstract service definition

Technologies de l'information — Systèmes de messagerie (MHS): Dépôt de message: Définition de service abstrait

Reference number
ISO/IEC 10021-5:1996(E)

Contents

Foreword

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

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

International Standard ISO/IEC 10021-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 18, *Document processing and related communication*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.413.

This third edition is a revision of the second edition (ISO/IEC 10021-5:1994).

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

— *Part 1: System and Service Overview*

— *Part 2: Overall Architecture*

— *Part 3: Abstract Service Definition Conventions*

— *Part 4: Message Transfer System: Abstract Service Definition and Procedures*

— *Part 5: Message Store: Abstract Service Definition*

— *Part 6: Protocol Specification*

— *Part 7: Interpersonal Messaging System*

— *Part 8: Electronic Data Interchange Messaging Service*

— *Part 9: Electronic Data Interchange Messaging System*

Annexes A to F form an integral part of this part of ISO/IEC 10021. Annexes G to J are for information only.

Introduction

This Recommendation | International Standard is one of a series of Recommendations | International Standards defining Message Handling in a distributed open systems environment.

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

This Recommendation | International Standard defines the Message Store abstract-service (MS abstract-service) which supports message-retrieval from a Message Store (MS) and message-submission through the MS in a Message Handling System (MHS). The MS abstract-service also provides message-administration services, as defined by the Message Transfer System (MTS) abstract-service.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – MESSAGE HANDLING SYSTEMS (MHS): MESSAGE STORE: ABSTRACT SERVICE DEFINITION

SECTION 1 – GENERAL

1 Scope

This Recommendation | International Standard defines the Message Store abstract-service. This abstract-service is provided by the Message Store access protocol (specified in ITU-T Rec. X.419 | ISO/IEC 10021-6) in conjunction with the MTS abstract-service (defined in ITU-T Rec. X.411 | ISO/IEC 10021-4), together with the Remote Operations Service Element (ROSE) services (defined in ITU-T Rec. X.219 | ISO/IEC 9072-1). The abstract-syntax for the application-layer protocols used in this Recommendation | International Standard is defined in ITU-T Rec. X.680 | ISO/IEC 8824-1.

Other Recommendations | parts of ISO/IEC 10021 define other aspects of the MHS. ITU-T Rec. F.400/X.400 | ISO/IEC 10021-1 defines the user-oriented services provided by the MHS. ITU-T Rec. X.402 | ISO/IEC 10021-2 provides an architectural overview of the MHS. ITU-T Rec. X.420 | ISO/IEC 10021-7 defines the abstract-service for Interpersonal Messaging and defines the format of Interpersonal Messages.

Section 2 of this Recommendation | International Standard contains the Message Store abstract-service definition. Clause 6 describes the MS model. Clause 7 defines the semantics and abstract-syntax of the MS-bind and the MS-unbind abstract-operations. Clause 8 defines the semantics and abstract-syntax of the operations of the MS abstract-service. Clause 9 defines the semantics and abstract-syntax of the errors of the abstract-service.

Section 3 of this Recommendation | International Standard defines the general-attribute-types, general-matching-rules, and general-auto-action-types related to the MS. Clause 10 contains an overview. Clause 11 defines the semantics and abstract-syntax of the general-attribute-types. Clause 12 defines the semantics and abstract-syntax of the general-matching-rules. Clause 13 defines the semantics and abstract-syntax of the general-auto-action-types.

Section 4 of this Recommendation | International Standard describes the procedures for Message Store and the ports realization. Clause 14 contains an overview. Clause 15 describes how the Message Transfer System abstract-service is consumed. Clause 16 describes how the Message Store abstract-service is supplied. Clause 17 describes how the MS ports are realized.

The requirements for conformance to this Recommendation | International Standard are stated in clause 10 of ITU-T Rec. X.419 | ISO/IEC 10021-6.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Reference Model references

This Recommendation | International Standard cites the following Reference Model specification:

– ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.*

2.2 Presentation references

This Recommendation | International Standard cites the following Presentation specifications:

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

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

– ITU-T Recommendation X.682 (1994) | ISO/IEC 8824-3:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*

– ITU-T Recommendation X.690 (1994) | ISO/IEC 8825-1:1995, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

2.3 Remote Operations references

This Recommendation | International Standard cites the following Remote Operations specification:

– ITU-T Recommendation X.880 (1994) | ISO/IEC 13712-1:1995, *Information technology – Remote Operations: Concepts, model and notation.*

2.4 Directory references

This Recommendation | International Standard cites the following Directory specifications:

– ITU-T Recommendation X.501 (1993) | ISO/IEC 9594-2:1995, *Information technology – Open Systems Interconnection – The Directory: Models.*

– ITU-T Recommendation X.509 (1993) | ISO/IEC 9594-8:1995, *Information technology – Open Systems Interconnection – The Directory: Authentication framework.*

– ITU-T Recommendation X.520 (1993) | ISO/IEC 9594-6:1995, *Information technology – Open Systems Interconnection – The Directory: Selected attribute types.*

2.5 Message Handling references

This Recommendation | International Standard cites the following Message Handling System specifications:

– ITU-T Recommendation F.400/X.400 (1993), *Message handling services: Message handling system and service overview.*

ISO/IEC 10021-1:1996, *Information technology – Message Handling Systems (MHS) – Part 1: System and service overview.*

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

– ITU-T Recommendation X.411 (1995) | ISO/IEC 10021-4:1996, *Information technology – Message Handling Systems (MHS): Message transfer system: Abstract service definition and procedures.*

– ITU-T Recommendation X.419 (1995) | ISO/IEC 10021-6:1996, *Information technology – Message Handling Systems (MHS): Protocol specifications.*

– ITU-T Recommendation X.420[1)] | ISO/IEC 10021-7...[1)], *Information technology – Message Handling Systems (MHS): Interpersonal messaging system.*

---

1) Presently at the stage of draft.

3 Definitions

3.1 Common Definitions for MHS

For a list of the common definitions for MHS refer to ITU-T Rec. X.402 | ISO/IEC 10021-2.

3.2 Message Store Definitions

For the purposes of this Recommendation | International Standard the following definitions apply:

3.2.1 abstract-association: An abstract binding between two communication partners. In this Service Definition, the binding between an MS-user and an MS for the provision of the MS abstract-service, or between an MS and the MTS for the provision of the MTS abstract-service.

3.2.2 Administration Port: A port which supports the administration services (of the MTS) within the MS abstract-service.

3.2.3 Alert abstract-operation: An abstract-operation which enables the MS to inform the MS-user that a message or report has been delivered to the MS. May be issued only over an existing abstract-association.

3.2.4 attribute: The information of a particular type appearing in an entry.

3.2.5 attribute-type: That component of an attribute which indicates the class of information given by that attribute.

3.2.6 attribute-value: A particular instance of the class of information indicated by an attribute type.

3.2.7 attribute-value-assertion: A proposition, which may be *true, false,* or *undefined,* concerning the values of an attribute in an entry.

3.2.8 auto-action: Actions that are performed automatically by the MS according to instructions previously registered by the MS-user.

3.2.9 Auto-action-log: An entry-class which contains entries that record the performance of certain auto-actions by the MS.

3.2.10 auto-action-event: An entry of the Auto-action-log entry-class which represents an auto-action execution.

3.2.11 auto-action-type: The type of an auto-action, e.g. Auto-alert.

3.2.12 Auto-alert: An auto-action which alerts the MS-user when a message or report is delivered.

3.2.13 Auto-correlate-reports: An auto-action which correlates delivery reports with the originally submitted messages or probes to which they are related.

3.2.14 Auto-delete: An auto-action that deletes messages whose storage period has expired.

3.2.15 Auto-forward: A class of auto-actions which causes the MS to forward a delivered message to one or more recipients. As the definition of Auto-forward is content-specific, it is not defined in this Service Definition. Rather, each type of Auto-forward auto-action is defined in the Specification for the content-type concerned.

3.2.16 Auto-modify: An auto-action which applies modifications to the attributes of newly created entries.

3.2.17 child-entry: An entry immediately subordinate to another entry (its parent-entry) in a tree-structured relationship. An entry which is not a child-entry is a main-entry.

3.2.18 child-sequence-number: A sequence-number in a parent-entry pointing to a child-entry. A parent-entry has one value of child-sequence-number for each child-entry it possesses.

3.2.19 constraining set: An information object set used to constrain the values of related components within a Set or Sequence. See ITU-T Rec. X.682 | ISO/IEC 8824-3.

3.2.20 content-specific: Describes a specification or action whose effect depends on the content-type of the message being handled.

3.2.21 creation-time: An attribute which records the date and time at which the entry was created by the MS.

3.2.22 Delete abstract-operation: An abstract-operation used to delete one or more entries of a specified entry-class.

3.2.23 delivered-EITs: A multi-valued attribute which indicates the encoded-information-types present in the content of a delivered-message.

3.2.24 delivered-message: An entry of the Delivery or Delivery-log entry-classes which represents a delivered-message.

3.2.25 delivered-report: An entry of the Delivery or Delivery-log entry-classes which represents a delivered-report.

3.2.26 Delivery: An entry-class which contains entries that represent messages and reports delivered by the MTS to the MS.

3.2.27 Delivery-log: An entry-class which contains entries that provide, for logging purposes, a restricted representation of messages and reports delivered by the MTS to the MS.

3.2.28 Draft: An entry-class which contains draft-message entries.

3.2.29 draft-message: An entry of the Draft entry-class which represents a message not yet submitted to the MTS.

3.2.30 entry: An information object stored in the MS. See 3.2.17, 3.2.45, and 3.2.61 for further classification of entries.

3.2.31 entry-class: A category of entry which represents a particular type of information object. The principal entry-classes are the Stored-message, the Message-log and the Auto-action-log entry-classes.

3.2.32 entry-information: A parameter, used in abstract-operations, which conveys selected information from an entry.

3.2.33 entry-information-selection: A parameter, used in abstract-operations, which indicates what information from an entry is being requested.

3.2.34 entry-type: An attribute which indicates whether an entry contains a delivered-message, a delivered-report, a returned-content, a submitted-message, a submitted-probe, a draft-message or an auto-action-event entry.

3.2.35 Fetch abstract-operation: An abstract-operation which allows an unrestricted set of attribute information for one selected entry of a specified entry-class to be fetched from the MS by the MS-user.

3.2.36 fetch-restrictions: Restrictions, imposed by the MS-user, on the type of information it is prepared to receive as a result of Fetch. The possible restrictions are on attribute-length, content-types and encoded information types.

3.2.37 filter: A parameter, used in abstract-operations, which applies a test to a particular entry and is either satisfied or not by that entry.

3.2.38 filter-item: An assertion about the presence or value(s) of an attribute of a particular type in an entry under test. Each such assertion is either *true*, *false*, or *undefined*.

3.2.39 forwarding-request: A parameter that may be present in the argument of the MS-message-submission abstract-operation, invoked by the MS-user, to request that a stored message is forwarded from the MS.

3.2.40 general-attribute: An attribute which is valid for all types of messages and reports, independent of content-type. Only attributes of this type are defined in this Service Definition.

3.2.41 general-auto-actions: An auto-action which is valid for all types of messages and reports, independent of content-type. Only auto-actions of this type are defined in this Service Definition.

3.2.42 grade: Defined in 5.2 of ITU-T Rec. X.402 | ISO/IEC 10021-2.

3.2.43 limit: A component of the selector parameter which specifies the maximum number of selected entries to be returned in the result of an abstract-operation.

3.2.44 List abstract-operation: An abstract-operation which allows the selection of entries of a specified entry-class and requests certain attribute information to be returned for those entries.

3.2.45 main-entry: An entry which may form the root of a set of related entries, organized in a tree-structured fashion. Zero or more child-entries may be associated with a main-entry.

3.2.46 matching-rule: A rule which allows entries to be selected by making assertions concerning their attribute-values.

3.2.47 message-group: A set of related entries. An entry's message-group-name attribute indicates the message-groups of which it is a member.

3.2.48 Message-log: An entry-class which incorporates all entries of the Submission-log and Delivery-log entry-classes.

3.2.49 Message-submission abstract-operation: An abstract-operation the MS invokes to submit a message to the MTS when the MS-user invokes MS-message-submission, or when certain content-specific auto-actions are performed.

3.2.50 Modify abstract-operation: An abstract-operation used to modify the attributes of one or more entries within an entry-class.

3.2.51 MS abstract-service: The set of capabilities that the MS offers to its users by means of its ports.

3.2.52 MS abstract-service-user: The user of the MS abstract-service. An end-user may employ different UAs, at different times, to act as the MS-abstract-service-user when accessing the MS abstract-service.

3.2.53 MS abstract-service-provider: The MS which provides the MS abstract-service.

3.2.54 MS-message-submission abstract-operation: An abstract-operation which enables the MS-user to submit a message to the MTS (and optionally store a copy), or to store a draft message.

3.2.55 MS-probe-submission abstract-operation: An abstract-operation which enables the MS-user to submit a probe to the MTS, and optionally store a copy.

3.2.56 MS-submission Port: A port which supports the MS-submission services of the MS abstract-service. The MS-submission abstract-service offers the same services as the Message-submission abstract-service (of the MTS abstract-service), and in addition, offers services for the storage of submitted messages, the logging of submitted messages, and the forwarding of messages residing in the MS.

3.2.57 MS-submission-options: A parameter used in abstract-operations and auto-actions to determine whether a message or probe is to be submitted to the MTS, or stored in the MS, or both submitted and stored.

3.2.58 MS-user: A shorter form of "MS abstract-service-user".

3.2.59 multi-valued attribute: An attribute which may have several values associated with it.

3.2.60 override: A component of the selector parameter which indicates that previously registered restrictions for the performance of theis Fetch abstract-operation shall not apply in this instance of the abstract-operation.

3.2.61 parent-entry: An entry immediately superior to one or more child-entries. A parent-entry which is not a child-entry of some other entry is a main-entry.

3.2.62 parent-sequence-number: A sequence-number which identifies a child-entry's parent-entry. Each child-entry has one value of parent-sequence-number.

3.2.63 partial-attribute-request: A form of the entry-information-selection parameter which requests the return of selected values only, from a multi-valued attribute. See 3.2.33.

3.2.64 range: A parameter used in abstract-operations to select a contiguous sequence of entries of a specified entry-class.

3.2.65 Register-MS abstract-operation: An abstract-operation which allows the MS-user to register information relevant to its interworking with the MS.

3.2.66 registration: Information which is registered with the MS and stored between abstract-associations (until changed by the Register-MS abstract-operation). See 3.2.65.

3.2.67 registration-identifier: An identifier for one particular set of registration parameters for an auto-action-type.

3.2.68 Retrieval Port: A port which supports the retrieval services of the MS abstract-service.

3.2.69 retrieval-status: An attribute which records whether an entry has been retrieved from the MS by the MS-user. Possible values are *new, listed* and *processed*.

3.2.70 returned-content: An entry of the Delivery entry-class which contains the returned content of a previously submitted message.

3.2.71 selector: A parameter used in abstract-operations to select entries of a specified entry-class.

3.2.72 sequence-number: An attribute which unambiguously identifies an entry. Sequence-numbers are allocated in ascending order.

3.2.73 single-valued attribute: An attribute which may have one value only associated with it.

3.2.74 Stored-message: An entry-class which incorporates all entries of the Submission and Delivery entry-classes.

3.2.75 Submission: An entry-class which contains entries that represent messages and probes which have been submitted to the MTS via the MS.

3.2.76 Submission-log: An entry-class which contains entries that provide, for logging purposes, a restricted representation of messages and probes which have been submitted to the MTS.

3.2.77 submitted-message: An entry of the Submission or Submission-log entry-classes which represents a submitted-message.

3.2.78 submitted-probe: An entry of the Submission or Submission-log entry-classes which represents a submitted-probe.

3.2.79 subscription: A long-term agreement between the MS supplier or administrator and the MS customers on the availability and use of optional MS features. This Service Definition assumes that such an agreement can be made, but does not prescribe the method by which it is made.

3.2.80 Summarize abstract-operation: An abstract-operation which provides an overview of the kind and number of entries of some entry-class which are currently stored in the MS.

3.2.81 UA-registration: Information registered with the MS which is specific to one of a user's UAs. See 3.2.66.

4 Abbreviations

The abbreviations used in this Service Definition are defined in clause 5, and in ITU-T Rec. X.402 | ISO/IEC 10021-2, except where noted below.

ASN.1	Abstract Syntax Notation One
DL	distribution-list
EIT	encoded-information-type
MASE-88	Message Administration Service Element 1988
MASE-94	Message Administration Service Element 1994
MHS	Message Handling Systems
MRSE-88	Message Retrieval Service Element 1988
MRSE-94	Message Retrieval Service Element 1994
MS	Message Store
MSSE	Message Submission Service Element
MS-MSSE	MS Message Submission Service Element
MTS	Message Transfer System
ROS	Remote Operations
UA	User Agent
UTC	Universal Coordinated Time (see ITU-T Rec. X.680 | ISO/IEC 8824-1)

5 Conventions

This Service Definition uses the descriptive conventions listed below.

5.1 Conventions for abstract-services

This Service Definition uses the following ASN.1-based descriptive conventions for the indicated purposes:

a) To define the information objects of the MS abstract-service, and other data types and values, ASN.1 itself.

b) To define the MS abstract-service, the MHS-OBJECT, PORT, ABSTRACT-OPERATION, and ABSTRACT-ERROR information object classes of ITU-T Rec. X.411 | ISO/IEC 10021-4. These are derived directly from the corresponding information object classes defined in ITU-T Rec. X.880 | ISO/IEC 13712-1, which also defines the CONTRACT and CONNECTION-PACKAGE information object classes.

c) To define attributes, the ATTRIBUTE information object class of 6.3.3.3.

d) To define auto-actions, the AUTO-ACTION and AUTO-ACTION-ERROR information object classes of 6.5.

e) To define matching-rules, the MATCHING-RULE information object class of ITU-T Rec. X.501 | ISO/IEC 9594-2 (see also 6.3.9.3).

Whenever this Service Definition describes a class of data structure having components, each component is categorized as one of the following **grades**:

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

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

c) **Conditional** (C): A conditional component shall be present in an instance of the class as specified by this Service Definition.

5.2 Conventions for attribute-types used in Tables 2 and 3 of clause 11

This Service Definition uses the conventions listed below in its definition of the attribute-types for the MS abstract-service.

For the column headed "Single/Multi-valued" the following values may occur:

S Single-valued
M Multi-valued

In Table 2, for the column headed "Support level by MS", the sub-heading "Stored-message entry-class" refers to the Stored-message (and the Submission, Delivery and Draft) entry-classes, and the sub-heading "Message-log entry-class" refers to the Message-log (and the Submission-log and Delivery-log) entry-classes. The following values may occur in this column in Tables 2 and 3:

M Mandatory
O Optional
– Not supported

For the columns headed "Presence in" any of delivered-message, delivered-report, returned-message, submitted-message, submitted-probe entry, draft-message, and Auto-action-log entries, the presence of each attribute-type is described by one of the following values:

P Always present in the entry because:

- it is mandatory for generation by the MS; or
- it is a mandatory or defaulted parameter in the relevant abstract-operation.

C Conditionally present in the entry. It is present if:

- it is supported by the MS and subscribed to by the user, and
- it is present in an optional parameter in the relevant abstract-operation.

– Always absent, otherwise.

For the columns headed "Available for List", and "Available for Summarize", the following values may occur:

N No
Y Yes

5.3 Conventions for attribute-types used in Table 4 of clause 11

This Service Definition uses the conventions listed below in its definition of the attribute-types for the MS abstract-service.

For the column headed "Single/Multi-valued", the following values may occur:

S Single-valued
M Multi-valued

For the column headed "Source generated by", the following values may occur:

Amod Auto-modify auto-action
Md Message-delivery abstract-operation
Mod Modify abstract-operation

MS	Message Store
Ms	Message-submission and MS-message-submission abstract-operations
Ps	Probe-submission and MS-probe-submission abstract-operations
Rd	Report-delivery abstract-operation.

5.4 General font conventions

Throughout this Service Definition, terms are rendered in **bold** when defined. Terms that are proper nouns are capitalized, generic terms are not. Multi-word generic terms are hyphenated.

5.5 Font conventions for ASN.1 definitions

Throughout this Service Definition, ASN.1 definitions appear in Times New Roman font and is one size smaller than normal text.

5.6 Rules for ASN.1 definitions

ASN.1 definitions appears both in the body of this Service Definition to aid the exposition, and again, formally, in annexes for reference. If differences are found between the ASN.1 used in the exposition and that formally defined in the corresponding annex, a specification error is indicated.

5.7 Conventions for previous editions of this Service Definition

This Service Definition uses the term "1988 Application Contexts" to refer to MS operation using the ms-access-88 and ms-reliable-access-88 Application Contexts originally defined in CCITT Rec. X.419 (1988) and (1992) | ISO/IEC 10021-6:1990. The term "1994 Application Contexts" refers to MS operation using the ms-access-94 and ms-reliable-access-94 Application Contexts defined in ITU-T Rec. X.419 (1995) | ISO/IEC 10021-6:1990, Amendment 1:1994.

The abstract-syntax defined in this Service Definition may be mapped to that used in previous editions as follows. All ASN.1 definitions of object sets and Enumerated types which contain the ASN.1 extensions marker ("...") are treated as if any extension additions following the marker are absent. For definitions where the extension marker is not used, the ASN.1 comment "-- *1994 extension* --" has a similar interpretation. The effect of this is that certain abstract-operations, entry-classes, entry-types, attribute-types, matching-rules, and auto-action-types are not standardized for use in 1988 Application Contexts. In addition, attribute-types and columns marked with an asterisk (*) in Table 2 are not defined for 1988 Application Contexts.

SECTION 2 – MESSAGE STORE ABSTRACT-SERVICE DEFINITION

6 Message Store model

The Message Store (MS) is modelled as an object which acts as a provider of services to the MS abstract-service-user (the MS-user), and as a user of services provided by the Message Transfer System (MTS).

The MS serves an intermediary role between the MS-user and the MTS. Its primary function is to provide storage for messages submitted by and delivered to a single MHS end-user, and to retain these for subsequent retrieval by the end-user's UA or UAs. The MS also provides message-submission and message-administration services to the MS-user, in effect, via 'pass-through' to the MTS. This enables the MS to provide additional functionality compared to submission directly to the MTS, such as the storage of submitted messages, the forwarding of messages residing in the MS, and the logging of submission and delivery events.

Like the MS-user, the MS acts on behalf of a single MHS end-user only; i.e. it does not provide a common or shared multi-user MS service.

NOTE – In a typical implementation, a single system will serve many users and hence comprise many Message Stores.

The MS is described using an abstract model to define the service it provides – the Message Store abstract-service. Figure 1 shows the MS abstract-service in relation to the MS-user and to the Message Transfer System abstract-service. In this figure, the open boxes represent the consumption of the abstract service, and the closed boxes represent the supply of the abstract service.

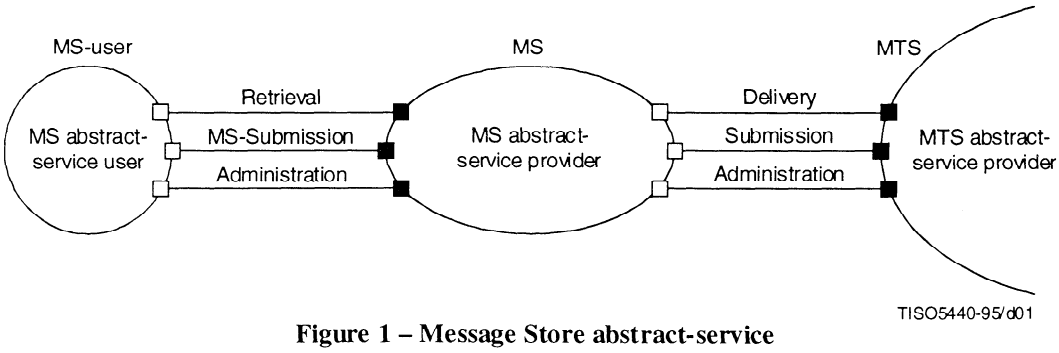

Figure 1 – Message Store abstract-service

For an introduction and description of the abstract-service concept and its definition conventions, see ITU-T Rec. X.402 | ISO/IEC 10021-2. These conventions make use of various information object classes used for the specification of ROS-based application protocols which are defined in ITU-T Rec. X.880 | ISO/IEC 13712-1.

In secure messaging the MS is treated as a separate object with a unique identity and has a separate key (or a set of keys) from the MS-user.

6.1 Message Store objects and contracts

The **Message Store** (MS) is modelled as an object which acts as provider of the MS abstract-service to the MS-user under the terms of an MS-access-contract. The MS is the responder to both the MS-access-contract-88 and the MS-access-contract-94, which are associated with 1988 and 1994 Application Contexts respectively.

```
ms MHS-OBJECT ::= {
    IS          {mts-user}
    RESPONDS    {ms-access-contract-88 | ms-access-contract-94}
    ID          id-ot-ms }
```

To provide the MS-user with access to the services of the MTS, the MS also acts as an MTS-user object; see Figure 2 (Part 2) and clause 8 of ITU-T Rec. X.411 | ISO/IEC 10021-4. In this role it consumes the Submission, Delivery, and Administration abstract-services of the MTS and acts as supplier of the MTS Submission and Administration abstract-services to the MS-user.

The **MS-user** is also modelled as an object. It initiates the MS-access-contract-94 and MS-access-contract-88 under which it acts as consumer of the MS abstract-service.

```
ms-user MHS-OBJECT ::= {
    INITIATES   {ms-access-contract-88 | ms-access-contract-94}
    ID          id-ot-ms-user }
```

The **MS-access-contract-94** defines an association contract established by means of the MS-connect connection package, under which the initiator (the MS-user) acts as consumer of the Retrieval Port and MS-submission Port abstract-services provided directly by the MS, and of the Administration Port abstract-services provided transparently by the MS.

```
ms-access-contract-94 CONTRACT ::= {
    CONNECTION                  ms-connect
    INITIATOR CONSUMER OF       {retrieval | ms-submission | administration}
    ID                          id-crt-ms-access-94 }
```

The **MS-access-contract-88** defines an association contract established by means of the MS-connect connection package under which the initiator (the MS-user) acts as consumer of the Retrieval-88 Port abstract-services provided directly by the MS, and of the Submission and Administration-88 Port abstract-services provided transparently by the MS.

```
ms-access-contract-88 CONTRACT ::= {
    CONNECTION                  ms-connect -- with all 1994 extension additions omitted --
    INITIATOR CONSUMER OF       {retrieval-88 | submission | administration-88}
    ID                          id-crt-ms-access-88 }
```

The **MS-connect** connection package specifies the operations used in the establishment and release of an abstract-association between the MS and MS-user.

```
ms-connect CONNECTION-PACKAGE ::= {
    BIND        ms-bind
    UNBIND      ms-unbind
    ID          id-cp-ms-connection }
```

6.2 Message Store ports

The MS provides the Retrieval, MS-submission, and Administration Ports to the MS-user. The collection of capabilities provided by these ports constitutes the MS abstract-service. The retrieval capabilities are unique to the MS. These capabilities include obtaining summaries of messages, fetching messages (in whole or in part), deleting messages residing in the MS, and registering requests for the MS to perform certain message management activities automatically.

Additional message management services are performed by the MS on the MS-user's behalf, for logging incoming and outgoing messages, for the automatic classification of stored messages, and for auto-correlating delivery reports with the submitted messages or probes to which they are related.

By means of the MS-bind abstract-operation, the MS authenticates an MS-user before providing it with any of the storage and retrieval capabilities indicated above. Similarly, the MTS abstract-service shall authenticate the MTS abstract-service user before extending its services to the MTS abstract-service-user (the MS).

Except for the Alert service of the Retrieval Port, the Submission-control service of the MS-submission Port, and the Change-credentials service of the Administration Port, all the services provided by the MS abstract-service are invoked by the MS-user and performed by the MS.

Security-labels may be assigned to the MS in line with the security-policy in force. The security-policy may also define how security-labels are to be used to enforce the security-policy. If security-labels are assigned to the MS, the handling of stored messages and reports bearing message-security labels may be affected by the security-policy in force. If security-labels are not assigned to the MS, the handling of stored messages and reports bearing message-security labels is discretionary.

If security-contexts are established between the UA and the MS, and between the MS and the MTS, the security-label that is assigned to a message or probe is confined by the security-context in line with the security-policy in force. If security-contexts are not established, the assignment of a message-security-label to a message or probe is at the discretion of the originator.

6.2.1 Retrieval Ports

The **Retrieval Port** is defined as follows:

```
retrieval PORT ::= {
        CONSUMER INVOKES   {summarize | list | fetch | delete | register-MS,
                            ... -- 1994 extension addition --,
                            modify}
        SUPPLIER INVOKES   {alert}
        ID                 id-pt-retrieval-94 }
```

Details of the **Retrieval Port** abstract-services are given in 8.2 and clause 9.

The **Retrieval-88 Port** provided under the MS-access-contract-88 is defined as follows:

```
retrieval-88 PORT ::= {
        -- With all 1994 extension additions to the abstract-operations absent --
        CONSUMER INVOKES   {summarize | list | fetch | delete | register-MS}
        SUPPLIER INVOKES   {alert}
        ID                 id-pt-retrieval-88 }
```

6.2.2 MS-submission Ports

The **MS-submission Port** is defined as follows:

```
ms-submission PORT ::= {
        CONSUMER INVOKES   {ms-message-submission | ms-probe-submission | ms-cancel-deferred-delivery}
        SUPPLIER INVOKES   {ms-submission-control}
        ID                 id-pt-ms-submission }
```

Details of the **MS-submission Port** abstract-services are given in 8.3 and clause 9.

The **Submission Port** provided under the MS-access-contract-88 is defined in 8.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

6.2.3 Administration Ports

The **Administration Port** is defined in 8.4 of (and the **Administration-88** Port in Annex C of) ITU-T Rec. X.411 | ISO/IEC 10021-4.

When invoked by the MS-user, the Change-credentials abstract-service operates end-to-end between the MS-user and the MTS-service-provider, and passes through the MS. The MS stores the new credentials for use when it subsequently binds with the MTS. If the MS-user needs to update the credentials it uses when binding to the MS, then the Register-MS abstract-operation is used (see 8.2.5).

6.3 Information model

The MS models each information object that it stores as an **entry**, which consists of the collection of information known about that object. Each entry belongs to an **entry-class**, which identifies the class of object which the entry describes. An entry comprises a set of **attributes**, which represent the component elements of information associated with the entry. The types of attribute which are present in an entry depend, in part, on the entry-class of the entry.

6.3.1 Entry-classes

An **entry-class** comprises a category or categories of entries that represent a particular type of information object. Three principal entry-classes are defined: the **Stored-message** entry-class, the **Message-log** entry-class, and the **Auto-action-log** entry-class.

a) The **Stored-message** entry-class contains entries which correspond to messages, reports, probes, and draft messages.

b) The **Message-log** entry-class contains entries that provide a restricted representation of messages, reports, and probes. These entries closely resemble the corresponding entries of the Stored-message entry-class, but contain only a subset of the attributes present in the latter in accordance with their logging function.

c) The **Auto-action-log** entry-class contains entries that record the performance of auto-actions by the MS (see 6.3.7.3).

Five subordinate entry-classes are defined which contain subsets of the entries contained in the principal entry-classes. The **Delivery**, **Submission**, and **Draft** entry-classes are subordinates of the Stored-message entry-class. The **Delivery-log** and **Submission-log** entry-classes are subordinates of the Message-log entry-class. Further details of the organization of entry-classes are given in 6.3.7.

When invoking the abstract-operations of the MS abstract-service the MS-user confines the effect of these abstract-operations to those entries of immediate concern by specifying the entry-class of the entries on which to operate.

```
EntryClass ::= INTEGER {
        delivery                (0),
        -- 1994 extensions --
        submission              (1),
        draft                   (2),
        stored-message          (3),
        delivery-log            (4),
        submission-log          (5),
        message-log             (6),
        auto-action-log         (7) } (0..ub-entry-classes)
```

NOTES

1 The term *information-base* used in CCITT Rec. X.413 (1988) and (1992) | ISO/IEC 10021-5:1990 has been replaced by *entry-class* in this Service Definition.

2 In this Service Definition the description of the MS abstract-service assumes that all defined entry-classes are available for use. In practice, the behaviour of a given MS implementation will depend on its support for optional service components (e.g. the optional entry-classes and attribute-types) and on subscription. See 6.3.7.4.

6.3.2 Entries

Each information object stored in the MS (e.g. each submitted-message or delivered-report) is represented as an **entry**. An entry is identified by its **sequence-number** which the MS generates when the entry is created. Sequence-numbers are generated in ascending order and are never re-used.

```
SequenceNumber ::= INTEGER (0..ub-messages)
```

No two entries within the MS possess the same sequence-number except in the following case. Because of the close relationship between entries of the Stored-message entry-class and entries of the Message-log entry-class, corresponding entries of these entry-classes have identical sequence-numbers.

NOTE – The MS may choose to allocate sequence-numbers by using the time of entry creation to a sufficient granularity to ensure uniqueness.

All entries have three further properties in common. The **entry-type** indicates the type of information object represented by the entry, the **creation-time** indicates the date and time at which the entry was created, and the **retrieval-status** indicates whether the entry has been retrieved by the MS-user.

6.3.3 Attributes

An **entry** consists of a set of **attributes**. This is depicted in Figure 2.

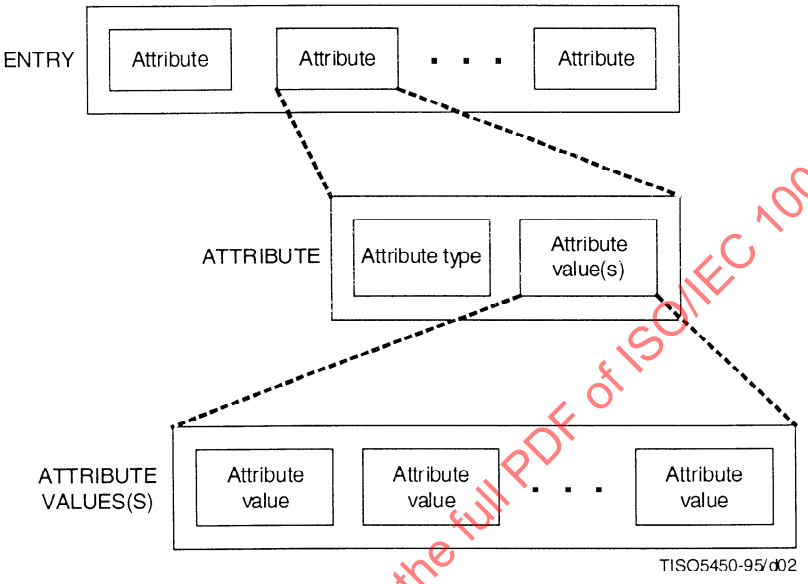

Figure 2 – The Components of an entry

Each **attribute** provides a piece of information about, or derived from, the data to which the entry corresponds. One such piece of information is the sequence-number of the entry itself, and another is the creation-time.

Each attribute is defined as an instance of the ATTRIBUTE information object class (see 6.3.3.3).

An **attribute** consists of an **attribute-type**, which identifies the class of information contained in the attribute, and one or more **attribute-values**, which are the particular instances of that class appearing in the entry. The constraining set (Attribute-table) is defined in 11.2.

```
Attribute ::= SEQUENCE {
    attribute-type      ATTRIBUTE.&id ({AttributeTable}),
    attribute-values    SEQUENCE SIZE (1..ub-attribute-values) OF ATTRIBUTE.&Type
                            ({AttributeTable} {@attribute-type}) }
```

NOTE 1 – Thus, for example, in a delivered-message entry (described in 6.3.6) the attribute-type could be the message's Priority, and the corresponding attribute-value could be *urgent*.

All attributes in an entry shall be of distinct attribute-types.

For some attribute-types, an attribute shall contain only a single attribute-value. Such an attribute-type is said to be **single-valued**. For others, an attribute may contain one or more attribute-values, all of the same ASN.1 data-type. Such an attribute-type is said to be **multi-valued**. Whether an attribute-type is single-valued or multi-valued is stated when the attribute-type is defined (see 6.3.3.3).

NOTE 2 – Thus, for example, the originator-name attribute-type (described in 11.2.47) is single-valued, whereas the other-recipient-names attribute-type (described in 11.2.50) is multi-valued.

In order to select entries from the MS according to the values of their attributes, a set of matching-rules may be associated with the attribute-type to specify the type of matches which may be applied to it (see 6.3.9).

6.3.3.1 Attribute-type

Some attribute-types are internationally standardized. Other attribute-types may be defined by national administrative authorities and private organizations. This implies that a number of separate authorities will be responsible for assigning types in a way that ensures that each is distinct from all other assigned types. This is accomplished by identifying each attribute-type with an Object Identifier when the attribute-type is defined.

```
AttributeType ::= OBJECT IDENTIFIER
```

Certain general-purpose attribute-types applicable to entries of all content-types are defined in clause 11. Such attribute-types are known as **general-attribute-types** and attributes of these types as **general-attributes**.

6.3.3.2 Attribute-values

Defining an attribute-type also involves specifying the syntax, and hence data-type to which every value in the attribute shall conform. The data-type of an attribute-value for the attribute-type is indicated by the Object Identifier for the attribute-type.

6.3.3.3 The ATTRIBUTE information object class

An attribute-type is defined as an instance of the ATTRIBUTE information object class. This class is defined by a set of named field specifications each of which corresponds to some property possessed by each instance of the class. These fields are:

a) an **identifier** for the attribute which denotes the attribute-type;

b) the **attribute-syntax** to which every value of the attribute shall conform;

c) a collection of **matching-rules** identifying the types of matches which may be applied to the attribute (see 6.3.9);

d) an indication of whether an attribute of this type shall contain only a single value, or may contain one or more values, all of the same ASN.1 type.

The defined syntax for specifying instances of the ATTRIBUTE information object class is shown below. This specifies a notation for defining attribute-types that is used throughout this Service Definition.

```
ATTRIBUTE ::= CLASS {
    &id                                 AttributeType UNIQUE,
    &Type,
    &equalityMatch                      MATCHING-RULE OPTIONAL,
    &substringsMatch                    MATCHING-RULE OPTIONAL,
    &orderingMatch                      MATCHING-RULE OPTIONAL,
    &numeration                         ENUMERATED {single-valued(0), multi-valued(1)},
            -- 1994 extension --
    &OtherMatches                       MATCHING-RULE OPTIONAL }
WITH SYNTAX {
    WITH ATTRIBUTE-SYNTAX               &Type,
    [EQUALITY MATCHING-RULE             &equalityMatch,]
    [SUBSTRINGS MATCHING-RULE           &substringsMatch,]
    [ORDERING MATCHING-RULE             &orderingMatch,]
    [OTHER MATCHING-RULES               &OtherMatches,]
    NUMERATION                          &numeration,
    ID                                  &id }
```

The field specifications have the following meaning:

a) The **&id** value field is the identifier field for the class. It distinguishes a particular attribute-type from all others in the class.

b) The **&Type** type field indicates the attribute-syntax to which every value of the attribute-type shall conform.

c) The **&equalityMatch** value field identifies the matching-rule used to evaluate the equality component of a filter-item when applied to the attribute (see 8.1.2.2).

d) The **&substringsMatch** value field identifies the matching-rule used to evaluate the substrings component of a filter-item when applied to the attribute (see 8.1.2.2).

e) The **&orderingMatch** value field identifies the matching-rule used to evaluate the ordering components (greater-or-equal and less-or-equal) of a filter-item when applied to the attribute (see 8.1.2.2).

f) The **&OtherMatches** value set field identifies any further matching-rules defined for the attribute-type.

g) The **&numeration** value field specifies whether the attribute-type is single-valued or multi-valued.

6.3.4 Main-entries, parent-entries, and child-entries

Although entries are generally independent of each other, certain entries are related. A **child-entry** may be the child of another, its **parent-entry**, in a tree-structured relationship. An entry which is not a **child-entry** is termed a **main-entry.**

This relationship is recorded by means of two special general-attributes:

a) **parent-sequence-number**: This single-valued attribute gives the sequence-number of a child-entry's parent-entry. It is absent from a main-entry. Its definition is given in 11.2.51.

b) **child-sequence-numbers**: This multi-valued attribute gives the sequence-numbers of all the child-entries of a parent-entry. It is absent from an entry which is not a parent-entry. Its definition is given in 11.2.8.

The abstract-operations of the MS abstract-service (see clause 8) act by default only on main-entries. Some may be directed to act on all entries, both main-entries and child-entries. In particular, the argument of the Delete abstract-operation (see 8.2.4) shall select only main-entries, in which case the main-entry and all its children and children's children, etc., will also be deleted.

The one case defined in this Service Definition where a child-entry is generated is where a delivered-report contains a returned-content. The delivered-report entry is the main-entry and the returned-content entry is its child-entry. Other Specifications which define message content-types may set down additional rules for the storage of message content which prescribe the use of a main-entry and one or more related child-entries.

6.3.5 Content-specific Attributes

This Service Definition defines only general-attribute-types which model those properties of messages, probes, and reports which pertain regardless of content-type. However, to draw information from the content of a message, the MS must know the content's syntax and semantics, as indicated by the content-type. The **content-specific attributes** are those attribute-types which are defined in the relevant Specification for support of a particular content-type.

A particular instance of the MS may have knowledge of no content-types (a 'generic' MS) or of one or more content-types. When an MS encounters a message with a content-type of which it has no knowledge, it is unable to generate any content-specific attributes in the message entry.

Except for the single case described in 6.3.4, the rules governing the use of child-entries for storing components of message-content are specific to each content-type. If child-entries are used in this way, a content-specific synopsis attribute may provide details of these child-entries (e.g. sizes, sequence-numbers).

Where the content-specific rules defined for a content-type contained in a returned-content entry require the generation of child-entries, such entries shall be assigned an entry-type of delivered-message.

NOTES

1 The Interpersonal Messaging content-type makes use of child-entries for storing message content (see 19.2 of ITU-T Rec. X.420 | ISO/IEC 10021-7). An IP-message which contains one or more forwarded IP-message body-parts is represented in the MS by a main-entry and one child-entry for each IP-message body-part. The **content** general-attribute of the main-entry comprises the complete content. Consequently, each IP-message body-part is notionally present both in its own child-entry and in the main-entry.

2 A body-part of type IP-message may itself contain further encapsulated IP-messages in a recursive fashion. The MS models this structure by the recursive allocation of child-entries.

3 The internal structure of the contents of an IP-message is defined by its IPM synopsis attribute. This attribute-type is an example of a content-specific synopsis attribute.

6.3.6 Entry-types

Every entry belongs to one of the **entry-types** defined below.

```
EntryType ::= INTEGER {
    delivered-message       (0),
    delivered-report        (1),
    returned-content        (2),
    -- 1994 extensions --
    submitted-message       (3),
    submitted-probe         (4),
    draft-message           (5),
    auto-action-event       (6) }
```

A **delivered-message** entry is created by the performance of the Message-delivery abstract-operation of the Delivery Port, and contains the information associated with a delivered message. **Delivered-report** entries and **returned-content** entries are created by the performance of the Report-delivery abstract-operation of the Delivery Port and represent, respectively, a delivered report, and the returned-content conditionally present in a delivered report.

A **submitted-message** entry is created by the performance of the MS-message-submission abstract-operation of the MS-submission Port, or by the performance of an auto-action that invokes the Message-submission abstract-operation of the MTS Submission Port, and contains the information associated with a submitted message. A **submitted-probe** entry is created by the performance of the MS-probe-submission abstract-operation of the MS-submission Port and contains the information associated with a submitted-probe.

A **draft-message** entry is created by the performance of the MS-message-submission abstract-operation of the MS-submission Port, and represents a message which has not been submitted to the MTS, but is stored for possible future retrieval and submission. The creation of a draft-message entry is at the discretion of the MS-user for each MS-message-submission.

An **auto-action-event** entry is created by the performance of an auto-action by the MS, and records the outcome of that auto-action execution (see 6.5). One or more entries may be created for each occasion on which an auto-action is executed by the MS, and so provide a record of the auto-actions performed by the MS on behalf of the MS-user. The definition of an auto-action states whether its performance may be recorded by the creation of auto-action-event entries.

A delivered-message, returned-content, submitted-message, or draft-message entry may contain a message content of any content-type.

The general-attribute-types present in each of the entry-types listed above are enumerated in Tables 2 and 3 and defined in clause 11.

6.3.7 Organization of entry-classes

The class of information object represented by and the entry-types present in each entry-class are defined below. Table 1 gives a summary of the entry-types present in each entry-class. The general-attribute-types supported in each entry-class are listed in Tables 2 and 3 and defined in clause 11.

Table 1 – Entry-types present in entry-classes

Entry-classes	Entry-types						
	delivered-message entry	delivered-report entry	returned-content entry	submitted-message entry	submitted-probe entry	draft-message entry	auto-action-event entry
Delivery	x	x	x	–	–	–	–
Submission	–	–	–	x	x	–	–
Draft	–	–	–	–	–	x	–
Stored-message	x	x	x	x	x	x	–
Delivery-log	x	x	–	–	–	–	–
Submission-log	–	–	–	x	x	–	–
Message-log	x	x	–	x	x	–	–
Auto-action-log	–	–	–	–	–	–	x
x entry-type present – entry-type absent							

6.3.7.1 The Stored-message entry-class

The **Stored-message** entry-class contains entries which represent complete messages, reports, probes, and draft messages for an unrestricted range of content-types. Three subordinate entry-classes are defined which contain subsets of the entries contained in the Stored-message entry-class:

a) The **Delivery** entry-class contains entries with entry-type delivered-message, delivered-report, and returned-content, which are created by the performance of the Message-delivery and Report-delivery abstract-operations of the Delivery Port. The MS-user has no selective control over the creation of entries of the Delivery entry-class.

b) The **Submission** entry-class contains entries with entry-type submitted-message and submitted-probe which are created by the performance of the MS-message-submission and MS-probe-submission abstract-

operations, or by the performance of auto-actions which invoke the Message-submission abstract-operation. The MS shall create a submitted-message or submitted-probe entry only if the submission is successful, and if so requested by the MS-user. The request may be made explicitly in the argument of the submission or the registration-parameter of the auto-action, or may be registered for all submissions by means of the Register-MS abstract-operation.

c) The **Draft** entry-class contains entries with entry-type draft-message. The value of the submission-options parameter (which is optionally present in the argument of the MS-message-submission abstract-operation) controls whether a Draft entry is created (with no submission to the MTS), or whether the message is actually submitted (see 8.1.6). The entries of the Draft entry-class differ mainly from those of the Submission entry-class in that they lack those attributes which are derived from the result parameter of the Message-submission abstract-operation returned by the MTS after successful submission.

6.3.7.2 The Message-log entry-class

The **Message-log** entry-class contains entries which provide a restricted representation of messages, reports, and probes for messages of any content-type. These entries are intended to be long-lived, and contain only a subset of the attributes present in the corresponding entries of the Stored-message entry-class (see Table 2).

Two subordinate entry-classes are defined which contain subsets of the entries contained in the Message-log entry-class:

a) The **Delivery-log** entry-class contains entries with entry-type delivered-message and delivered-report.

b) The **Submission-log** entry-class contains entries with entry-type submitted-message and submitted-probe.

A close relationship exists between entries of the Message-log and Stored-message entry-classes, the Delivery-log and Delivery entry-classes, and the Submission-log, and Submission entry-classes. The relationship between these paired entry-classes is realized by the application of the following rules:

a) One Message-log entry exists for every Stored-message main-entry, each possessing identical sequence-number and creation-time attributes. However, a Message-log entry may be retained after the corresponding Stored-message entry has been deleted. A Message-log entry is created at the same time as the corresponding Stored-message entry is created.

b) A Submission-log entry shall be created upon the submission of a message or probe, regardless of whether the submission is successful or of whether a Submission entry is also created (see 6.3.7.1). If the submission is not successful, the Submission-log entry shall not contain those attributes which are derived from Message-submission or Probe-submission result but shall contain an MS-submission-error attribute to indicate the reason for the failure of the submission. The MS-user has no selective control over the creation of Submission-log or Delivery-log entries.

c) Every attribute of a Message-log entry (except for deletion-time and MS-submission-error, which are specific to Message-log entries) is identical with its counterpart in the corresponding Stored-message entry. The attributes of Message-log entries shall not be subject to modification directly (by means of the Modify abstract-operation or the Auto-modify auto-action), except for the message-group-name attribute which may be modified if no corresponding Stored-message entry exists. However, when an attribute of a Stored-message entry is modified the modification is also apparent in the corresponding Message-log entry.

d) Certain attributes present in Stored-message entries shall be absent from the corresponding Message-log entries. These are message-delivery-envelope, message-submission-envelope, probe-submission-envelope, report-delivery-envelope, and content. The set of attribute-types available in Message-log entries is not prescribed, and is determined by subscription.

e) A Message-log entry cannot be deleted before any corresponding Stored-message entry has been deleted. However, as a matter of local policy, the deletion of Message-log entries may be subject to additional restrictions. The MS shall continue to update attributes which hold correlation information in a Message-log entry even after the corresponding Stored-message entry has been deleted.

f) The deletion-time attribute of a Message-log entry is created when the corresponding Stored-message entry is deleted. In the case of a Submission-log entry for which no Submission entry is created, the deletion-time attribute is generated when the Submission-log entry is created, and is assigned the same value as the entry's creation-time.

g) Other Specifications which define message content-types may set down rules for the creation of a Message-log child-entry for each child-entry present in a Stored-message entry. No such rules are defined in this Service Definition. In particular, the Message-log entry-class shall not contain a child-entry representing the returned-content of a delivered-report.

NOTE – Entries of the Draft and Auto-action-log entry-classes are not related to entries of the Message-log entry-class.

6.3.7.3 The Auto-action-log entry-class

The **Auto-action-log** entry-class contains entries of a single entry-type which record the outcome of auto-action executions (see 6.5). No subordinate entry-classes are defined. The definition of an auto-action-type specifies whether it may cause the generation of Auto-action-log entries. As a subscription option, the MS may record all such auto-action executions or only those which result in an auto-action-error, or may record no executions (see 6.5.3).

6.3.7.4 Entry-class support

When using a 1988 Application Context only the Delivery entry-class shall be available.

When using a 1994 Application Context support for the Delivery entry-class is mandatory. Support for a subordinate entry-class implies support of the entry-class to which it is subordinate. Consequently, support for the Stored-message entry-class is also mandatory (but if no other entry-classes are supported, it shall contain only those entries also present in the Delivery entry-class). Similarly, the Message-log entry-class shall be supported if either the Submission-log or Delivery-log entry-classes is supported. The Submission-log entry-class shall be supported if both the Submission entry-class and the Delivery-log entry-class are supported. The Delivery-log entry-class shall be supported if the Submission-log entry-class is supported.

The availability of the optional entry-classes is subject to subscription.

NOTE – Support for an auto-action does not in itself imply support for the Auto-action-log entry-class.

6.3.8 Retrieval-status

An important property of each entry in any entry-class is its **retrieval-status**:

```
RetrievalStatus ::= INTEGER {
    new          (0),
    listed       (1),
    processed    (2) }
```

The values of retrieval-status are as follows:

a) *new:* The entry has neither been Listed by the MS-user nor has it been automatically *processed* by the MS.

b) *listed:* Information about the entry has been returned to the MS-user in either a List abstract-operation or a Fetch abstract-operation, but the entry has not yet been completely *processed*.

c) *processed:* Either an MS-user has 'completely fetched' the entry, or the MS has performed an auto-action on it, and the definition of that auto-action causes its retrieval-status to be set to *processed*. (Note that some auto-actions result in the message being deleted.) The meaning of 'completely fetched' depends on the message content-type and is defined in the relevant Specification.

The retrieval-status of a delivered-report becomes *processed* when the report-delivery-envelope is retrieved. The retrieval-status of an entry in the Auto-action-log entry-class becomes *processed* when all its attributes are retrieved.

The retrieval-status of a child-entry is maintained according to the same rules as apply to a main-entry. A change in the retrieval-status of an entry may result from operations performed on the entry's parent or child-entry. If a child-entry is logically present in an attribute of its parent-entry, the retrieval of the attribute is regarded as equivalent to the retrieval of all attributes of the child-entry. If an attribute of an entry is logically present in one or more child-entries, then retrieval of all those child-entries is equivalent to retrieval of the attribute.

The retrieval of attributes from a Message-log entry shall not affect the retrieval-status of that entry or of the corresponding Stored-message entry. The retrieval-status of a Message-log entry shall not be affected by the deletion of the corresponding Stored-message entry.

NOTE – The term *entry-status* used in CCITT Rec. X.413 (1988) and (1992) | ISO/IEC 10021-5:1990 has been replaced by the term *retrieval-status* in this Service Definition.

6.3.9 Matching-rules

An important capability of the Message Store abstract-service is the ability to select a set of entries of some entry-class based on assertions concerning attribute-values held by those entries.

A matching-rule allows entries to be selected by making a particular assertion concerning their attribute-values.

The most primitive type of assertion is the attribute-value-assertion (see 8.1.2.3). More complex assertions may be supported using matching-rule-assertions. A matching-rule-assertion is a proposition, which may be *true*, *false* or *undefined*, concerning the presence in an entry of an attribute-value which meets the criteria defined by the matching-rule.

An attribute-value-assertion or matching-rule-assertion is evaluated based on the matching-rule associated with the assertion.

A matching-rule is defined through the specification of:

- the types of matches supported by the rule;
- the syntax required to express an assertion of each type of match.

Some matching-rules are defined in ITU-T Rec. X.501 | ISO/IEC 9594-2 and ITU-T Rec. X.520 | ISO/IEC 9594-6 and in clause 12. Other matching-rules required to support content-specific attributes may be defined in the Specifications for those content-types, and yet others may be defined by national administrative authorities and private organisations.

NOTE – No restrictions are placed on the matching-rules that may be defined for a content-type or may be locally defined. However, locally defined rules may not be widely supported by MSs and UAs. Wherever possible, the matching rules defined in this or other Specifications should be used in preference to the creation of new ones.

Sometimes there will be a one-to-one correspondence between a matching-rule and the types of matches supported. For example, the MS Abstract Service supports a presence relation allowing a present match.

Sometimes there will be a many to one correspondence between a rule and the types of matches supported. For example, the MS Abstract Service supports a generic ordering rule allowing greater than or equal and less than or equal types of matches.

6.3.9.1 Generic Matching Rules

There are a number of generic matching rules common to values of many different types of attribute. These rules are:

- present;
- equality;
- substrings;
- ordering;
- approximate.

Syntax for asserting certain types of matches associated with these generic rules has been built into the MS Abstract Service:

- a **present** syntax for the present match;
- an **equality** syntax for the equality match;
- **initial, any** and **final** syntaxes for the substrings match;
- **greaterOrEqual** and **lessOrEqual** syntaxes for the ordering match;
- an **approximateMatch** syntax for the approximate match.

The MS supports a present matching-rule which may be applied to an attribute of any type. The present match tests for the presence of any value of a particular type.

The MS supports an approximate matching-rule whose definition is a local matter to an MS.

When an attribute-type is defined, the matching-rules associated with the equality, substrings, and ordering generic rules supported by the MS Abstract Service shall be indicated. These rules are used when evaluating assertions of the equality, ordering and substrings relations made using the syntax provided in the MS Abstract Service. If specific rules are not defined, then the only assertion which may be made concerning the attribute is the present match.

When a generic match is performed, the syntax of the presented value must be identical with the attribute-syntax. However, a matching-rule may be defined with an assertion-syntax which is a CHOICE of types, so that it can be applied to attribute-types with different attribute-syntaxes (e.g. different character string types). Consequently, the possible matching-rules that may be specified as a generic matching-rule in an attribute definition are constrained to those for which the attribute-syntax is a subtype of the assertion-syntax.

A matching-rule may be used both as a generic matching-rule for one attribute-type and as an other-match rule for some other attribute-type (see 8.1.2.2).

6.3.9.2 Matching Rule definition

The definition of a matching-rule involves:

a) assigning an Object Identifier to the matching-rule;

b) specifying the different types of matches supported by the rule;

c) defining the syntax of an assertion of the matching-rule;

d) defining the appropriate rules for evaluating a presented assertion with respect to target attribute-values held in an entry.

A matching-rule may apply to different types of attributes with different attribute syntaxes.

The definition of a matching-rule shall include a specification of the syntax of an assertion of the matching-rule, and the way in which values of this syntax are used to perform a match. This does not require a full specification of the attribute syntax to which the matching-rule may apply. The definition of a matching-rule for use with attributes with different ASN.1 syntaxes shall specify how matches shall be performed. The syntax used in the matching-rule-assertion (i.e. for the match-value component of the matching-rule-assertion) is the matching-rule's assertion-syntax.

6.3.9.3 The MATCHING-RULE information object class

Each matching-rule defined in this Service Definition is an instance of the MATCHING-RULE information object class, which is defined in ITU Rec. X.501 | ISO/IEC 9594-2. It is reproduced below as an aid to exposition:

```
MATCHING-RULE ::= CLASS {
        &AssertionType          OPTIONAL,
        &id                     OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        [SYNTAX     &AssertionType]
        ID          &id }
```

The object class contains the following fields:

a) The **&AssertionType** type field indicates the assertion-syntax, i.e. the syntax to which every value in a presented matching-rule-assertion shall conform. It is always present in matching-rules used in this Service Definition.

b) The **&id** value field is the identifier field for the class and contains an Object Identifier.

NOTES

1 No support is provided in the defined syntax for actually defining the matching-rules themselves; this must be done by natural language or by other means.

2 The defined syntax is used to define matching-rules in this and other Specifications.

6.4 Message grouping

The purpose of **message-grouping** is to provide an organizational framework which enables the user to establish a classification scheme for the storage of messages, supported by the manual and automatic assignment of messages to user-defined categories. The allows the user to arrange that submitted messages and probes, delivered messages and reports, and draft messages are stored in such a way that they may be subsequently located, by selection based upon classifications devised by the user and tailored to suit that user's pattern of communication.

Message-grouping employs the **message-group-name** attribute-type (see 11.2.33) which has the following attribute-syntax:

```
MessageGroupName ::= SEQUENCE SIZE(1..ub-group-depth) OF GroupNamePart

GroupNamePart ::= GeneralString (SIZE(1..ub-group-part-length))
```

Zero or more message-group-name values may be attached to any entry in the MS, with the exception of entries of the Auto-action-log entry-class. All entries which possess the same message-group-name attribute-value may be regarded as members of the same message-group. The MS provides the following facilities to support message-grouping:

a) The MS-user shall register with the MS, by means of the Register-MS abstract-operation, the set of message-group-names that the user intends to employ for the classification of messages. This is analogous

to the creation of a directory structure in a filestore. Subsequently, the MS shall verify that only registered message-group-names are assigned to entries, to ensure internal consistency. A message-group-descriptor may accompany each message-group-name registered to provide an informal description of its contents (see 8.2.5).

b) The MS allows the registration of a message-group-name containing more than one group-name-part only if a registration exists for that message-group-name's parent, i.e. the message-group-name formed by omitting the last group-name-part of the name.

c) The MS allows the deregistration of a registered message-group-name provided that the name is not present in the message-group-name attribute of any entry, that the name is not the parent of any other registered message-group, and that the name is not referenced in the modifications component of a registered Auto-modify auto-action.

d) The MS may be instructed to automatically assign appropriate message-group-names to messages, controlled by the specification of selection criteria. This is performed by the Auto-modify auto-action (see 13.2).

e) The MS-user may manually amend the membership of a message-group. This is performed by the Modify abstract-operation (see 8.2.7).

f) The MS-user may retrieve details of the set of registered message-group-names for display to the end-user. This is performed by the Register-MS abstract-operation.

g) The membership of a message-group may be determined by selection based on the value of the message-group-name attribute. The MS need not make any special provision for this as selection of entries based on the values of their attributes is a basic capability of the MS.

NOTE – However, some performance gains may be derived from organizing the storage of entries to reflect message-group membership.

Annex H contains further description of message-grouping and its uses.

6.5 Auto-actions

An **auto-action** is an action performed automatically by the MS whenever the conditions defined for its performance are satisfied. These conditions are registered with the MS either by subscription, or by means of the Register-MS abstract-operation (see 8.2.5). The result of the performance of an auto-action is visible externally to the MS.

Each type of auto-action is identified by its **auto-action-type**. The definition of some auto-actions specifies a parameter which the MS will use in performing the auto-action. This parameter, the **registration-parameter**, is presented to the MS when the auto-action is registered. As certain auto-action-types permit more than one registration, each registration is identified by a **registration-identifier**. Registered auto-actions of a given type are executed in ascending order of registration-identifier.

Where a registration-parameter is defined for an auto-action-type, the information it contains may be conveyed to the MS by means of subscription. However, some auto-actions may require that the MS also supports registration of its registration-parameter by means of the Register-MS abstract-operation.

The **general-auto-actions** apply to entries of all content-types and are defined in clause 13. Other auto-action-types specific to a given content-type are defined in the Specification for that content-type.

6.5.1 The AUTO-ACTION information object class

Each type of auto-action is defined as an instance of the AUTO-ACTION information object class:

```
AUTO-ACTION ::= CLASS {
        &id                                 AutoActionType UNIQUE,
        &RegistrationParameter              OPTIONAL,
        &Errors                             AUTO-ACTION-ERROR OPTIONAL }
WITH SYNTAX {
        [REGISTRATION PARAMETER IS          &RegistrationParameter]
        [ERRORS                             &Errors]
        IDENTIFIED BY                       &id }

AutoActionType ::= OBJECT IDENTIFIER
```

The object class contains the following fields:

a) The **&id** is the identifier field for the class and contains an Object Identifier.

b) The **&RegistrationParameter** type field defines the syntax of the registration-parameter associated with the auto-action-type (see 6.5.2). The field is optional as some auto-action-types require no registration-parameter.

c) The **&Errors** value set field defines a set of errors, any of which may arise from an unsuccessful performance of the auto-action (see 6.5.3).

6.5.2 Auto-action registration

An auto-action is registered with the MS by means of the **auto-action-registrations** parameter of the Register-MS abstract-operation (see 8.2.5.1, item a).

```
AutoActionRegistration ::= SEQUENCE {
        auto-action-type            AUTO-ACTION.&id ({AutoActionTable}),
        registration-identifier     [0] INTEGER (1..ub-per-auto-action) DEFAULT 1,
        registration-parameter      [1] AUTO-ACTION.&RegistrationParameter ({AutoActionTable}
                                        {@auto-action-type}) OPTIONAL }
```

The components of **auto-action-registration** are as follows:

a) **Auto-action-type** (M): The Object Identifier which identifies the auto-action-type. The constraining set (Auto-action-table) is defined in clause 13.

b) **Registration-identifier** (O): Used to distinguish this auto-action-registration from other registrations of the same auto-action-type. This component also governs the order of execution of registered auto-actions of this auto-action-type.

c) **Registration-parameter** (O): The parameter that the MS shall use when it performs this registered auto-action.

6.5.3 Auto-action errors

The Auto-action-log entry-class provides a record of the auto-actions performed by the MS (see 6.3.7.3), and is available to the MS-user subject to subscription. The definition of an auto-action-type specifies whether execution of that auto-action may result in the generation of an entry in the Auto-action-log entry-class. Where auto-action processing results in an error, an auto-action-error attribute may be attached to the Auto-action-log entry associated with that execution (if processing is successful, the attribute is absent). In the first abstract-association established after the occurrence of an auto-action-error, the MS informs the MS-user of the error by means of the auto-action-error-indication parameter of the MS-bind-result. This parameter indicates the sequence-number of the first Auto-action-log entry, created since the last abstract-association was established, in which an auto-action-error attribute is present.

Each type of auto-action-error is an instance of the AUTO-ACTION-ERROR information object class which is identical with the ABSTRACT-ERROR information object class:

```
AUTO-ACTION-ERROR ::= ABSTRACT-ERROR
```

Whereas abstract-errors are defined in the context of ROS, and are reported across the boundary between open-systems, auto-action-errors cause the creation of auto-action-error attributes (see 11.2.4) in entries of the Auto-action-log entry-class. This attribute-type has the following syntax:

```
AutoActionError ::= SET {
        error-code          [0] AUTO-ACTION-ERROR.&errorCode ({AutoActionErrorTable}),
        error-parameter     [1] AUTO-ACTION-ERROR.&ParameterType ({AutoActionErrorTable}
                                {@error-code}) OPTIONAL }
```

The constraining set (Auto-action-error-table) is defined in clause 13. This Service Definition does not define auto-action-errors where identical abstract-errors have already been defined. Any abstract-error may be used as an auto-action-error.

6.5.4 Auto-action execution

Auto-actions are performed autonomously by the MS (possibly in the absence of an abstract-association with the MS-user) when given conditions are fulfilled. The first stage of this process is the identification of an entry which has become a candidate for auto-action processing as a consequence of some event. The most common event which initiates

auto-action processing is the creation of an entry (as the result of Message-delivery or Message-submission, for example). Other events, such as the expiry of a timer, may also initiate auto-action processing.

The second stage in the performance of an auto-action is the evaluation of selection criteria optionally present in the auto-action registration-parameter, that the candidate entry must satisfy before the auto-action is applied to it. Some auto-action-types allow more than one set of auto-action registration-parameters to be registered with the MS. The definition of such auto-action-types determines whether the execution of the first auto-action for which the entry has satisfied the selection criteria is the only one performed for that auto-action-type, or whether all registered auto-actions of that type are applied. The order of execution of registered auto-actions of a given auto-action-type is fixed according to the ascending value of the auto-action registration-identifier. The order of execution of each defined auto-action-type is described in clauses 15 and 16.

6.6 MS extensions

This Service Definition defines MS operation regardless of the content-types of the messages present. To accommodate the additional functions which may be required to support specific content-types, and other extensions, the following information object class is defined:

MS-EXTENSION ::= TYPE-IDENTIFIER

This information object class has two fields: an identifier field (an Object Identifier), and a type field which defines the ASN.1 type of each instance of that class of information. See Annex A of ITU-T Rec. X.681 | ISO/IEC 8824-2. This object class is used where the MS abstract-syntax requires a placeholder for information which is specific to each possible content-type. Refer to the Specification for a given content-type for the definition of each instance of use of this object class.

NOTE 1 – For example, 19.5.2.2 of ITU-T Rec. X.420 | ISO/IEC 10021-7 defines IPMS-specific submission options that occupy the MS extensions placeholder defined in MS-submission-options (see 8.1.6).

The following types are derived from the MS-EXTENSION information object class:

MSExtensionItem ::= INSTANCE OF MS-EXTENSION

MSExtensions ::= SEQUENCE SIZE(1..ub-extensions) OF MSExtensionItem

Where the argument of an abstract-operation (other than MS-bind) contains an MS-extension component unknown to the MS, the MS shall generate an MS-extension-error (see 9.12). An MS-extension specific to a particular content-type shall be disregarded when processing an entry of some other content-type. Where the result of an abstract-operation contains an MS-extension component unknown to the MS-user, the MS-user shall ignore the MS-extension.

NOTE 2 – The MS-bind abstract-operation reports unsupported extensions in the MS-bind-result, to permit the establishment of the abstract-association even where an unsupported extension is present.

7 MS-bind and MS-unbind operations

7.1 MS-bind abstract-operation

The **MS-bind** abstract-operation binds the MS-submission, Retrieval and Administration Ports of the MS-user (consumer) to the MS (supplier). The initiator (of the MS-bind) is the MS-user, while the responder is the MS itself. Information exchanged in the argument and result of MS-bind shall apply for the duration of the abstract-association. MS-bind is defined as follows:

```
ms-bind ABSTRACT-OPERATION ::= {
        ARGUMENT        MSBindArgument
        RESULT          MSBindResult
        ERRORS          {ms-bind-error} }
```

NOTE – The MS abstract-service does not provide any explicit mechanisms to support multiple associations; the effect of simultaneously invoking abstract-operations over different abstract-associations is not defined. Where multiple associations are supported (as a local matter) it is the responsibility of the MS-user to co-ordinate such invocations so that mutual interference is avoided (e.g. simultaneous invocations of Modify abstract-operations could fail or produce unintended results).

7.1.1 MS-bind-argument

The **MS-bind-argument** parameters are used to identify, authenticate and set the security-context for an MS-user. They also contain a set of restrictions for entries to be returned as the result of the Fetch abstract-operation, identify a set of

registrations associated with this instance of the MS-user, and may contain a request to be informed of the entry-classes, auto-action-types, attribute-types, matching-rules, message-group support and content-types to which the MS-user has subscribed.

```
MSBindArgument ::= SET {
    initiator-name              ORAddressAndOrDirectoryName,
    initiator-credentials       [2] InitiatorCredentials,
    security-context            [3] IMPLICIT SecurityContext OPTIONAL,
    fetch-restrictions          [4] Restrictions OPTIONAL -- default is none --,
    ms-configuration-request    [5] BOOLEAN DEFAULT FALSE,
                                -- 1994 extensions --
    ua-registration-identifier  [6] RegistrationIdentifier OPTIONAL,
    bind-extensions             [7] MSExtensions OPTIONAL }
```

The parameters of **MS-bind-argument** have the following meaning:

a) **Initiator-name** (C): This parameter contains the name of the MS-user and is defined further in 8.1.1.1.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

b) **Initiator-credentials** (M): This parameter contains the *credentials* of the MS-user. The **initiator-credentials** may be used by the MS to authenticate the identity of the MS-user (see ITU-T Rec. X.509 | ISO/IEC 9594-8).

If simple-authentication is used, the initiator-credentials comprise a simple password. The password is defined in 8.5.11 of ITU-T Rec. X.411 | ISO/IEC 10021-4. If strong-authentication is used, the initiator-credentials comprise an initiator-bind-token, and, optionally, an initiator-certificate. The initiator-bind-token and initiator-certificate are defined in 8.1.1.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4. The initiator-credentials of the MS-user may differ from the initiator-credentials used in the MTS-bind as defined in 8.1.1.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

c) **Security-context** (O): This parameter identifies the **security-context** within which the MS-user proposes to operate. The security-context is defined in 8.1.1.1.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

The security-context comprises one or more security-labels that define the sensitivity of interactions that may occur between the MS-user and the MS for the duration of the abstract-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 MS-user and by the security-labels associated with the MS.

In the absence of this parameter, security-contexts are not established between the MS-user and the MS, and the sensitivity of interactions that may occur between the MS-user and the MS is at the discretion of the invoker of an abstract-operation.

d) **Fetch-restrictions** (O): This specifies restrictions on the entries to be returned as result of the Fetch abstract-operation. The **fetch-restrictions** remain in force for the duration of the abstract-association and apply unless the Fetch argument explicitly overrides the restriction.

In the absence of this argument, the MS shall not enforce any **fetch-restrictions** during the abstract-association.

```
Restrictions ::= SET {
    allowed-content-types       [0] SET SIZE (1..ub-content-types) OF OBJECT IDENTIFIER OPTIONAL
                                -- default is no restriction --,
    allowed-EITs                [1] MS-EITs OPTIONAL -- default is no restriction --,
    maximum-attribute-length    [2] INTEGER OPTIONAL -- default is no restriction -- }
```

The parameter contains the following components:

1) **Allowed-content-types** (C): The content-types that the MS abstract-service-user is prepared to accept as the result of the Fetch abstract-operation. Any message with a content-type other than those specified will not be returned, but result in a fetch-restriction-error.

In the absence of this component, the MS shall not enforce fetch-restrictions on the content-types of entries.

2) **Allowed-EITs** (C): The encoded-information-types that the MS abstract-service-user is prepared to accept as the result of the Fetch abstract-operation. If a message contains encoded-information-types other than those specified, a filtering will take place so that disallowed EIT parts are not returned along with the text of the message. If the whole message consists of disallowed EITs, a fetch-restriction-error will be reported.

```
MS-EITs ::= SET SIZE (1..ub-encoded-information-types) OF MS-EIT

MS-EIT ::= OBJECT IDENTIFIER
```

In the absence of this component, the MS shall not enforce fetch-restrictions on the encoded-information-types of entries.

NOTE 1 – The allowed-EITs component is present for historical reasons and its use is deprecated. The body-parts present in the content of an entry may be discovered by inspecting a content-specific synopsis attribute, and an implementation need only Fetch those body-parts which are acceptable to it.

3) **Maximum-attribute-length** (C): The maximum attribute length, in octets, that the MS-user is prepared to accept as the result of the Fetch abstract-operation. Where the MS-user attempts to Fetch an attribute-value where the length of the encoding exceeds the maximum-attribute-length, then the MS shall return a fetch-restriction-error.

NOTE 2 – The size in octets of the encoding of an attribute-value may vary where ASN.1's Basic Encoding Rules permit several encodings (e.g. primitive and constructed).

NOTE 3 – In 88 Application Contexts, the maximum-attribute-length restriction applies only to an entry's content attribute.

In the absence of this component, the MS shall not enforce fetch-restrictions on the length of attributes.

e) **MS-configuration-request** (C): This parameter specifies whether the MS is requested to return information which identifies the auto-action-types, attribute-types, entry-classes, content-types, matching-rules, and any additional capabilities to which the MS-user has subscribed, and which consequently are available in the course of the abstract-association.

In the absence of this parameter, the MS shall not return MS-configuration information in the MS-bind-result.

NOTE 4 – In practice, an MS-user may cache this information rather than request it in every MS-bind. The MS may indicate a change in the services supported by means of the service-information parameter of MS-bind-result.

f) **UA-registration-identifier** (O): This parameter specifies an identifier for a set of registration information. If a previous invocation of Register-MS recorded a UA-registration bearing this value of **UA-registration-identifier**, then that UA-registration shall apply for the duration of the present abstract-association. If the parameter does not identify a previously recorded UA-registration, then the MS shall indicate this in the MS-bind-result. If this parameter is absent, then the general defaults (list-attribute-defaults, fetch-attribute-defaults, and submission-defaults) shall apply, if previously registered by means of Register-MS (see 8.2.5.1).

```
RegistrationIdentifier ::= PrintableString
    (SIZE(1..ub-ua-registration-identifier-length))
```

g) **Bind-extensions** (O): This parameter specifies extensions requested by the MS-user; if the MS supports the requested bind-extensions, they shall apply for the duration of the abstract-association. Any bind-extension not supported by the MS shall be reported in the MS-bind-result. Any bind-extension that is improperly specified shall be reported in MS-bind-error. Extensions associated with a particular message content-type may be defined in the relevant Specification. In the absence of this parameter, no extensions are requested.

7.1.2 MS-bind-result

```
MSBindResult ::= SET {
    responder-credentials          [2]  ResponderCredentials,
    available-auto-actions         [3]  SET SIZE (1..ub-auto-actions) OF AUTO-ACTION.&id
                                        ({AutoActionTable}) OPTIONAL,
    available-attribute-types      [4]  SET SIZE (1..ub-attributes-supported) OF ATTRIBUTE.&id
                                        ({AttributeTable}) OPTIONAL,
    alert-indication               [5]  BOOLEAN DEFAULT FALSE,
    content-types-supported        [6]  SET SIZE (1..ub-content-types) OF OBJECT IDENTIFIER OPTIONAL,
                                        -- 1994 extensions --
    entry-classes-supported        [7]  SET SIZE (1..ub-entry-classes) OF EntryClass OPTIONAL,
    matching-rules-supported       [8]  SET SIZE (1..ub-matching-rules) OF OBJECT IDENTIFIER OPTIONAL,
    additional-capabilities        [9]  MSExtensions OPTIONAL,

    message-group-depth            [10] INTEGER (1..ub-group-depth) OPTIONAL,
    auto-action-error-indication   [11] AutoActionErrorIndication OPTIONAL,
    unsupported-extensions         [12] SET SIZE (1..ub-extensions) OF OBJECT IDENTIFIER OPTIONAL,
    ua-registration-id-unknown     [13] BOOLEAN DEFAULT FALSE,
    service-information            [14] GeneralString (SIZE (1..ub-service-information-length)) OPTIONAL }
```

The parameters of **MS-bind-result** have the following meaning:

a) **Responder-credentials** (M): This parameter contains the credentials of the responding MS. The **responder-credentials** may be used by the MS-user to authenticate the identity of the MS (see ITU-T Rec. X.509 | ISO/IEC 9594-8).

If simple-authentication is used, the responder-credentials comprise a simple password associated with the MS. The password is defined in 8.5.11 of ITU-T Rec. X.411 | ISO/IEC 10021-4. If strong-authentication is used, the responder-credentials comprise a responder-bind-token generated by the MS. The responder-bind-token is defined in 8.1.1.1.2.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

b) **Available-auto-actions** (C): This identifies the set of auto-action-types to which the MS-user has subscribed. This parameter shall be present if the MS-configuration-request parameter of the MS-bind-argument is *true* and at least one auto-action-type is available; it is absent otherwise. The constraining set (Auto-action-table) is defined in clause 13.

c) **Available-attribute-types** (C): This identifies the set of attribute-types to which the MS-user has subscribed. This parameter shall be present if the MS-configuration-request parameter of the MS-bind-argument is *true*; it is absent otherwise.

NOTE 1 – All available attribute-types should be identified in this parameter, including those for which support is mandatory. The set of attribute-types for which support is mandatory may change over time.

d) **Alert-indication** (C): If *true*, then an **alert** condition has occurred since the last successful **alert-indication**.

e) **Content-types-supported** (C): This identifies the set of content-types to which the MS-user has subscribed, and for which the MS offers specific support, as defined in the relevant content-type Specification. A message whose content-type is absent from this set may still be submitted, or delivered, and subsequently retrieved. In this case, none of the attribute-types or auto-action-types specific to that content-type will be available to the MS-user.

This parameter shall be present if the MS-configuration-request parameter of the MS-bind-argument is *true* and at least one content-type is supported; it is absent otherwise.

f) **Entry-classes-supported** (C): This identifies the entry-classes supported by the MS. Support for an entry-class implies that it shall be available to the Summarize, List, Fetch, Delete, and Modify abstract-operations, and to the auto-action-types supported by the MS. In addition, support for the Submission entry-class implies support for its use in the MS-message-submission and MS-probe-submission abstract-operations. Support for the Draft entry-class implies support for its use in the MS-message-submission abstract-operation. This parameter shall be present if the MS-configuration-request parameter of the MS-bind-argument is *true*; it is absent otherwise.

g) **Matching-rules-supported** (C): This identifies the matching-rules supported by the MS. Support for a matching-rule implies that it shall be available for use with an attribute for which the matching-rule is defined, where support for that attribute-type is also claimed. This parameter shall be present if the MS-configuration-request parameter of the MS-bind-argument is *true* and at least one matching-rule is supported; it is absent otherwise.

NOTE 2 – All supported matching-rules should be identified in this parameter, including those for which support is mandatory. The set of matching-rules for which support is mandatory may change over time.

h) **Additional-capabilities** (C): This parameter indicates additional capabilities, possibly content-specific, supported by the MS. It shall be present if the MS-configuration-request parameter of the MS-bind-argument is *true* and at least one additional capability is supported; it is absent otherwise.

i) **Message-group-depth** (C): This indicates the maximum number of group-name-parts that may be present in a message-group-name. It shall be present if the MS-configuration-request parameter of the MS-bind-argument is *true* and the MS supports the message-group-name attribute-type; it is absent otherwise. If the MS-user presents, in the argument of an abstract-operation or in an auto-action registration-parameter, a message-group-name which contains more group-name-parts than are specified in this parameter, then a message-group-error results.

j) **Auto-action-error-indication** (C): This shall be present if an auto-action performed by the MS since the previous abstract-association was established caused the generation of an auto-action-error. Since this information cannot be reported when a 1988 Application Context is in use, the establishment of such an association shall not affect the setting of this parameter which shall be presented when a 1994 Application Context is next used.

```
AutoActionErrorIndication ::= CHOICE {
        indication-only          [0] NULL,
        auto-action-log-entry    [1] SequenceNumber }
```

If the Auto-action-log entry-class is subscribed to, the parameter indicates the sequence-number of the first entry which records an auto-action-error which occurred since the previous abstract-association (which used a 1994 Application Context) was established. Otherwise it contains a Null. The parameter shall be absent if no auto-action-errors occurred since the previous abstract-association was established.

k) **Unsupported-extensions** (C): If the MS-user supplied a bind-extensions parameter in the MS-bind-argument and one or more of these extensions are not supported by the MS, this parameter identifies all such unsupported-extensions. The parameter contains the Object Identifier values of each of these unsupported-extensions, drawn from the bind-extensions parameter. The parameter is present if the bind-extensions parameter of the MS-bind-argument is present and at least one of the extensions is not supported; it is absent otherwise.

l) **UA-registration-id-unknown** (C): This parameter is present and has the value *true* if the MS-bind-argument specifies a UA-registration-identifier which has not been registered previously by means of the Register-MS abstract-operation. It is absent otherwise.

m) **Service-information** (C): This provides information concerning the operation of the service. An MS implementation may use this to report operational matters such as service availability and resource usage. The parameter shall be present at the discretion of the MS service provider.

7.1.3 MS-bind-error

An **MS-bind-error** reports a problem in attempting to establish an abstract-association. It is defined as follows:

```
ms-bind-error ABSTRACT-ERROR ::= {
    PARAMETER    CHOICE {
        unqualified-error    BindProblem,
                             -- 1994 extension --
        qualified-error      SET {
            bind-problem                [0] BindProblem,
            supplementary-information   [1] GeneralString (SIZE(1..ub-supplementary-info-length)) OPTIONAL,
            bind-extension-errors       [2] SET SIZE(1..ub-extensions) OF OBJECT IDENTIFIER OPTIONAL } } }

BindProblem ::= ENUMERATED {
    authentication-error                (0),
    unacceptable-security-context       (1),
    unable-to-establish-association     (2),
    ... -- 1994 extension addition --,
    bind-extension-problem              (3) }
```

The parameters of **MS-bind-error** are as follows:

a) **Unqualified-error** (C): This indicates the nature of the error. If *authentication-error* is indicated, the abstract-association cannot be established because the initiator's credentials are not acceptable or are improperly specified. If *unacceptable-security-context* is indicated, then the security-context proposed by the initiator of the abstract-association is unacceptable to the responder. If *unable-to-establish-association* is indicated, then the responder has rejected the initiator's attempt to establish an abstract-association for an unspecified reason. If *bind-extension-problem* is indicated, the abstract-association cannot be established because the bind-extensions parameter specified in the MS-bind-argument contains an improperly specified extension.

b) **Qualified-error** (C): This parameter indicates the nature of the error, and may, in addition, specify **supplementary-information** which expands on the error. Where one or more bind-extensions parameters have been improperly specified, the Object Identifier values of these extensions shall be reported in **bind-extension-errors**.

7.2 MS-unbind abstract-operation

The **MS-unbind** abstract-operation closes the abstract-association, and causes the relaxation of any **fetch-restrictions** that were specified in the argument of the MS-bind abstract-operation. There is no argument, result, or error associated with the MS-unbind abstract-operation.

```
ms-unbind ABSTRACT-OPERATION ::= emptyUnbind
```

8 Abstract-operations

This clause defines the capabilities of the MS abstract-service provided to an MS-user at the Retrieval Port and MS-submission Port of the MS. These capabilities are modelled as abstract-operations which may cause abstract-errors when invoked. The abstract-operations available at the Retrieval Port are defined in 8.2 and those of the MS-submission Port in 8.3. The abstract-errors they may give rise to are the subject of clause 9. The ports are formally defined in 6.2.

8.1 Common data-types used in abstract-operations

This subclause defines a number of common data-types which are used in several of the abstract-operations defined in clause 8. The following common data-types are defined:

a) Range;

b) Filter;

c) Selector;

d) Entry Information Selection;

e) Entry Information;

f) MS Submission Options;

g) Common Submission Results.

8.1.1 Range

A **range** parameter is used to select a contiguous sequence of entries of a specified entry-class.

```
Range ::= CHOICE {
    sequence-number-range       [0] NumberRange,
    creation-time-range         [1] TimeRange }

NumberRange ::= SEQUENCE {
    from        [0] SequenceNumber OPTIONAL -- omitted means no lower bound --,
    to          [1] SequenceNumber OPTIONAL -- omitted means no upper bound -- }

TimeRange ::= SEQUENCE {
    from        [0] CreationTime OPTIONAL -- omitted means no lower bound --,
    to          [1] CreationTime OPTIONAL -- omitted means no upper bound -- }

CreationTime ::= UTCTime
```

The components of range have the following meaning:

a) **Sequence-number-range** (C); and

b) **Creation-time-range** (C): Both of these parameters identify the contiguous sequence of entries to be selected. The **sequence-number-range** is given in terms of sequence-numbers, and the **creation-time-range** is given in terms of creation-times. The creation-time of an entry is the time at which the MS generated the entry. The sequence-numbers of successive entries are always in ascending order, but several adjacent entries may have the same creation-time. The parameters of both number-range and time-range have the following meaning:

1) **From** (O): This is the lower bound for the range.

   In the absence of this component, the default is no lower bound, and the selection starts with the earliest entry (lowest sequence-number) of the specified entry-class.

2) **To** (O): This is the upper bound for the range.

   In the absence of this component, the default is no upper bound, and the selection finishes with the latest entry (highest sequence-number) of the specified entry-class.

If **from** is greater than **to**, then the range is defined in *descending* order; otherwise, the range is defined in *ascending* order (see 8.1.3). If a 1988 Application Context is in use, range shall be specified in ascending order. A range is inclusive of its upper and lower bounds.

8.1.2 Filters

8.1.2.1 Filter

A **filter** parameter applies a test to a particular entry and is either satisfied or not by the entry. The **filter** is expressed in terms of assertions about the presence or value(s) of certain attributes of the entry, and is satisfied if and only if it evaluates to *true*. A **filter** may be *true, false,* or *undefined.*

```
Filter ::= CHOICE {
    item        [0] FilterItem,
    and         [1] SET OF Filter,
    or          [2] SET OF Filter,
    not         [3] Filter }
```

A filter is either a filter-item, or an expression involving simpler filters composed together with the logical operators **and**, **or**, and **not**:

a) A filter which is an **item** has the value of the filter-item (i.e. *true, false,* or *undefined*).

b) A filter which is the **and** of a set of filters is *true* if the set is empty or if each filter is *true*; it is *false* if at least one filter is *false*; otherwise it is *undefined* (i.e. at least one filter is *undefined* and no filters are *false*).

c) A filter which is the **or** of a set of filters is *false* if the set is empty or if each filter is *false*; it is *true* if at least one filter is *true*; otherwise it is *undefined* (i.e. at least one filter is *undefined* and no filters are *true*).

d) A filter which is the **not** of a filter is *true* if the filter is *false*; *false* if it is *true*; and *undefined* if it is *undefined*.

8.1.2.2 Filter-item

A **filter-item** is an assertion about the presence or value(s) of an attribute of a particular type in the entry under test. Each such assertion is *true, false,* or *undefined*.

```
FilterItem ::= CHOICE {
    equality                [0] AttributeValueAssertion,
    substrings              [1] SEQUENCE {
        type                        ATTRIBUTE.&id ({AttributeTable}),
        strings                     SEQUENCE OF CHOICE {
            initial                     [0] ATTRIBUTE.&Type ({AttributeTable} {@substrings.type}),
            any                         [1] ATTRIBUTE.&Type ({AttributeTable} {@substrings.type}),
            final                       [2] ATTRIBUTE.&Type ({AttributeTable} {@substrings.type}) } },
    greater-or-equal        [2] AttributeValueAssertion,
    less-or-equal           [3] AttributeValueAssertion,
    present                 [4] ATTRIBUTE.&id ({AttributeTable}),
    approximate-match       [5] AttributeValueAssertion,
                            -- 1994 extension --
    other-match             [6] MatchingRuleAssertion }

MatchingRuleAssertion ::= SEQUENCE {
    matching-rule           [0] MATCHING-RULE.&id ({MatchingRuleTable}),
    attribute-type          [1] ATTRIBUTE.&id,
    match-value             [2] MATCHING-RULE.&AssertionType ({MatchingRuleTable} {@matching-rule}) }
```

Every **filter-item** includes an attribute-type which identifies the attribute under test.

An assertion about the value of an attribute can be evaluated only if properly specified. If the MS does not support the attribute-type (or it is not subscribed to by the MS-user), or the MS does not support the matching-rule, or the presented value does not conform to the matching-rule's assertion-syntax, or the definition of the attribute-type does not include the type of match requested, then the abstract-operation or auto-action which contains the assertion shall fail; the abstract-operation shall return an attribute-error.

NOTE – In the case where the presented attribute-syntax or assertion-syntax is invalid, this may be detected by the Presentation Service and cause the rejection of the protocol data unit.

Assertions about the values of an attribute are evaluated using the generic matching-rules (**equality, substrings, greater-or-equal, less-or-equal,** and **approximate-match**), or any other matching-rules defined for that attribute-type. The **present** match is supported for all attribute-types. A matching-rule used as a generic matching-rule is constrained to have an assertion-syntax compatible with the syntax of the corresponding component of **filter-item.** A matching-rule used as an other-match is not constrained in this way. A matching-rule may be used as a generic matching-rule for one attribute-type and as an other-match for a different attribute-type.

All the matching-rules defined in this Service Definition have an outcome of *true* or *false*. However, matching-rules defined elsewhere, may also allow the outcome *undefined*.

If the **filter-item** is properly specified, then where it asserts:

a) **equality**, it is *true* if and only if there is a value of the attribute for which the attribute's equality matching-rule applied to that value and the presented value returns *true*;

b) **substrings**, it is *true* if and only if there is a value of the attribute in which the attribute's **substring** matching-rule applied to that value and the presented value in **strings** returns *true*. The meaning of the components of **strings** is defined in the appropriate matching-rule definition; see 12.2.3 for an example of a substrings matching-rule.

c) **greater-or-equal**, it is *true* if and only if there is a value of the attribute for which the attribute's **ordering** matching-rule applied to that value and the presented value returns *false*, i.e. there is a value of the attribute which is *greater than or equal to* the presented value;

d) **less-or-equal**, it is *true* if and only if there is a value of the attribute for which either the attribute's **equality** matching-rule or **ordering** matching-rule applied to that value and the presented value returns *true*, i.e. there is a value of the attribute which is *less than or equal to* the presented value;

e) **present**, it is *true* if and only if such an attribute is present in the entry;

f) **approximate-match**, it is *true* if and only if there is a value of the attribute for which the **approximate-match** matching-rule applied to that value and the presented value returns *true*. The approximate-match matching-rule may provide phonetic matching to accommodate spelling variations, and is locally defined; there are no specific guide-lines for approximate matching in this Service Definition. If **approximate-match** is not supported, this filter-item shall be treated as a match for **equality**;

g) **other-match**, it is *true* if and only if there is a value of the attribute with the indicated attribute-type for which the matching-rule, applied to that value and the presented-value, returns *true*. If the matching-rule is one of those specified as an other-matching-rule in the definition of the specified attribute-type, and is supported by the MS, then matching proceeds as specified in the matching-rule definition; otherwise an attribute-error of inappropriate-matching may result. As a local matter, an MS may support matching-rules in addition to those defined in this Service Definition. The constraining set (MatchingRuleTable) is defined in clause 12.

8.1.2.3 Attribute-value-assertion

An **attribute-value-assertion** is a proposition, which may be *true, false,* or *undefined,* concerning the values of an entry. It is evaluated using a matching-rule specified for the attribute-type, and which is appropriate for the context in which the attribute-value-assertion is evaluated. It involves an attribute-type and an attribute-value:

```
AttributeValueAssertion ::= SEQUENCE {
    attribute-type      ATTRIBUTE.&id ({AttributeTable}),
    attribute-value     ATTRIBUTE.&Type ({AttributeTable} {@attribute-type}) }
```

If the attribute-value-assertion is properly specified (see 8.1.2.2) its evaluation has one of the following outcomes:

a) *true*, if the entry contains an attribute of that attribute-type, one of whose attribute-values matches the presented attribute-value;

b) *undefined*, if the entry contains an attribute of that attribute-type, and the specified matching-rule declares the result of the match to be undefined;

c) *false*, otherwise.

NOTE – None of the matching-rules defined in this Service Definition returns *undefined*. However, matching-rules defined in other Specifications for given content-types may do so.

8.1.3 Selector

A **selector** parameter is used to select entries of a specified entry-class. The selection operates in three stages. Firstly, the total set of entries of the entry-class may be restricted to a particular contiguous set by specifying its range. Secondly, entries from within this set may be selected by specifying a filter which the selected entries shall satisfy. Thirdly, a limit may be placed on the number of entries thus selected. If the range is defined in ascending order (or is omitted), those entries with the lowest sequence-numbers are selected; if the range is defined in descending order, those entries with the highest sequence numbers are selected (see 8.1.1). Abstract-operations are applied to selected entries according to their

range order, i.e. where the range is defined in ascending order, the selected entry with the lowest sequence-number is operated on first.

```
Selector ::= SET {
    child-entries   [0] BOOLEAN DEFAULT FALSE,
    range           [1] Range OPTIONAL -- default is unbounded --,
    filter          [2] Filter OPTIONAL -- default is all entries within the specified range --,
    limit           [3] INTEGER (1..ub-messages) OPTIONAL,
    override        [4] OverrideRestrictions OPTIONAL
                        -- by default, any fetch-restrictions in force apply -- }
```

The components of **selector** have the following meaning:

a) **Child-entries** (O): If *false*, only main-entries are considered for selection. If *true*, both main-entries and child-entries are considered for selection. In the absence of this component, only main-entries are considered for selection.

b) **Range** (O): The abstract-syntax-notation of range is given in 8.1.1. In the absence of this component, the range is unbounded.

c) **Filter** (O): The abstract-syntax-notation of filter is given in 8.1.2.1. In the absence of this component, all entries within the specified range are selected.

d) **Limit** (O): This allows the specification of an upper limit to the number of entries selected. In the absence of this component, there is no limit to the number of entries selected.

NOTE – The primary role of the limit is to protect against huge results from an abstract-operation as a consequence of badly formulated selections. It may also be used to retrieve an exact number of information-sets to fit a particular output device.

e) **Override** (O): If an override of any of the **fetch-restrictions** is required, the corresponding component(s) of **override-restrictions** shall be present.

```
OverrideRestrictions ::= BIT STRING {
    override-content-types-restriction          (0),
    override-EITs-restriction                   (1),
    override-attribute-length-restriction       (2) } (SIZE (1..ub-restrictions))
```

The values of **override-restrictions** have the following meaning:

1) **Override-content-types-restriction** (M): This bit shall be set to 1 if the **fetch-restrictions** on content-types are to be overridden. If this bit is set to 0, the **fetch-restrictions** on content-types as specified in the MS-bind-argument shall apply.

2) **Override-EITs-restriction** (M): This bit shall be set to 1 if the **fetch-restrictions** on encoded-information-types are to be overridden. If this bit is set to 0, the **fetch-restrictions** on encoded-information-types as specified in the MS-bind-argument shall apply.

3) **Override-attribute-length-restriction** (M): This bit shall be set to 1 if the **fetch-restrictions** on attribute length are to be overridden. If this bit is set to 0, the **fetch-restrictions** on attribute length as specified in the MS-bind-argument shall apply.

In the absence of **override-restrictions**, all the **fetch-restrictions** specified in the MS-bind-argument shall apply.

8.1.4 Entry-information-selection

An **entry-information-selection** parameter indicates what information from an entry is being requested. An empty set indicates that information about the entry itself, rather than the attributes of the entry, is being requested.

```
EntryInformationSelection ::= SET SIZE (0..ub-per-entry) OF AttributeSelection

AttributeSelection ::= SET {
    type    ATTRIBUTE.&id ({AttributeTable}),
    from    [0] INTEGER (1..ub-attribute-values) OPTIONAL -- used if type is multi-valued --,
    count   [1] INTEGER (1..ub-attribute-values) OPTIONAL -- used if type is multi-valued -- }
```

The components of **attribute-selection** have the following meaning:

a) **Type** (M): This indicates the attribute-type of the attribute. If an attribute of that type is not present in the entry, then no values are returned for the attribute. If the attribute-type is not supported by the MS or is not subscribed to by the MS-user, then an attribute-error shall be generated.

b) **From** (O): When an attribute is multi-valued, this Integer gives the relative position of the first value to be returned. If it specifies a value beyond those present in the attribute, no values are returned. This component shall not be present unless the attribute-type is multi-valued. If this component is omitted, values starting at the first value are returned.

c) **Count** (O): When an attribute is multi-valued, this Integer gives the maximum number of values to be returned. It is not an error if **count** is greater than the number of values present in the entry. This component shall not be present unless the attribute-type is multi-valued. If this component is omitted, there is no limit as to how many values are returned.

Where either **from** or **count** (or both) are specified, the attribute-selection constitutes a **partial-attribute-request**.

Where an attribute of the type specified in an attribute-selection is supported by the MS, but is absent from a particular entry, or is not applicable to an entry of that entry-type, then that attribute-selection shall be regarded as absent when applying the associated abstract-operation to the entry.

8.1.5 Entry-information

An **entry-information** parameter conveys selected information from an entry.

```
EntryInformation ::= SEQUENCE {
    sequence-number          SequenceNumber,
    attributes               SET SIZE (1..ub-per-entry) OF Attribute OPTIONAL,
                             -- 1994 extension --
    value-count-exceeded     [0] SET SIZE (1..ub-per-entry) OF AttributeValueCount OPTIONAL }

AttributeValueCount ::= SEQUENCE {
    type     [0]     ATTRIBUTE.&id ({AttributeTable}),
    total    [1]     INTEGER }
```

The components of **entry-information** have the following meaning:

a) **Sequence-number** (M): The sequence-number identifying the entry (see 6.3.2).

b) **Attributes** (C): The set of selected attributes from the entry. Where the corresponding attribute-selection constitutes a partial-attribute-request, only the requested subset of the attribute-values actually present in that attribute are returned. This component is absent if the requested-attributes parameter was omitted from the list-argument or fetch-argument, and no (UA-specific or general) list-attribute-defaults or fetch-attribute-defaults are in force. The component is also absent if none of the requested-attributes are present in the entry.

c) **Value-count-exceeded** (C): This component is present where, in the performance of the List or Fetch abstract-operation, **count** is specified in the attribute-selection, and the specified attribute contains values beyond the limit set by count. The **type** indicates the attribute-type concerned, and **total** indicates the total number of values possessed by the attribute. One value of the component is present for each attribute-selection which exceeds the limit set by count. The component is absent if none of the attribute-selections exceeded their count limits.

8.1.6 MS-submission-options

The **MS-submission-options** parameter is used to request the creation of an entry in a specified entry-class, containing attributes derived from the argument and result of the associated MS-message-submission or MS-probe-submission abstract-operation. Other components of MS-submission-options control the assignment of further attributes to the created entry.

```
MSSubmissionOptions ::= SET {
    object-entry-class           [0] EntryClass (submission | submission-log | draft) OPTIONAL,
    disable-auto-modify          [1] BOOLEAN DEFAULT FALSE,
    add-message-group-names      [2] SET SIZE (1..ub-message-groups) OF MessageGroupName OPTIONAL,
    ms-submission-extensions     [3] MSExtensions OPTIONAL }
```

The components of **MS-submission-options** have the following meaning:

a) **Object-entry-class** (O): This specifies the entry-class of the entry to be created, that will contain details of the associated message or probe submission. Three values are defined:

1) **Submission**. The message or probe shall be submitted to the MTS and an entry created in the Submission, and, subject to subscription, the Submission-log entry-classes.

2) **Submission-log**. The message or probe shall be submitted to the MTS and an entry created in the Submission-log entry-class only.

3) **Draft**. An entry shall be created in the Draft entry-class but the message shall not be submitted to the MTS. This option is not available for MS-probe-submission.

If the specified object-entry-class is not subscribed to then an error is generated. If the **object-entry-class** is omitted, and the Submission-log entry-class is subscribed to, the component defaults to **Submission-log**. If the **object-entry-class** is omitted, and the Submission-log entry-class is not subscribed to, then the message or probe shall be submitted to the MTS and no storage shall take place.

b) **Disable-Auto-modify** (O): This specifies whether the currently registered Auto-modify auto-actions shall be applied to the entry of the Submission, or Submission-log, or Draft entry-class, which may be created as a result of performing the associated message or probe submission. This component shall be omitted if no entry is to be created. In the absence of this component the currently registered Auto-modify auto-actions shall be applied to the new entry.

c) **Add-message-group-names** (O): This specifies one or more message-group-name attribute-values which shall be assigned to the entry of the Submission, or Submission-log, or Draft entry-class, which may be created as a result of performing the associated message or probe submission. This attribute-type is defined in 11.2.33. This component shall be omitted if no entry is to be created. In the absence of this component no message-group-name attribute is attached to the new entry.

NOTE – Message-group-names assigned to an entry by the performance of the Auto-modify auto-action are not affected by the setting of this component.

d) **MS-submission-extensions** (O): This component allows for future general and content-specific extensions to MS-submission-options. No extensions are defined in this Service Definition. The Specification for a given content-type defines its use of this component. In the absence of the component no MS-submission-extensions are specified.

8.1.7 Common-submission-results

The **common-submission-results** parameter contains information common to the results of the MS-message-submission and MS-probe-submission abstract-operations.

```
CommonSubmissionResults ::= SET {
    created-entry                   [0] SequenceNumber OPTIONAL,
    auto-action-error-indication    [1] AutoActionErrorIndication OPTIONAL,
    ms-submission-result-extensions [2] MSExtensions OPTIONAL }
```

The components of **common-submission-results** have the following meaning:

a) **Created-entry** (C): This shall be present if the MS-user requested storage of a submitted message or probe in the Submission or Submission-log entry-classes, or storage of a draft-message entry in the Draft entry-class. It indicates the sequence-number of the newly created entry.

b) **Auto-action-error-indication** (C): This shall be present if an auto-action performed by the MS concerning the submitted-message or probe, or stored draft-message, caused the generation of an auto-action-error. If the Auto-action-log entry-class is subscribed to, this parameter indicates the sequence-number of the first entry which records these unsuccessful auto-action executions. Otherwise it contains a Null (see 7.1.2, item j). The parameter shall be absent if no auto-action-errors occurred.

c) **MS-submission-result-extensions** (C): This component allows for future general and content-specific extensions to common-submission-results. No extensions are defined in this Service Definition. It shall be present if the MS-user specified MS-submission-extensions in the submission-options parameter of the MS-message-submission or MS-probe-submission argument, and those extensions caused the generation of this corresponding result.

8.2 Retrieval Port abstract-operations

The following abstract-operations are available at the Retrieval Port:

a) Summarize;

b) List;

c) Fetch;

d) Delete;

e) Register-MS;

f) Alert;

g) Modify.

Abstract-operations are performed asynchronously subject to the following conditions:

– the Delete, Register-MS, and Modify abstract-operations shall not be performed until all outstanding abstract-operations have been completed;

– the Delete, Register-MS, and Modify abstract-operations are performed in the order in which they are invoked and are allowed to complete before any other abstract-operation is performed.

Because of this, and the fact that the List and Fetch abstract-operations may change the retrieval-status of an entry, the results of the Summarize, List and Fetch abstract-operations may be non-deterministic.

8.2.1 Summarize abstract-operation

The **Summarize** abstract-operation returns summary counts of selected entries of a specified entry-class. In addition to these summaries, a count of the entries selected and their lowest and highest sequence-numbers is also returned. Zero or more individual summaries may be requested.

The Summarize abstract-operation will be successful only when access to the entry-class is permitted according to the security-context and the security-policy in force.

The attributes that may be requested for summaries are restricted. For the general-attributes of entries of the various entry classes, the restrictions are defined in Tables 2 and 3. For content-specific attributes, the attribute definition in the relevant Specification shall state whether the attribute is available for Summarize.

```
summarize ABSTRACT-OPERATION ::= {
    ARGUMENT        SummarizeArgument
    RESULT          SummarizeResult
    ERRORS          {attribute-error | invalid-parameters-error | range-error |
                    security-error | sequence-number-error | service-error,
                    ... -- 1994 extension additions --,
                    entry-class-error | ms-extension-error}
    CODE            op-summarize }
```

NOTE – An example of the Summarize abstract-operation is given in Annex I.

8.2.1.1 Summarize-argument

```
SummarizeArgument ::= SET {
    entry-class         [0] EntryClass DEFAULT delivery,
    selector            [1] Selector,
    summary-requests    [2] SEQUENCE SIZE (1..ub-summaries) OF ATTRIBUTE.&id ({AttributeTable})
                            OPTIONAL -- absent if no summaries are requested --,
                        -- 1994 extension --
    summarize-extensions [3] MSExtensions OPTIONAL }
```

The parameters of **summarize-argument** have the following meaning:

a) **Entry-class** (O): This specifies which entry-class is addressed by the abstract-operation (see 6.3.1).

b) **Selector** (M): This is a set of selection criteria to determine which entries shall be summarized (see 8.1.3).

c) **Summary-requests** (O): This is the sequence of attribute-types for which summaries are requested. This parameter is present only if a summary is requested.

d) **Summarize-extensions** (O): This parameter allows for future general and content-specific extensions to summarize-argument. No extensions are defined in this Service Definition.

8.2.1.2 Summarize-result

Should the request succeed, the **summarize-result** will be returned.

```
SummarizeResult ::= SET {
    next                        [0] SequenceNumber OPTIONAL,
    count                       [1] INTEGER (0..ub-messages) -- of the entries selected --,
    span                        [2] Span OPTIONAL -- of the entries selected,
                                    -- omitted if count is zero --,
    summaries                   [3] SEQUENCE SIZE (1..ub-summaries) OF Summary OPTIONAL,
                                -- 1994 extension --
    summarize-result-extensions [4] MSExtensions OPTIONAL }
```

The parameters of **summarize-result** have the following meaning:

a) **Next** (C): This is returned in the case where the number of entries selected would have been greater if it were not for the limit specified in the selector. The parameter contains the sequence-number of the next entry that would have been selected, as determined by the range order which determines the direction of searching (see 8.1.3).

b) **Count** (M): This is an integer giving the count of entries that match the selection criteria.

c) **Span** (C): This contains the lowest and highest sequence-numbers of the entries that match the selection criteria. It is absent if there are no such entries.

```
Span ::= SEQUENCE {
    lowest      [0] SequenceNumber,
    highest     [1] SequenceNumber }
```

The components of **span** have the following meaning:

1) **Lowest** (M): Identifies the entry with the lowest sequence-number which matches the selection criteria.

2) **Highest** (M): Identifies the entry with the highest sequence-number which matches the selection criteria.

NOTE – The values of **lowest** and **highest** are not affected by the order of the range (ascending or descending); see 8.1.1.

d) **Summaries** (C): One **summary** is returned for each summary-request. The **summaries** are returned in the order that they were requested.

```
Summary ::= SET {
    absent  [0] INTEGER (1..ub-messages) OPTIONAL -- count of entries where attribute is absent --,
    present [1] SET SIZE (1..ub-attribute-values) OF -- one for each attribute value present --
                SEQUENCE {
                    type    ATTRIBUTE.&id ({AttributeTable}),
                    value   ATTRIBUTE.&Type ({AttributeTable} {@.type}),
                    count   INTEGER (1..ub-messages)} OPTIONAL }
```

The components of **summary** have the following meaning:

1) **Absent** (C): A count of the entries that do not contain an attribute of the attribute-type specified in the request. The component is omitted if there are no such entries.

2) **Present** (C): A summary of the entries that contain an attribute of the attribute-type specified, broken down by the attribute-values actually present. The component is omitted if there are no such entries.

The components of **present** have the following meaning:

i) **Type** (M): The type of the attribute.

ii) **Value** (M): The attribute-value for which the count is given.

iii) **Count** (M): A count of entries with this attribute-value.

e) **Summarize-result-extensions** (C): This parameter allows for future general and content-specific extensions to summarize-result. No extensions are defined in this Service Definition.

8.2.1.3 Summarize abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in clause 9. The attribute-error abstract-error shall be reported both in the case of an improperly specified filter, and where the summary-requests parameter specifies an attribute-type which is not supported by the MS, or is not subscribed to by the MS-user.

8.2.2 List abstract-operation

The **List** abstract-operation is used to search a specified entry-class for entries of interest, and to return selected information from those entries.

The List abstract-operation will be successful only when access to the entry-class is permitted according to the security-context and the security-policy in force.

The information that may be selected for entries of a given entry-class may be restricted. For the general-attributes of entries of the various entry-classes, the restrictions are defined in Tables 2 and 3.

```
list ABSTRACT-OPERATION ::= {
    ARGUMENT        ListArgument
    RESULT          ListResult
    ERRORS          {attribute-error | invalid-parameters-error | range-error |
                    security-error | sequence-number-error | service-error,
                    ... -- 1994 extension additions --,
                    entry-class-error | ms-extension-error}
    CODE            op-list }
```

8.2.2.1 List-argument

```
ListArgument ::= SET {
    entry-class             [0] EntryClass DEFAULT delivery,
    selector                [1] Selector,
    requested-attributes    [3] EntryInformationSelection OPTIONAL,
                            -- 1994 extension --
    list-extensions         [4] MSExtensions OPTIONAL }
```

The parameters of **list-argument** have the following meaning:

a) **Entry-class** (O): This specifies which entry-class is addressed by the abstract-operation (see 6.3.1).

b) **Selector** (M): This is a set of selection criteria to determine which entries shall be returned (see 8.1.3).

c) **Requested-attributes** (O): This indicates what information from the selected entries is to be returned in the result (see 8.1.4). If this parameter is absent, the registered set of UA-list-attribute-defaults is used. If no UA-registration was specified when the abstract-association was established, or no UA-list-attribute-defaults are registered for the currently active UA-registration, then the general (non UA-specific) list-attribute-defaults are used (see 8.2.5.1, items c and g).

d) **List-extensions** (O): This parameter allows for future general and content-specific extensions to list-argument. No extensions are defined in this Service Definition.

8.2.2.2 List-result

Should the request succeed, the **list-result** will be returned.

```
ListResult ::= SET {
    next                    [0] SequenceNumber OPTIONAL,
    requested               [1] SEQUENCE SIZE (1..ub-messages) OF EntryInformation OPTIONAL
                                -- omitted if none found --,
                            -- 1994 extension --
    list-result-extensions  [2] MSExtensions OPTIONAL }
```

The parameters of **list-result** have the following meaning:

a) **Next** (C): This is returned in the case where the number of entries selected would have been greater if it were not for the limit specified in the selector. The parameter contains the sequence-number of the next entry that would have been selected, as determined by the range order which determines the direction of searching (see 8.1.3).

b) **Requested** (C): This conveys the requested entry-information (see 8.1.5) from each selected entry. Each item of entry-information appears in the same order (ascending or descending) as the range order (see 8.1.3). The parameter is absent if no entries were selected.

c) **List-result-extensions** (C): This parameter allows for future general and content-specific extensions to list-result. No extensions are defined in this Service Definition.

8.2.2.3 List abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in clause 9.

8.2.3 Fetch abstract-operation

The **Fetch** abstract-operation is used to return selected information from a specific entry of some entry-class. Alternatively, it is used to return selected information from the first entry from among several entries of interest; in this case the sequence-numbers of the other selected entries are also returned. The Fetch abstract-operation will be successful only when access to the entry-class is permitted according to the security-context and the security-policy in force.

Information from an entry may be fetched several times, until the entry is explicitly deleted using the Delete abstract-operation.

```
fetch ABSTRACT-OPERATION ::= {
    ARGUMENT            FetchArgument
    RESULT              FetchResult
    ERRORS              {attribute-error | fetch-restriction-error | invalid-parameters-error |
                         range-error | security-error | sequence-number-error | service-error,
                        ... -- 1994 extension additions --,
                        entry-class-error | ms-extension-error}
    CODE                op-fetch }
```

8.2.3.1 Fetch-argument

```
FetchArgument ::= SET {
    entry-class             [0] EntryClass DEFAULT delivery,
    item                    CHOICE {
                        search      [1] Selector,
                        precise     [2] SequenceNumber},
    requested-attributes    [3] EntryInformationSelection OPTIONAL,
                            -- 1994 extension --
    fetch-extensions        [4] MSExtensions OPTIONAL }
```

The parameters of **fetch-argument** have the following meaning:

a) **Entry-class** (O): This specifies which entry-class is addressed by the abstract-operation (see 6.3.1).

b) **Item** (M): One of the components described below shall be specified in order to determine which entry to fetch.

1) **Search** (C): This is a selector specifying a set of entries. Of the entries selected (if any) the entry fetched shall be the one with the highest sequence number if the range is defined in descending order, and that with the lowest sequence-number otherwise (see 8.1.3).

2) **Precise** (C): This is the sequence-number of the entry to be fetched (see 6.3.2).

c) **Requested-attributes** (O): This indicates what information from the selected entry is to be returned in the result (see 8.1.4). If this parameter is absent, the registered set of UA-fetch-attribute-defaults is used. If no UA-registration was specified when the abstract-operation was established, or the currently active UA-registration does not specify UA-fetch-attribute-defaults, then the general (non UA-specific) fetch-attribute-defaults are used (see 8.2.5.1, items d and g).

d) **Fetch-extensions** (O): This parameter allows for future general and content-specific extensions to fetch-argument. No extensions are defined in this Service Definition.

8.2.3.2 Fetch-result

Should the request succeed, the **fetch-result** will be returned.

```
FetchResult ::= SET {
    entry-information          [0] EntryInformation OPTIONAL -- if an entry was selected --,
    list                       [1] SEQUENCE SIZE (1..ub-messages) OF SequenceNumber OPTIONAL,
    next                       [2] SequenceNumber OPTIONAL,
                               -- 1994 extension --
    fetch-result-extensions    [3] MSExtensions OPTIONAL }
```

The parameters of **fetch-result** have the following meaning:

a) **Entry-information** (C): This is the set of all those requested attributes that are present in the selected entry (see 8.1.5). It is not present in the case that a search was performed and no entry was selected.

b) **List** (C): This is returned in the case that more than one entry matched the search selector. The list gives the sequence-numbers of these further entries in the same order (ascending or descending) as the range order (see 8.1.3).

c) **Next** (C): This is returned in the case where the number of entries selected would have been greater if it were not for the limit specified in the selector. The parameter contains the sequence-number for the next entry that would have been selected, as determined by the range order which determines the direction of searching.

d) **Fetch-result-extensions** (C): This parameter allows for future general and content-specific extensions to fetch-result. No extensions are defined in this Service Definition.

8.2.3.3 Fetch abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in clause 9.

8.2.4 Delete abstract-operation

The **Delete** abstract-operation is used to delete selected entries of a specified entry-class. A main-entry and all its dependent child-entries shall only be deleted together. This is achieved by specifying just the main-entry as an argument. The Delete abstract-operation will be successful only when the entry-class permits deletion, and this is allowed by the security-context and the security-policy in force.

Certain entries are not subject to deletion. An attempt to delete a child-entry shall result in a delete-error of *child-entry-specified*. An attempt to delete an entry of the Delivery entry-class which has a retrieval-status of *new* (see 6.3.8) shall result in a delete-error of *new-entry-specified*; the retrieval-status shall not be considered when entries of entry-classes other than Delivery are deleted. The retrieval-status of any child-entry associated with a main-entry shall not be considered when applying the Delete abstract-operation to the main-entry. An attempt to delete an entry of the Message-log entry-class for which a corresponding entry exists in the Stored-message entry-class shall result in a delete-error of *stored-message-exists*.

As a subscription option, the deletion of the entries of any entry-class or entry-classes may be restricted or prohibited, and cause a delete-error of *entry-class-restriction*. Further restrictions may be defined for entries of a given content-type as indicated in the Specification which defines that content-type. An attempt to violate such a delete restriction causes the generation of a delete-error of *delete-restriction-problem*.

NOTE – Implementations may choose to prohibit the deletion of entries of the Message-log entry-class except for the oldest entry, or a contiguous sequence that includes the oldest.

```
delete ABSTRACT-OPERATION ::= {
    ARGUMENT    DeleteArgument
    RESULT      DeleteResult
    ERRORS      {deleteError | invalid-parameters-error | range-error | security-error |
                sequence-number-error | service-error,
                ... -- 1994 extension additions --,
                entry-class-error | ms-extension-error}
    CODE        op-delete }
```

8.2.4.1 Delete-argument

```
DeleteArgument ::= SET {
    entry-class             [0]  EntryClass DEFAULT delivery,
    items                   CHOICE {
        selector                    [1]  Selector,
        sequence-numbers            [2]  SET SIZE (1..ub-messages) OF SequenceNumber },
        -- 1994 extension --
    delete-extensions       [3]  MSExtensions OPTIONAL }
```

The parameters of **delete-argument** have the following meaning:

a) **Entry-class** (O): This specifies which entry-class is addressed by the abstract-operation (see 6.3.1).

b) **Items** (M): One of the components described below shall be specified in order to determine which entries to delete.

1) **Selector** (C): See 8.1.3.

2) **Sequence-numbers** (C): An unordered list of **sequence-numbers** (see 6.3.2).

c) **Delete-extensions** (O): This parameter allows for future general and content-specific extensions to delete-argument. No extensions are defined in this Service Definition.

NOTE – The Interpersonal Messaging content-type makes use of delete-extensions as defined in 19.5.3 of ITU-T Rec. X.420 | ISO/IEC 10021-7.

8.2.4.2 Delete-result

Should the request succeed, the **delete-result** shall be returned:

```
DeleteResult ::= CHOICE {
    delete-result-88        NULL,
                            -- 1994 extension --
    delete-result-94        SET {
        entries-deleted             [0]  SEQUENCE SIZE (1..ub-messages) OF SequenceNumber OPTIONAL,
        delete-result-extensions    [1]  MSExtensions OPTIONAL } }
```

The parameters of **delete-result** have the following meaning:

a) **Delete-result-88** (C): This parameter is returned if a 1988 Application Context is in use.

b) **Delete-result-94** (C): This parameter is returned if a 1994 Application Context is in use. It contains the following components:

1) **Entries-deleted** (C): This identifies the entries deleted. It is present if the selector component was present in the delete-argument, and at least one entry has been deleted. It is absent otherwise.

2) **Delete-result-extensions** (C): This component allows for future general and content-specific extensions to delete-result. No extensions are defined in this Service Definition.

8.2.4.3 Delete abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in clause 9.

8.2.5 Register-MS abstract-operation

The **Register-MS** abstract-operation is used to register or deregister various information with the MS, and retrieve registered information from it:

a) auto-actions;

b) default requested attribute-types for List and Fetch;

c) credentials;

d) user-security-labels;

e) UA registrations;

f) submission defaults;

g) message-group-names.

Where an MS supports both a 1988 and 1994 Application Context for some MS-user, then registrations made using one Application Context shall be effective when the other Application Context is in use.

NOTE – For example, the registered general submission-defaults registered when using a 1994 Application Context will apply to submissions invoked in an abstract-association established using a 1988 Application Context.

```
register-MS ABSTRACT-OPERATION ::= {
    ARGUMENT        Register-MSArgument
    RESULT          Register-MSResult
    ERRORS          {attribute-error | auto-action-request-error | invalid-parameters-error |
                    security-error | service-error | old-credentials-incorrectly-specified |
                    new-credentials-unacceptable,
                    ... -- 1994 extension additions --,
                    message-group-error | ms-extension-error | register-ms-error}
    CODE            op-register-ms }
```

8.2.5.1 Register-MS-argument

```
Register-MSArgument ::= SET {
    auto-action-registrations       [0] SET SIZE (1..ub-auto-registrations) OF AutoActionRegistration
                                        OPTIONAL,
    auto-action-deregistrations     [1] SET SIZE (1..ub-auto-registrations) OF AutoActionDeregistration
                                        OPTIONAL,
    list-attribute-defaults         [2] SET SIZE (0..ub-default-registrations) OF ATTRIBUTE.&id
                                        ({AttributeTable}) OPTIONAL,
    fetch-attribute-defaults        [3] SET SIZE (0..ub-default-registrations) OF ATTRIBUTE.&id
                                        ({AttributeTable}) OPTIONAL,
    change-credentials              [4] SEQUENCE {
        old-credentials                     [0] Credentials,
        new-credentials                     [1] Credentials } OPTIONAL,
    user-security-labels            [5] SET SIZE (1..ub-labels-and-redirections) OF SecurityLabel OPTIONAL,
                                    -- 1994 extension --
    ua-registrations                [6] SET SIZE (1..ub-ua-registrations) OF UARegistration OPTIONAL,
    submission-defaults             [7] MSSubmissionOptions OPTIONAL,
    message-group-registrations     [8] MessageGroupRegistrations OPTIONAL,
    registration-status-request     [9] RegistrationTypes OPTIONAL,

    register-ms-extensions          [10] MSExtensions OPTIONAL }
```

The parameters of **register-MS-argument** have the following meaning:

a) **Auto-action-registrations** (O): This is a set of **auto-action-registration** (see 6.5.2), one for each auto-action to be registered. The new auto-action registration-parameter replaces any previously registered auto-action (if any) with that registration-identifier and auto-action-type.

In the absence of auto-action-registrations, no new auto-actions are registered.

b) **Auto-action-deregistrations** (O): This is a set of **auto-action-deregistration**, one for each auto-action to be deregistered. Any registered auto-action with registration-identifier and auto-action-type matching those in an auto-action-deregistration is deregistered.

```
AutoActionDeregistration ::= SEQUENCE {
    auto-action-type            AUTO-ACTION.&id ({AutoActionTable}),
    registration-identifier     [0] INTEGER (1..ub-per-auto-action) DEFAULT 1 }
```

In the absence of auto-action-deregistrations, the existing set of registered auto-actions remains unchanged, except where updated by auto-action-registrations.

c) **List-attribute-defaults** (O): This specifies the types of the attributes which shall be returned in any subsequently invoked List abstract-operation if the requested-attributes argument is absent, and either no UA-registration was specified when the abstract-association was established, or the currently active UA-registration does not specify UA-list-attribute-defaults.

In the absence of list-attribute-defaults, there is no change to the registered default (if any). The list-attribute-defaults are the empty set until explicitly changed by the MS-user via the Register-MS abstract-operation.

NOTE 1 – In versions of this Service Definition published prior to 1994, the Set had a lower bound of one.

d) **Fetch-attribute-defaults** (O): This specifies the types of the attributes which shall be returned in any subsequently invoked Fetch abstract-operation if the requested-attributes argument is absent, and either no UA-registration was specified when the abstract-association was established, or the currently active UA-registration does not specify UA-fetch-attribute-defaults.

In the absence of fetch-attribute-defaults, there is no change to the registered default (if any). The fetch-attribute-defaults are the empty set until explicitly changed by the MS-user via the Register-MS abstract-operation.

NOTE 2 – In versions of this Service Definition published prior to 1994, the Set had a lower bound of one.

e) **Change-credentials** (O): This specifies a change in the credentials used in MS-bind to authenticate the identity of the MS-user to the MS.

The **old-credentials** are the current credentials of the MS-user, and the **new-credentials** are the credentials proposed for future use in MS-bind.

In the absence of this parameter, the previously registered credentials remain unchanged.

The credentials of the MS-user may differ in value from the **initiator-credentials** detailed in 8.1.1.1.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4. The credentials used between the MTS-user (the MS) and the MTS are maintained separately and may be changed by use of the Administration Port's Register abstract-operation.

f) **User-security-labels** (O): This contains the **security-label(s)** of the MS abstract-service-user, if they are to be changed. User-security-labels is defined in 8.4.1.1.1.7 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

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

NOTE 3 – Some security-policies may permit the user-security-labels to be changed in this way only if a secure link is employed. Other local means of changing the user-security-labels in a secure manner may be provided.

g) **UA-registrations** (O): This specifies one or more amendments to the set of UA-registrations. If the MS-bind-argument of the present or a subsequently established abstract-association identifies a UA-registration that was registered by means of this parameter, then the defaults specified below, associated with that registration, shall apply for the duration of that abstract-association (see 7.1.1, item f).

NOTE 4 – On different occasions a user may employ different UAs to communicate with the MS.

```
UARegistration ::= SET {
    ua-registration-identifier      [0]  RegistrationIdentifier,
    ua-list-attribute-defaults      [1]  SET SIZE (0..ub-default-registrations) OF ATTRIBUTE.&id
                                             ({AttributeTable}) OPTIONAL,
    ua-fetch-attribute-defaults     [2]  SET SIZE (0..ub-default-registrations) OF ATTRIBUTE.&id
                                             ({AttributeTable}) OPTIONAL,
    ua-submission-defaults          [3]  MSSubmissionOptions OPTIONAL,
    content-specific-defaults       [4]  MSExtensions OPTIONAL }
```

The components of **UA-registration** are as follows:

1) **UA-registration-identifier** (M): An identifier for this UA-registration. If the same value of UA-registration-identifier was specified in a previous invocation of this abstract-operation, then the components present in the new registration replace the corresponding components of the previous registration; otherwise a new UA-registration is created. If the value of UA-registration-identifier matches that optionally present in the MS-bind-argument which established the current abstract-association, then the new UA-registration takes immediate effect.

If the UA-registration-identifier is the only component present in a UA-registration, then no UA-registration is recorded and any existing UA-registration bearing this UA-registration-identifier is deleted.

NOTE 5 – A UA-registration is likely to contain values influenced both by the design of the UA and by user controlled options. It may be desirable for the UA-registration-identifier to be made configurable by the user rather than fixed in the UA implementation.

2) **UA-list-attribute-defaults** (O): This specifies the types of the attributes which shall be returned in a subsequently performed List abstract-operation if the entry-information-selection parameter is absent. In the absence of this component, there is no change to the UA-list-attribute-defaults defined for this UA-registration (if any).

3) **UA-fetch-attribute-defaults** (O): This specifies the types of the attributes which shall be returned in a subsequently performed Fetch abstract-operation if the entry-information-selection parameter is absent. In the absence of this component, there is no change to the UA-fetch-attribute-defaults defined for this UA-registration (if any).

4) **UA-submission-defaults**(O): This specifies the submission-options parameter which shall be used in a subsequently performed MS-message-submission or MS-probe-submission abstract-operation when the submission-options parameter is absent. The value *draft* is not permitted for the object-entry-class component (see 8.1.6). In the absence of this component, there is no change to the UA-submission-defaults defined for this UA-registration (if any).

5) **Content-specific-defaults** (O): This specifies registrations specific to a particular content-type. The definition of a content-specific default appears in the Specification for the content-type concerned. In the absence of this component, there is no change to the content-specific-defaults defined for this UA-registration (if any).

Any existing UA-registration omitted from UA-registrations remains unaltered. In the absence of UA-registrations, no change is made to existing registrations.

h) **Submission-defaults** (O): If the submission-options argument of MS-message-submission or MS-probe-submission is absent when those abstract-operations are invoked, and no UA-registration is specified when the abstract-association is established (or that UA-registration did not specify UA-submission-defaults), or where the MS itself performs submission while processing an auto-action, this parameter supplies defaults for the submission-options argument. The initial state of the parameter, which is described in 8.1.6, is that all its components are absent and assume their default values. These values persist until changed by an invocation of the Register-MS abstract-operation. The value *draft* is not permitted for the object-entry-class component. In the absence of this parameter, the value of submission-defaults is unchanged.

i) **Message-group-registrations** (O): This specifies amendments to the set of message-group-names registered by the MS-user which may be employed for the classification and organization of MS entries (see 6.4). An attempt to add a value to an entry's message-group-name attribute shall result in an error unless the specified value was previously registered by means of this parameter.

```
MessageGroupRegistrations ::= SEQUENCE SIZE (1..ub-default-registrations) OF CHOICE {
    register-group          [0] MessageGroupNameAndDescriptor,
    deregister-group        [1] MessageGroupName,
    change-descriptors      [2] MessageGroupNameAndDescriptor }

MessageGroupNameAndDescriptor ::= SET {
    message-group-name      [0] MessageGroupName,
    message-group-descriptor [1] GeneralString (SIZE (1..ub-group-descriptor-length)) OPTIONAL }
```

The components of **message-group-registrations** are as follows:

1) **Register-group** (O): The message-group-name specified is added to the set of registered message-groups. A message-group-name remain registered until explicitly deregistered in a subsequent Register-MS abstract-operation. An error results from an attempt to register a message-group-name which is already registered. Where a specified message-group-name contains more than one group-name-part, an error results unless a registration currently exists for that name less its final group-name-part.

NOTE 6 – Consequently, every message-group-name that contains more than one group-name-part has a parent message-group-name above it in the notional message-group-name hierarchy.

The **message-group-descriptor** is an optional component which provides an informal description of each message-group registered.

2) **Deregister-group** (O): The message-group-name specified is removed from the set of message-group-registrations. A message-group-error is reported if any child message-group-name (i.e. one formed by adding a group-name-part to the specified message-group-name) is currently registered. A message-group-error is also reported if the specified message-group-name is not registered, or is in use, i.e. is assigned to the message-group-name attribute of any entry, or is referenced in a registered Auto-modify auto-action.

3) **Change-descriptors** (O): The supplied value of message-group-descriptor replaces the stored value for the message-group-name specified. A message-group-error is reported if a specified message-group-name is not registered.

The components of message-group-registrations shall be applied in the order presented. In the absence of message-group-registrations, no change is made to the registered set of message-group-names and descriptors.

j) **Registration-status-request** (O): This parameter is used to request information from the MS concerning the current settings of defaults and other registered items. The result returned reflects the state of registered information after all other parameters of register-MS-argument have been processed. The

parameter contains several elements each of which, if set, requests the registration status of the corresponding class of information. In the absence of this parameter, no registration status information is requested.

```
RegistrationTypes ::= SET {
    registrations                [0] BIT STRING {
        auto-action-registrations        (0),
        list-attribute-defaults          (1),
        fetch-attribute-defaults         (2),
        ua-registrations                 (3),
        submission-defaults              (4),
        message-group-registrations      (5) } OPTIONAL,
    extended-registrations       [1] SET OF MS-EXTENSION.&id OPTIONAL,
    restrict-message-groups      [2] MessageGroupsRestriction OPTIONAL }
```

The components of **registration-types** have the following meaning:

1) **Registrations** (O): This specifies the items of registered information requested. One or more of auto-action-registrations, list-attribute-defaults, fetch-attribute-defaults, UA-registrations, submission-defaults, and message-group-registrations may be requested.

2) **Extended-registrations** (O): This specifies those items of extended registered information requested.

3) **Restrict-message-groups** (O): If message-group-registrations is requested, this component specifies restrictions on the information returned.

```
MessageGroupsRestriction ::= SET {
    parent-group                 [0] MessageGroupName OPTIONAL,
    immediate-descendants-only   [1] BOOLEAN DEFAULT TRUE,
    omit-descriptors             [2] BOOLEAN DEFAULT TRUE }
```

If **parent-group** is specified only those registered-message-groups which are children of this parent-group shall be returned, i.e. those message-group-names which are formed by adding one or more group-name-parts to the parent-group name. If **immediate-descendants-only** is *true*, then only those message-group-names formed by adding a single group-name-part to the parent-group are returned; otherwise all descendant message-group-names are returned. If **immediate-descendants-only** is *true* and parent-group is absent, then only those message-group-names which contain a single group-name-part are returned. If **omit-descriptors** is *true*, then message-group-descriptors shall be omitted from the result.

If **restrict-message-groups** is absent (and message-group-registrations is requested), then all registered message-group-names and descriptors are returned. **Restrict-message-groups** shall be omitted if message-group-registrations is not specified.

k) **Register-MS-extensions** (O): This parameter allows for future general and content-specific extensions to register-MS-argument. No extensions are defined in this Service Definition.

8.2.5.2 Register-MS-result

Should the request succeed, the **register-MS-result** shall be returned.

```
Register-MSResult ::= CHOICE {
    no-status-information        NULL,
                                 -- 1994 extension --
    registered-information       SET {
        auto-action-registrations        [0] SET SIZE (1..ub-auto-registrations) OF AutoActionRegistration
                                             OPTIONAL,
        list-attribute-defaults          [1] SET SIZE (1..ub-default-registrations) OF ATTRIBUTE.&id
                                             ({AttributeTable}) OPTIONAL,
        fetch-attribute-defaults         [2] SET SIZE (1..ub-default-registrations) OF ATTRIBUTE.&id
                                             ({AttributeTable}) OPTIONAL,
        ua-registrations                 [3] SET SIZE (1..ub-ua-registrations) OF UARegistration OPTIONAL,
        submission-defaults              [4] MSSubmissionOptions OPTIONAL,
        message-group-registrations      [5] SET SIZE (1..ub-message-groups) OF
                                             MessageGroupNameAndDescriptor OPTIONAL,
        register-ms-result-extensions    [6] MSExtensions OPTIONAL } }
```

The parameters of **register-MS-result** have the following meaning:

a) **No-status-information** (C): Present if registration-status-request was absent from the register-MS-argument.

b) **Registered-information** (C): This contains the information requested in the registration-status-request parameter of the register-MS-argument. Each component of **registered-information** corresponds to the similarly named component of the register-MS-argument. The presence of an element of information is conditional upon the setting of the corresponding component of registration-status-request, and upon the availability of a value for that element. The information conveyed in the result reflects the values of the registered information subsequent to the processing of any registrations or deregistrations requested in the register-MS-argument. The register-MS-result-extensions component allows for future general and content-specific extensions to register-MS-result. No extensions are defined in this Service Definition.

8.2.5.3 Register-MS abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in clause 9. In addition to those abstract-errors which arise directly (e.g. from an invalid change-credentials parameter), the MS shall generate an abstract-error where a registration affecting the subsequent performance of List, Fetch, or an auto-action would cause that abstract-operation or auto-action to fail.

NOTE – This occurs, for example, where list-attribute-defaults include an unavailable attribute-type, or where an auto-action-registration contains a filter which makes reference to an unavailable attribute-type.

8.2.6 Alert abstract-operation

The **Alert** abstract-operation enables the MS to inform the MS-user of the delivery of a message or report, whose attributes match the selection criteria of one of the auto-alert-registration-parameters (see 13.1) previously supplied by means of the Register-MS abstract-operation or by subscription.

The Alert abstract-operation may be invoked only during an existing abstract-association initiated by the MS-user, and only as a result of entries created after the establishment of the abstract-association.

The presence of entries which have been created between abstract-associations, and match the registered selection criteria, shall be indicated in the result of the MS-bind abstract-operation which establishes the next abstract-association. No Alert abstract-operation will be invoked for these entries (see 7.1.2, item d)).

The Alert abstract-operation will be successful only when access to the Delivery entry-class is permitted according to the security-context and the security-policy in force.

```
alert ABSTRACT-OPERATION ::= {
    ARGUMENT        AlertArgument
    RESULT          AlertResult
    ERRORS          {security-error}
    CODE            op-alert }
```

8.2.6.1 Alert-argument

```
AlertArgument ::= SET {
    alert-registration-identifier       [0] INTEGER (1..ub-auto-actions),
    new-entry                           [2] EntryInformation OPTIONAL }
```

The parameters of the **alert-argument** have the following meaning:

a) **Alert-registration-identifier** (M): Indicates the registration-identifier of the registered Auto-alert auto-action which caused the alert (see 6.5.2 and 13.1).

b) **New-entry** (O): This conveys the information from the new entry which was requested in the auto-alert-registration-parameter (see 13.1). It may be absent if the MS-user did not specify requested-attributes in the auto-alert-registration-parameter.

NOTE – Even if the requested-attributes component is not specified in the auto-alert-registration-parameter, new-entry, if present, will indicate the sequence-number of the entry.

8.2.6.2 Alert-result

Should the request succeed, the **alert-result** will be returned.

```
AlertResult ::= NULL
```

8.2.6.3 Alert abstract-errors

Should the request fail, the security error will be reported. This occurs when the abstract-operation violates the security-policy in force.

8.2.7 Modify abstract-operation

The **Modify** abstract-operation enables the MS-user to add or remove attributes, or individual attribute-values associated with one or more MS entries. Only certain general-attribute-types are subject to modification using this abstract-operation (see 11.6). Attribute-types specific to a particular content-type, and subject to modification, are defined in the relevant Specification. The definition of an attribute-type which is subject to modification may specify restrictions for the performance of the Modify abstract-operation. The Modify abstract-operation is not standardized for use with 1988 Application Contexts.

When Modify is invoked, the MS first verifies that the supplied argument is valid; if any errors are found, one of the listed abstract-errors is returned. All static errors, i.e. those which may be detected by examining the modify-argument, shall be detected before any attempt is made to apply the specified modifications to the selected entries; the static errors are summarized at the foot of 8.2.7.1.

If the argument is valid the specified modifications are applied to each of the selected entries in the order specified in the entries argument. All modifications shall be applied to a selected entry before any modifications are applied to the next selected entry. If the modification of an entry cannot be performed, then that entry is restored to its original state and the abstract-operation terminates with a modify-error. However, entries already modified successfully, remain in their modified state and are reported in the modify-error.

```
modify ABSTRACT-OPERATION ::= {
    ARGUMENT        ModifyArgument
    RESULT          ModifyResult
    ERRORS          {attribute-error | invalid-parameters-error | security-error |
                    sequence-number-error | service-error | modify-error | message-group-error |
                    entry-class-error | ms-extension-error,
                    ... -- For future extension additions -- }
    CODE            op-modify }
```

8.2.7.1 Modify-argument

```
ModifyArgument ::= SET {
    entry-class         [0] EntryClass DEFAULT delivery,
    entries             CHOICE {
        selector            [1] Selector,
        specific-entries    [2] SEQUENCE SIZE (1..ub-messages) OF SequenceNumber },
    modifications       [3] SEQUENCE SIZE (1..ub-modifications) OF EntryModification,
    modify-extensions   [4] MSExtensions OPTIONAL }
```

The parameters of **modify-argument** have the following meaning:

a) **Entry-class** (O): This specifies the entry-class which contains the entries to be modified. If an entry-class not subject to modification is specified then an entry-class-error is generated.

b) **Entries** (M): The entries to be modified are identified either by **selector** (see 8.1.3), or explicitly by sequence-number. If the **specific-entries** component is present, the existence of each entry shall be verified before any modifications are applied and a sequence-number-error reported if one or more does not exist. Entries shall be processed in the order specified.

c) **Modifications** (M): This defines a sequence of modifications, which shall be applied to each selected entry in the order specified. The attribute-types present in the parameter shall be examined before any modifications are applied, and an attribute-error reported if an unsupported attribute-type, or one not subject to modification, is specified.

```
EntryModification ::= SET {
    strict                  [0] BOOLEAN DEFAULT FALSE,
    modification CHOICE {
        add-attribute       [1] Attribute,
        remove-attribute    [2] ATTRIBUTE.&id ({AttributeTable}),
        add-values          [3] OrderedAttribute,
        remove-values       [4] OrderedAttribute } }
```

```
OrderedAttribute ::= SEQUENCE {
    attribute-type          ATTRIBUTE.&id ({AttributeTable}),
    attribute-values        SEQUENCE SIZE (1..ub-attribute-values) OF SEQUENCE {
      -- at least one must be specified --
      value                 [0] ATTRIBUTE.&Type ({AttributeTable} {@attribute-type}) OPTIONAL,
      position              [1] INTEGER (1..ub-attribute-values) OPTIONAL } }
```

The components of **entry-modification** have the following meaning:

1) **Strict** (O): This specifies whether a **strict** interpretation of the modification requests shall be followed. For each type of modification defined below, one or more minor error conditions are defined. If **strict** is *true*, the occurrence of any of these minor errors shall cause the termination of the abstract-operation with a modify-error. If **strict** is *false*, then the performance of the abstract-operation shall continue regardless of the occurrence of a minor error.

2) **Modification** (M): This indicates the modification requested.

   If **add-attribute** is specified, this identifies a new attribute to be added to the selected entries, which is fully specified by the argument. If strict interpretation is requested an attempt to add an already existing attribute to an entry causes the termination of the abstract-operation with a modify-error. If strict interpretation is not requested then the new attribute replaces any existing attribute.

   If **remove-attribute** is specified, this identifies (by its type) an attribute to be removed from the selected entries. If strict interpretation is requested an attempt to remove an attribute which is not present in an entry causes the generation of a modify-error; otherwise the abstract-operation continues.

   If **add-values** is specified, this identifies an attribute by the attribute-type in the argument and one or more values to be added to it for each selected entry. If strict interpretation is requested and an attribute of the specified type is not already present in the entry, a modify-error is generated; otherwise the attribute is created. If strict interpretation is requested, an attempt to add a value which is already present in the attribute causes the generation of a modify-error; otherwise the value is added. If strict interpretation is requested, an attempt to add a second instance of a value to an attribute which does not permit multiple instances of the same value causes the generation of a modify-error; otherwise the request is ignored. If **position** is specified, it identifies the position within the sequence of attribute-values that the supplied value shall occupy. If a **position** is specified which exceeds the number of attribute-values present by more than one, a modify-error is generated, regardless of the value of the strict parameter. If an attribute-value already occupies the specified **position**, then that value, and all values which follow it, are displaced. Where **position** is omitted, the new values are added to the end of the sequence of attribute-values. An attribute's first value occupies position 1.

   If **remove-values** is specified, this identifies an attribute by the attribute type in the argument and one or more values to be removed from it for each selected entry. If strict interpretation is requested and the attribute is absent from the entry (or lacks any of the specified values), then a modify-error is generated; otherwise the abstract-operation continues. If more than one attribute-value matches a specified value, and a **position** component is not present, then all such attribute-values are removed. If a **position** component is present, then only the attribute-value at that position is removed; the **value** component need not be supplied and is ignored if present. If a **position** is specified which exceeds the number of attribute-values present, then a modify-error is generated regardless of the value of the strict parameter. If all attribute-values are removed, then the attribute is removed from the entry.

   An attribute-error is generated if the add-values or remove-values modification is specified for a single-valued attribute-type. This static error is detected before any modifications are applied.

d) **Modify-extensions** (O): This parameter allows for future general and content-specific extensions to modify-argument. No extensions are defined in this Service Definition.

NOTE – To summarize, the following static errors are detected before any modifications are attempted:

a) an entry is specified which does not exist, or belongs to an entry-class not subject to modification;

b) modification is requested for an unavailable attribute-type (i.e. one not supported by the MS or not subscribed to by the MS-user), or for an attribute-type not subject to modification;

c) an add-values or remove-values modification is specified for a single-valued attribute-type;

d) a request is made to assign an unregistered message-group-name to an entry.

8.2.7.2 Modify-result

Should the request succeed, the **modify-result** shall be returned.

```
ModifyResult ::= SET {
    entries-modified            [0] SEQUENCE SIZE (1..ub-messages) OF SequenceNumber OPTIONAL,
    modify-result-extensions    [1] MSExtensions OPTIONAL }
```

The parameters of **modify-result** have the following meaning:

a) **Entries-modified** (C): The sequence-numbers of the entries selected for modification. Present if at least one entry was selected for modification.

NOTE – It is possible for an entry selected for modification to remain unchanged after the modifications have been applied.

b) **Modify-result-extensions** (C): This parameter allows for future general and content-specific extensions to modify-result. No extensions are defined in this Service Definition.

8.2.7.3 Modify Abstract-errors

Should the request fail, one of the listed abstract-errors shall be reported. Clause 9 defines the circumstances under which each abstract-error is reported.

8.3 MS-submission Port abstract-operations

The following abstract-operations are available at the MS-submission Port:

a) MS-message-submission;

b) MS-probe-submission;

c) MS-cancel-deferred-delivery;

d) MS-submission-control.

These abstract-operations correspond directly to those supplied at the Submission Port of the MTS Abstract Service, but offer additional functionality related to the storage capabilities of the MS. When an MS-message-submission abstract-operation or MS-probe-submission abstract-operation is invoked, the MS extracts those parameters of the abstract-operation argument which concern MS-specific activity and, where submission is requested, invokes the corresponding operation at the MTS Submission Port. Where submission to the MTS is not requested, the effect of the MS-message-submission abstract-operation is to create an entry in the Draft entry-class.

8.3.1 MS-message-submission abstract-operation

The **MS-message-submission** abstract-operation is used to submit a message to the MTS and optionally store a copy of it, either in the Submission-log entry-class, or in both the Submission and Submission-log entry-classes. Alternatively, the message may be stored (without submission) in the Draft entry-class. MS-message-submission makes use of the MTS Message-submission abstract-operation, which is defined in 8.2.1.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
ms-message-submission ABSTRACT-OPERATION ::= {
    ARGUMENT    MSMessageSubmissionArgument
    RESULT      MSMessageSubmissionResult
    ERRORS      {submission-control-violated | element-of-service-not-subscribed |
                originator-invalid | recipient-improperly-specified | inconsistent-request |
                security-error | unsupported-critical-function | remote-bind-error,
                ... -- 1994 extension additions --,
                ms-extension-error | message-group-error | entry-class-error | service-error}
    CODE        op-ms-message-submission }
```

8.3.1.1 MS-message-submission-argument

```
MSMessageSubmissionArgument ::= SEQUENCE {
    COMPONENTS OF   MessageSubmissionArgument -- This imported type has IMPLICIT tags --,
                    -- 1994 extension --
    submission-options      [4] MSSubmissionOptions OPTIONAL }
```

The parameters of **MS-message-submission-argument** have the following meaning:

a) **Message-submission-argument** (M): This contains the argument of the Message-submission abstract-operation as defined in 8.2.1.1.1 of ITU-T Rec. X.411 | ISO/IEC 10021–4.

This Service Definition defines an extension (see 9.1 of ITU-T Rec. X.411 | ISO/IEC 10021–4) to the message-submission-argument of the MTS abstract-service. This extension, **forwarding-request**, may be present only if a 1988 Application Context is in use.

```
forwarding-request EXTENSION ::= {
    SequenceNumber,
    IDENTIFIED BY standard-extension:36 }
```

This extension is present if the MS-user requests that a delivered-message stored in the MS is to be forwarded to other users. It indicates the sequence-number of the entry to be forwarded. The request is processed as defined in the Specification for the content-type of the forwarding message. An error in processing the **forwarding-request** is reported using the inconsistent-request abstract-error of 8.2.2.7 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

NOTE 1 – When a 1988 Application Context is in use forwarding-request provides a mechanism for forwarding delivered-message entries. In the 1994 Application Contexts a content-specific mechanism may be available to enable submitted messages to incorporate the content (in whole or part) of entries of the Stored-message entry-class. See 19.5.1 of ITU-T Rec. X.420 | ISO/IEC 10021-7 for an instance of use of each of these mechanisms.

b) **Submission-options** (O): This contains submission requests which are specific to MS operation (see 8.1.6). If these indicate that submission to the MTS is requested, the MS removes this parameter from the argument before invoking Message-submission. If the parameter is absent and the present abstract-association identified a UA-registration in its MS-bind-argument, the submission-options are drawn from the UA-submission-defaults of that UA-registration (see 8.2.5.1, item g). If the parameter is absent and the present abstract-association did not identify a UA-registration, or the UA-registration did not contain UA-submission-defaults, the submission-options are drawn from the general (non UA-specific) submission-defaults registered by Register-MS.

NOTE 2 – Where submission-options request the creation of a draft-message entry, the message-submission-argument must be syntactically complete, in order to conform to the abstract-syntax of the abstract-operation.

8.3.1.2 MS-message-submission-result

Should the request succeed, the **MS-message-submission-result** shall be returned.

```
MSMessageSubmissionResult ::= CHOICE {
    mts-result SET {
        COMPONENTS OF    MessageSubmissionResult -- This imported type has IMPLICIT tags --,
                         -- 1994 extension --
        ms-message-result [4] CommonSubmissionResults OPTIONAL },
                         -- 1994 extension --
    store-draft-result   [4] CommonSubmissionResults }
```

The parameters of **MS-message-submission-result** have the following meaning:

a) **MTS-result** (C): This is returned only if the MS-user requested submission of the message to the MTS and that submission was successful. It contains the following components:

1) **Message-submission-result** (M): This contains the result of the Message-submission abstract-operation defined in 8.2.1.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

2) **MS-message-result** (C): This contains results specific to MS operation. The components of common-submission-results shall be present in the cases defined in 8.1.7.

b) **Store-draft-result** (C): This is returned if the MS-user requested creation of a draft-message entry in the Draft entry-class rather than submission to the MTS. The created-entry component indicates the sequence-number of the newly created draft-message entry. The other components of store-draft-result are defined in 8.1.7.

8.3.1.3 MS-message-submission Abstract-errors

Should the request fail, one of the abstract-errors defined in clause 9 of this Service Definition, or in 8.2.1.1.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4 shall be reported.

8.3.2 MS-probe-submission abstract-operation

The **MS-probe-submission** abstract-operation is used to submit a probe to the MTS and optionally store a copy of it, either in the Submission-log entry-class, or in both the Submission and Submission-log entry-classes. MS-probe-submission makes use of the MTS Probe-submission abstract-operation, which is defined in 8.2.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
ms-probe-submission ABSTRACT-OPERATION ::= {
    ARGUMENT    MSProbeSubmissionArgument
    RESULT      MSProbeSubmissionResult
    ERRORS      {submission-control-violated | element-of-service-not-subscribed |
                originator-invalid | recipient-improperly-specified | inconsistent-request |
                security-error | unsupported-critical-function | remote-bind-error,
                ... -- 1994 extension additions --,
                ms-extension-error | message-group-error | entry-class-error | service-error}
    CODE        op-ms-probe-submission }
```

8.3.2.1 MS-probe-submission-argument

```
MSProbeSubmissionArgument ::= SET {
    COMPONENTS OF  ProbeSubmissionArgument -- This imported type has IMPLICIT tags --,
                   -- 1994 extension --
    submission-options   [4] MSSubmissionOptions OPTIONAL }
```

The parameters of **MS-probe-submission-argument** have the following meaning:

a) **Probe-submission-argument** (M): This specifies the argument of the Probe-submission abstract-operation as defined in 8.2.1.2.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

b) **Submission-options** (O): This contains submission requests which are specific to MS operation (see 8.1.6). The MS removes this parameter from the argument before invoking Probe-submission. The object-entry-class component of this parameter shall not contain the value *draft*; the relevant Specification may indicate what action (if any) is taken concerning the MS-submission-extensions component, if one is present. If submission-options is absent and the present abstract-association identified a UA-registration in its MS-bind-argument, the submission-options are drawn from the UA-submission-defaults of that UA-registration (see 8.2.5.1, item g). If the parameter is absent and the present abstract-association did not identify a UA-registration, or the UA-registration did not contain UA-submission-defaults, the submission-options are drawn from the general (non UA-specific) submission-defaults registered by Register-MS.

8.3.2.2 MS-probe-submission-result

Should the request succeed, the **MS-probe-submission-result** is returned.

```
MSProbeSubmissionResult ::= SET {
    COMPONENTS OF    ProbeSubmissionResult -- This imported type has IMPLICIT tags --,
                     -- 1994 extension --
    ms-probe-result  [4] CommonSubmissionResults OPTIONAL }
```

The parameters of **MS-probe-submission-result** have the following meaning:

a) **Probe-submission-result** (M): This is the result of the Probe-submission abstract-operation as defined in 8.2.1.2.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

b) **MS-probe-result** (C): This contains results specific to MS operation. The components of common-submission-results shall be present in the cases defined in 8.1.7.

8.3.2.3 MS-probe-submission Abstract-errors

Should the request fail, one of the abstract-errors defined in clause 9 of this Service Definition, or 8.2.1.2.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4 shall be reported.

8.3.3 MS-cancel-deferred-delivery abstract-operation

The argument, result, and errors of the **MS-cancel-deferred-delivery** abstract-operation are identical with those of the Cancel-deferred-delivery abstract-operation defined in 8.2.1.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
ms-cancel-deferred-delivery ABSTRACT-OPERATION ::= cancel-deferred-delivery
```

If the request for cancellation of deferred delivery is successful, the MS searches for an entry in the Submission and Submission-log entry-classes corresponding to the submitted-message for which deferred-delivery has been cancelled. If this entry is present, the MS shall attach a deferred-delivery-cancellation-time attribute to it to record the date and time at which delivery cancellation occurred, and shall update the AC-report-summary attribute to record the cancellation.

8.3.4 MS-submission-control abstract-operation

The argument, result, and errors of the **MS-submission-control** abstract-operation are identical with those of the Submission-control abstract-operation defined in 8.2.1.4 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
ms-submission-control ABSTRACT-OPERATION ::= submission-control
```

9 Abstract-errors

This clause defines the following **abstract-errors** associated with using the abstract-operations at the Retrieval Port and MS-submission Port:

a) Attribute error;

b) Auto-action request error;

c) Delete error;

d) Fetch restriction error;

e) Invalid parameters error;

f) Range error;

g) Security error;

h) Sequence-number error;

i) Service error;

j) Message-group error;

k) MS-extension error;

l) Old credentials incorrectly specified;

m) New credentials unacceptable;

n) Register-MS error;

o) Modify error;

p) Entry-class error.

9.1 Error precedence

When an error is detected during the performance of an abstract-operation, the MS shall discontinue processing the abstract-operation and return the error to the MS-user. Where more than one error is manifest, the actual errors returned are at the discretion of the MS.

NOTE – When using a 1988 Application Context, the MS is not required to discontinue processing the abstract-operation when an error is encountered. Consequently, several abstract-errors allow more than one error to be reported.

9.2 Attribute-error

An **Attribute-error** reports an attribute related problem.

```
attribute-error ABSTRACT-ERROR ::= {
    PARAMETER SET     {
        problems      [0]   SET SIZE (1..ub-per-entry) OF SET {
            problem           [0] AttributeProblem,
            type              [1] ATTRIBUTE.&id ({AttributeTable}),
            value             [2] ATTRIBUTE.&Type ({AttributeTable} {@.type}) OPTIONAL} }
    CODE              err-attribute-error }
```

```
AttributeProblem ::= INTEGER {
        invalid-attribute-value             (0),
        unavailable-attribute-type          (1),
        inappropriate-matching              (2),
        attribute-type-not-subscribed       (3),
        inappropriate-for-operation         (4),
        -- 1994 extensions --
        inappropriate-modification          (5),
        single-valued-attribute             (6) } (0..ub-error-reasons)
```

The parameter has the following meaning:

Problems (M): The particular problems encountered. Each problem is accompanied by an indication of the attribute-type, and, if necessary to avoid ambiguity, the value which caused the problem:

a) *invalid-attribute-value:* A purported attribute-value specified as an argument of the abstract-operation does not conform to the data-type defined for the attribute-type concerned.

b) *unavailable-attribute-type:* A purported attribute-type used as an argument of the abstract-operation is not one of those which is supported by the MS.

c) *inappropriate-matching:* The filter contains a filter-item in which an attribute is matched using a matching rule (equality, ordering, substrings, or other-match) that is not defined for that attribute-type or is not supported by the MS.

d) *attribute-type-not-subscribed:* An attribute-type used as an argument of the abstract-operation is not one of those to which the MS-user has subscribed.

NOTE – A change in subscription is not necessarily reflected in the attributes present in an entry created before the change.

e) *inappropriate-for-operation:* An attribute-type used as an argument of the abstract-operation is unsuitable for its required use.

f) *inappropriate-modification:* An attribute-type which is not subject to modification has been used as an argument of the Modify abstract-operation.

g) *single-valued-attribute:* An attempt has been made to use the Modify abstract-operation to apply an add-values or remove-values modification to a single-valued attribute-type.

9.3 Auto-action-request-error

An **Auto-action-request-error** reports a problem related to the registration of an auto-action.

```
auto-action-request-error ABSTRACT-ERROR ::= {
    PARAMETER  SET {
        problems        [0]  SET SIZE (1..ub-auto-registrations) OF SET {
            problem             [0]  AutoActionRequestProblem,
            type                [1]  AUTO-ACTION.&id ({AutoActionTable}) } }
    CODE            err-auto-action-request-error }

AutoActionRequestProblem ::= INTEGER {
        unavailable-auto-action-type        (0),
        auto-action-type-not-subscribed     (1),
        -- 1994 extension --
        not-willing-to-perform              (2) } (0..ub-error-reasons)
```

The parameter has the following meaning:

Problems (M): The particular problems encountered. Each problem is accompanied by an indication of the auto-action-type which caused the problem:

a) *unavailable-auto-action-type:* An auto-action-type used as an argument of the abstract-operation is not one of those which is supported by the MS abstract-service-provider.

b) *auto-action-type-not-subscribed:* An auto-action-type used as an argument of the abstract-operation is not one of those to which the MS-user has subscribed.

c) *not-willing-to-perform:* The auto-action registration is refused because, when executed, it would lead to an excessive consumption of resources.

9.4 Delete-error

A **Delete-error** reports a problem in an attempt to delete one or more entries of an entry-class.

```
delete-error ABSTRACT-ERROR ::= {
    PARAMETER SET {
        problems          [0]  SET SIZE (1..ub-messages) OF SET {
            problem               [0]  DeleteProblem,
            sequence-number       [1]  SequenceNumber},
                                  -- 1994 extension --
        entries-deleted   [1]  SET SIZE (1..ub-messages) OF SequenceNumber OPTIONAL }
    CODE      err-delete-error }

DeleteProblem ::= INTEGER {
    child-entry-specified          (0),
    delete-restriction-problem     (1),
    -- 1994 extensions --
    new-entry-specified            (2),
    entry-class-restriction        (3),
    stored-message-exists          (4) } (0..ub-error-reasons)
```

The parameter has the following meaning:

a) **Problems** (M): The particular problems encountered. Each problem is accompanied by an indication of the sequence-number of the entry which caused the problem:

   1) *child-entry-specified:* An attempt has been made to delete a child-entry.

   2) *delete-restriction-problem:* An attempt has been made to violate a restriction specified for the Delete abstract-operation (see 8.2.4).

   3) *new-entry-specified:* An attempt has been made to delete an entry of the Delivery entry-class which has a retrieval-status of *new*.

   4) *entry-class-restriction:* An attempt has been made to delete an entry of an entry-class for which deletion is restricted or prohibited.

   5) *stored-message-exists:* An attempt has been made to delete an entry of the Message-log entry-class for which a corresponding entry exists in the Stored-message entry-class.

b) **Entries-deleted** (C): This parameter identifies the entries deleted before the abstract-operation terminated. The parameter is absent if no entries were deleted.

9.5 Fetch-restriction-error

A **Fetch-restriction-error** reports an attempt to violate a restriction associated with the Fetch abstract-operation.

```
fetch-restriction-error ABSTRACT-ERROR ::= {
    PARAMETER  SET {
        problems [0]  SET SIZE (1..ub-default-registrations) OF SET {
            problem     [3] FetchRestrictionProblem,
            restriction CHOICE {
                content-type          [0]  OBJECT IDENTIFIER,
                eit                   [1]  MS-EITs,
                attribute-length      [2]  INTEGER} } }
    CODE       err-fetch-restriction-error }

FetchRestrictionProblem ::= INTEGER {
    content-type-problem           (1),
    eit-problem                    (2),
    maximum-length-problem         (3) } (0..ub-error-reasons)
```

The parameter has the following meaning:

Problems (M): The particular problems encountered. Each problem is accompanied by an indication of the offending content-type, encoded-information-type or attribute-length which caused the problem:

a) *content-type-problem:* The content-type of the message being fetched is disallowed by the fetch-restrictions currently in force.

b) *EIT-problem:* The encoded-information-types requested in the Fetch abstract-operation are disallowed by the fetch-restrictions currently in force.

c) *maximum-length-problem:* The length of the encoding of an attribute-value being fetched exceeds that allowed by the fetch-restrictions currently in force.

9.6 Invalid-parameters-error

An **Invalid-parameters-error** reports an undefined problem in processing the argument of an abstract-operation. This error may be used, for example, to report that an optional parameter was present in the wrong context, or to report that a value for one of the parameters is inappropriate.

```
invalid-parameters-error ABSTRACT-ERROR ::= {
    PARAMETER    NULL
    CODE         err-invalid-parameters-error }
```

This error has a Null parameter.

9.7 Range-error

A **Range-error** reports a problem related to the range specified in a selector as an argument of an abstract-operation.

```
range-error ABSTRACT-ERROR ::= {
    PARAMETER  SET {
        problem      [0]  RangeProblem}
    CODE         err-range-error }

RangeProblem ::= INTEGER {
        reversed     (0) } (0..ub-error-reasons)
```

The parameter has the following meaning:

Problem (M): The particular problem encountered. The value *reversed* is returned if the upper bound indicated a sequence-number or creation-time before that indicated by the lower bound.

This abstract-error shall not be generated when a 1994 Application Context is in use.

9.8 Security-error

A **Security-error** reports that the requested abstract-operation cannot be provided because it would violate the security-policy in force. This error is defined in 8.2.2.8 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

9.9 Sequence-number-error

A **Sequence-number-error** reports a problem related to the sequence-number specified in the argument of an abstract-operation.

```
sequence-number-error ABSTRACT-ERROR ::= {
    PARAMETER  SET {
        problems       [1]  SET SIZE (1..ub-messages) OF SET {
            problem            [0]  SequenceNumberProblem,
            sequence-number    [1]  SequenceNumber} }
    CODE       err-sequence-number-error }

SequenceNumberProblem ::= INTEGER {
    no-such-entry              (0) } (0..ub-error-reasons)
```

The parameter has the following meaning:

Problems (M): The particular problems encountered. Each problem is accompanied by an indication of the sequence-number which caused the problem. If *no-such-entry* is returned the sequence-number supplied does not match that of any entry of the specified entry-class.

9.10 Service-error

A **Service-error** reports an error related to the provision of the service.

```
service-error ABSTRACT-ERROR ::= {
    PARAMETER     ServiceErrorParameter
    CODE          err-service-error }

ServiceErrorParameter ::= SET {
    problem                      [0]  ServiceProblem,
                                 -- 1994 extension --
    supplementary-information    [1]  GeneralString (SIZE (1..ub-supplementary-info-length)) OPTIONAL }
```

```
ServiceProblem ::= INTEGER {
    busy                    (0),
    unavailable             (1),
    unwilling-to-perform    (2) } (0..ub-error-reasons)
```

The parameter has the following meaning:

a) **Problem** (M): The particular problem encountered:

1) *busy:* The MS, or some part of it, is too busy at present to perform the requested abstract-operation but may be able to do so after a short while.

2) *unavailable:* The MS, or some part of it, is unavailable at present.

3) *unwilling-to-perform:* The MS is not prepared to execute this request because it would lead to excessive consumption of resources.

b) **Supplementary-information** (C): This provides further details of the reported problem. It is present at the discretion of the MS service provider.

9.11 Message-group-error

A **Message-group-error** reports a problem in the use of a message-group-name.

```
message-group-error ABSTRACT-ERROR ::= {
    PARAMETER  MessageGroupErrorParameter
    CODE       err-message-group-error }

MessageGroupErrorParameter ::= SET {
    problem    [0] MessageGroupProblem,
    name       [1] MessageGroupName }

MessageGroupProblem ::= INTEGER {
    name-not-registered         (0),
    name-already-registered     (1),
    parent-not-registered       (2),
    group-not-empty             (3),
    name-in-use                 (4),
    child-group-registered      (5),
    group-depth-exceeded        (6) } (0..ub-error-reasons)
```

The parameter has the following meaning:

a) **Problem** (M): This identifies the problem encountered. The following problems may be indicated:

1) *name-not-registered:* An attempt has been made to assign an unregistered message-group-name to an entry's message-group-name attribute using the Modify abstract-operation, or to deregister an unregistered message-group-name using the Register-MS abstract-operation.

2) *name-already-registered:* An attempt has been made to register a message-group name which is already registered.

3) *parent-not-registered:* An attempt has been made to register a message-group-name containing more than one group-name-part, whose parent is not registered (where the parent is defined as the name formed by omitting the message-group-name's last group-name-part).

4) *group-not-empty:* An attempt has been made to deregister a message-group-name which is present in at least one entry's message-group-name attribute.

5) *name-in-use:* An attempt has been made to deregister a message-group-name which is referenced in a registered Auto-modify auto-action.

6) *child-group-registered:* An attempt has been made to deregister a message-group-name for which at least one child message-group-name is currently registered (where child is defined as a name formed by adding one group-name-part to the message-group-name).

7) *group-depth-exceeded:* An attempt has been made to register a message-group-name containing more than the maximum number of group-name-parts supported by the MS.

b) **Name** (M): This indicates the message-group-name which is the subject of the abstract-error.

9.12 MS-extension-error

An **MS-extension-error** reports a problem concerning an extension parameter presented in the argument of an abstract-operation.

```
ms-extension-error ABSTRACT-ERROR ::= {
    PARAMETER  MSExtensionErrorParameter
    CODE       err-ms-extension-error }

MSExtensionErrorParameter ::= CHOICE {
        ms-extension-problem      [0]  MSExtensionItem,
        unknown-ms-extension      [1]  OBJECT IDENTIFIER }
```

The parameter has the following meaning:

a) **MS-extension-problem** (C): This identifies the problem encountered. The definition of an instance of an **MS-extension-problem** may appear in the Specification for a given content-type.

NOTE – The MS-extension-problems specified for Interpersonal Messaging are enumerated in the IP-submission-errors information object set (see 19.5.2.3 of ITU-T Rec. X.420 | ISO/IEC 10021-7).

b) **Unknown-MS-extension** (C): An extension presented in the argument of an abstract-operation is not known to the MS. The parameter contains the Object Identifier value of the unknown extension drawn from its identifier field.

9.13 Register-MS-error

A **Register-MS-error** reports a problem in an attempt to register information with the MS.

```
register-ms-error ABSTRACT-ERROR ::= {
    PARAMETER SET {
        problem             [0]  RegistrationProblem,
        registration-type   [1]  RegistrationTypes }
    CODE     err-register-ms-error }

RegistrationProblem ::= ENUMERATED {
    registration-not-supported          (0),
    registration-improperly-specified   (1),
    registration-limit-exceeded         (2),
    ... -- For future extension additions -- }
```

The parameter has the following meaning:

a) **Problem** (M): The particular problem encountered. Either the requested registration is not one of those supported by the MS, or the request was improperly specified, or a pragmatic limit has been reached in the number of registrations that the MS is able to record.

b) **Registration-type** (M): This indicates the type of registration requested by the MS-user with which the problem is associated. See 8.2.5.1, item j, for the definition of registration-types.

NOTE – Where a different abstract-error provides a more precise description of a registration problem, Register-MS-error is not reported.

9.14 Old-credentials-incorrectly-specified

An **Old-credentials-incorrectly-specified** abstract-error reports that the credentials cannot be changed because the current (old-) credentials were incorrectly specified. This error is defined in 8.4.2.3 of ITU-T Rec. X.411 | ISO/IEC-10021-4.

9.15 New-credentials-unacceptable

A **New-credentials-unacceptable** abstract-error reports that the credentials cannot be changed because the new-credentials are unacceptable. This error is defined in 8.4.2.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

9.16 Modify-error

A **Modify-error** reports a problem in an attempt to modify the attributes of an entry.

```
modify-error ABSTRACT-ERROR ::= {
    PARAMETER  ModifyErrorParameter
    CODE       err-modify-error }

ModifyErrorParameter ::= SET {
    entries-modified      [0] SEQUENCE SIZE (1..ub-messages) OF SequenceNumber OPTIONAL,
    failing-entry         [1] SequenceNumber,
    modification-number   [2] INTEGER,
    problem               [3] ModifyProblem }
```

The parameter has the following meaning:

a) **Entries-modified** (C): The sequence-numbers of the entries to which modifications were applied successfully. Present if at least one entry was modified successfully.

b) **Failing-entry** (M): The sequence-number of the entry for which a modification failed.

c) **Modification-number** (M): This identifies which modification, in the sequence presented in the modifications parameter of modify-argument, caused the generation of the error. The first modification is numbered '1'.

d) **Problem** (M): This identifies the problem encountered.

```
ModifyProblem ::= INTEGER {
    attribute-not-present              (0),
    value-not-present                  (1),
    attribute-or-value-already-exists  (2),
    invalid-position                   (3),
    modify-restriction-problem         (4) } (0..ub-error-reasons)
```

1) *attribute-not-present:* The entry lacks one of the attribute-types specified as an argument of the operation.

2) *value-not-present:* The entry lacks one of the attribute values specified as an argument of the operation.

3) *attribute-or-value-already-exists:* An attempt has been made to add an attribute which already existed in the entry, or a value which already existed in the attribute.

4) *invalid-position:* The attribute contains too few values for the modification to be valid.

5) *modify-restriction-problem:* An attempt has been made to violate a restriction specified for the Modify abstract-operation.

If no entries have been modified successfully and an error other than those defined in **problem** occurs, the appropriate abstract-error shall be returned. However, if some entries have been modified successfully, the MS shall return a modify-error which specifies the entries-modified and reports the error.

9.17 Entry-class-error

An **Entry-class-error** reports a problem in an attempt to address an entry-class in an abstract-operation.

```
entry-class-error ABSTRACT-ERROR ::= {
    PARAMETER  EntryClassErrorParameter
    CODE       err-entry-class-error }

EntryClassErrorParameter ::= SET {
    entry-class  [0] EntryClass,
    problem      [1] BIT STRING {
        unsupported-entry-class      (0),
        entry-class-not-subscribed   (1),
        inappropriate-entry-class    (2) } }
```

The parameter has the following meaning:

a) **Entry-class** (M): This indicates the entry-class specified in the abstract-operation.

b) **Problem** (M): Either the entry-class specified is not one of those supported by the MS, or is not one of those to which the MS-user has subscribed, or is inappropriate for the requested abstract-operation.

SECTION 3 – GENERAL-ATTRIBUTE-TYPES, MATCHING-RULES AND AUTO-ACTION-TYPES

10 Overview

The MS information model and the attribute, matching-rule, and auto-action concepts are introduced in 6.3 and 6.5. Clause 11 defines the **general-attribute-types** which are specified for the MS. Clause 12 defines the **general-matching-rules** which are specified for the MS. Clause 13 defines the **general-auto-action-types** which are specified for the MS.

11 General-attribute-types

The general-attribute-types are valid for all message content-types. Other attribute-types, which are content-specific, are defined in their respective Specification, e.g. the IPMS-specific attribute-types for MS are defined in clause 19 of ITU-T Rec. X.420 | ISO/IEC 10021-7.

11.1 General-attribute-types overview

The general-attributes defined for the entries of each of the entry-classes are listed in Tables 2 and 3. They are constructed mainly from the parameters of the Message-submission, Probe-submission, Message-delivery and Report-delivery abstract-operations of the MTS abstract-service as defined in clause 8 of ITU-T Rec. X.411 | ISO/IEC 10021-4, and such attributes are correspondingly named. Some general-attributes are generated, and some of these also maintained by the MS.

Table 2 defines the following for each of the general-attribute-types:

- for entries of the Delivery and Delivery-log entry-classes, whether the attribute-type is always present, conditionally present, or absent in a delivered-message entry, a delivered-report entry, or a returned-content entry respectively;
- for entries of the Submission and Submission-log entry-classes, whether the attribute-type is always present, conditionally present, or absent in a submitted-message entry and submitted-probe entry;
- for entries of the Draft entry-class, whether the attribute-type is always present, conditionally present, or absent in a draft-message entry.
- whether support by the MS is mandatory or optional for entries of the Stored-message and Message-log entry classes;
- whether the attribute-type is single-valued or multi-valued;
- whether values of the attribute may be returned in a List abstract-operation (and, for entries of the Delivery entry-class only, in an Alert abstract-operation);
- whether the attribute-type may be the subject of summary-requests in a Summarize abstract-operation.

Table 3 defines equivalent information for entries of the Auto-action-log entry-class.

For a more detailed description of the classifications in Tables 2 and 3 refer to the conventions in 5.2.

An optional attribute-type which is supported by an MS may be available to the MS-user subject to subscription. Each optional attribute-type may be the subject of individual subscription.

Attribute-types defined as present in a delivered-message entry may not always be present in a child-entry. The rules governing the presence of attribute-types in child-entries may be supplemented in the Specification which defines the content-type of the child-entry.

11.1.1 MS support requirements for general-attribute-types

The general-attribute-types for which support by the MS is mandatory are shown in Tables 2 and 3. Where an attribute is supported, it shall be supported in all entry-classes for which it is defined (with the possible exception of the Message-log entry-class). All supported attributes shall be available for retrieval by the Fetch abstract-operation, and, where indicated in Tables 2 and 3, by the List abstract-operation. Attribute-types that may be present in entries of the Delivery entry-class and are shown in Table 2 as available for List are also available for the Alert abstract-operation.

Selection of an entry (by means of Filter) for a supported attribute-type using one of its matching rules shall be available if support is also claimed for the matching-rule concerned.

NOTES

1 Support for an attribute-type does not in itself require support for all matching-rules defined for that attribute-type (see 12.5).

2 For some attribute-types originally defined in versions of this Service Definition published prior to 1994, additional generic matching-rules are defined in this version. In 1988 Application Contexts, support for these additional generic matches is not prohibited. However, some MS implementations will generate an attribute-error if requested to perform a match of this type.

Table 2 – Message Store common general-attribute-types

Attribute-type name	Presence in:						Support level by MS		Single/multi-valued	Available for List	Available for Summarize
	delivered-message entry	delivered-report entry	returned-content entry	submitted-message entry*	submitted-probe entry*	draft-message entry*	Stored-message entry-class	Message-log entry-class*			
AC-correlated-report-list*	–	–	–	C	C	–	O	O	M	Y	N
AC-report-summary*	–	–	–	C	C	–	O	O	M	Y	N
AC-uncorrelated-report-list*	–	–	–	C	C	–	O	O	M	Y	N
Child-sequence-numbers	C	C	C	C	–	C	M	M	M	Y	N
Content	P	–	P	P	–	P	M	–	S	N	N
Content-confidentiality-algorithm-identifier	C	–	–	C	–	C	O	O	S	Y	N
Content-correlator	–	C	–	C	C	C	O	O	S	Y	N
Content-identifier	C	C	–	C	C	C	O	O	S	Y	N
Content-integrity-check	C	–	–	–	–	–	O	O	S	Y	N
Content-length	P	–	P	P	C	P	O	O	S	Y	N
Content-returned	–	P	–	–	–	–	O	O	S	Y	Y
Content-type	P	C	C	P	P	P	M	M	S	Y	Y
Conversion-with-loss-prohibited	C	–	–	C	C	C	O	O	S	Y	N
Converted-EITs	C	–	–	–	–	–	O	O	M	Y	N
Creation-time	P	P	P	P	P	P	M	M	S	Y	N
Deferred-delivery-cancellation-time*	–	–	–	C	–	–	O	O	S	Y	N
Deferred-delivery-time*	–		–	C	–	C	O	O	S	Y	N
Deletion-time*	C	C	–	C	C	–	–	M	S	Y	N
Delivered-EITs	P	–	–	–	–	–	O	O	M	Y	N
Delivery-flags	P	–	–	–	–	–	O	O	S	Y	N
DL-expansion-history	C	C	–	–	–	–	O	O	M	Y	N
DL-expansion-prohibited*	–	–		C	C	C	O	O	S	Y	N
Entry-type	P	P	P	P	P	P	M	M	S	Y	Y
Internal-trace-information*	C	C	–	–	–	–	O	O	M	N	N
Latest-delivery-time*	–	–	–	C	–	C	O	O	S	Y	N
Marked-for-deletion*	C	C	–	C	C	C	O	O	S	Y	Y
Message-delivery-envelope	P	–	–	–	–	–	M	–	S	N	N
Message-delivery-time	P	–	–	–	–	–	O	O	S	Y	N
Message-group-name*	C	C	–	C	C	C	O	O	M	Y	Y
Message-identifier	P	–	–	P	P	–	O	O	S	Y	N
Message-notes*	C	C	C	C	C	C	O	O	M	Y	N
Message-origin-authentication-check	C	–	–	C	–	C	O	O	S	Y	N
Message-security-label	C	C	–	C	C	C	O	O	S	Y	N
Message-submission-envelope*	–	–	–	P	–	P	M	–	S	N	N
Message-submission-time	P	–	–	P	P	–	O	O	S	Y	N
Message-token	C	–	–		–	–	O	O	S	Y	N

Table 2 *(concluded)* – Message Store common general-attribute-types

Attribute-type name	Presence in:						Support level by MS		Single/multi-valued	Available for List	Available for Summarize
	delivered-message entry	delivered-report entry	returned-content entry	submitted-message entry*	submitted-probe entry*	draft-message entry*	Stored-message entry-class	Message-log entry-class*			
MS-originated*	–	–	–	C	–	–	O	O	S	Y	Y
MS-submission-error*	–	–	–	C	C	–	–	O	S	Y	N
Original-EITs	C	C	–	C	C	C	O	O	M	Y	N
Originally-intended-recipient-name	C	–	–	–	–	–	O	O	S	Y	N
Originating-MTA-certificate*	–	–	–	C	–	–	O	O	S	Y	N
Originator-certificate	C	–	–	C	C	–	O	O	S	Y	N
Originator-name	P	–	–	P	P	P	O	O	S	Y	N
Originator-report-request*	–	–	–	P	P	P	O	O	M	Y	N
Originator-return-address*	C	–	–	C	–	C	O	O	S	Y	N
Other-recipient-names	C	–	–	–	–	–	O	O	M	Y	N
Parent-sequence-number	C	–	P	C	–	C	M	M	S	Y	N
Per-message-indicators*	–	–	–	P	P	P	O	O	S	Y	N
Per-recipient-message-submission-fields*	–	–	–	P	–	P	O	O	M	N	N
Per-recipient-probe-submission-fields*	–	–	–	–	P	–	O	O	M	N	N
Per-recipient-report-delivery-fields	–	P	–	–	–	–	M	M	M	Y	N
Priority	P	–	–	P		P	O	O	S	Y	Y
Probe-origin-authentication-check*	–	–	–	–	C	–	O	O	S	Y	N
Probe-submission-envelope*	–	–	–	–	P	–	O	–	S	N	N
Proof-of-delivery-request	C	–	–	–	–	–	O	O	S	Y	N
Proof-of-submission*	–	–	–	C	–	–	O	O	S	Y	N
Recipient-names*	–	–	–	P	P	P	O	O	M	Y	N
Recipient-reassignment-prohibited*	–	–	–	C	C	C	O	O	S	Y	N
Redirection-history	C	C	–	–	–	–	O	O	M	Y	N
Report-delivery-envelope	–	P	–	–	–	–	M	–	S	N	N
Reporting-DL-name	–	C	–	–	–	–	O	O	S	Y	N
Reporting-MTA-certificate	–	C	–	–	–	–	O	O	S	Y	N
Report-origin-authentication-check	–	C	–	–	–	–	O	O	S	Y	N
Retrieval-status	P	P	P	P	P	P	M	M	S	Y	Y
Security-classification	C	C	–	C	C	C	O	O	S	Y	Y
Sequence-number	P	P	P	P	P	P	M	M	S	Y	N
Storage-period*	C	C	–	C	C	C	O	O	S	Y	N
Storage-time*	C	C	–	C	C	C	O	O	S	Y	N
Subject-submission-identifier	–	P	–	–	–	–	M	M	S	Y	N
This-recipient-name	P	–	–	–	–	–	O	O	S	Y	N
Trace-information*	C	C	–	–	–	–	O	O	M	N	N

NOTE – Attribute-types and table columns marked with an asterisk (*) are not defined for 1988 Application Contexts.

Table 3 – General-attribute-types for the Auto-action-log entry-class

Attribute-type name	Presence in Auto-action-log entry	Support level by MS	Single/ multi-valued	Available for List	Available for Summarize
Auto-action-error	C	M	S	Y	N
Auto-action-registration-identifier	P	M	S	Y	N
Auto-action-subject-entry	P	M	S	Y	N
Auto-action-type	P	M	S	Y	Y
Content-type	C	M	S	Y	Y
Creation-time	P	M	S	Y	N
Entry-type	P	M	S	Y	Y
Retrieval-status	P	M	S	Y	Y
Sequence-number	P	M	S	Y	N
NOTE – The auto-action-log entry-class is not defined for 1988 Application Contexts					

11.1.2 MS-user support requirements for general-attribute-types

No requirements are placed on the MS-user for the support of any of the general-attribute-types.

NOTE – The MS-user may employ a variety of strategies for retrieving information for display to the user. For example, the MS-user may choose to retrieve the complete message-delivery-envelope, or, alternatively, may selectively retrieve attributes which are derived from the message-delivery-envelope. Since both approaches are valid, no one method is prescribed.

11.2 Description of the general-attribute-types

The following subclauses contain a short description of each general-attribute-type together with its abstract-syntax using the defined syntax of the ATTRIBUTE information object class described in 6.3.3.3.

NOTE – Some general-attributes are used primarily for filtering and listing purposes while others may contain more complex (further structured ASN.1 data-types) and potentially voluminous information. Only a few general-attributes are suitable for use with the Summarize abstract-operation.

11.2.1 AC-correlated-report-list

This general-attribute, which is multi-valued, identifies the delivered-report entries which have been correlated with the originally specified recipients of a submitted message or probe. The attribute contains one value for each value of the *recipient-name* argument of the Message-submission or Probe-submission abstract-operation. Values at corresponding positions in this attribute and in the recipient-names attribute refer to the same recipient. Each value indicates the sequence-numbers of relevant delivered-report entries, and the positions within their per-recipient-report-delivery-fields attribute of the values which relate to that specified recipient. The attribute is generated by the MS. Subscription to this attribute requires subscription to the Auto-correlate-reports auto-action (see 13.3).

```
ms-ac-correlated-report-list ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX   ReportLocation,
    NUMERATION              multi-valued,
    ID                      id-att-ac-correlated-report-list }

ReportLocation ::= CHOICE {
    no-correlated-reports   [0] NULL,
    location                [1] SEQUENCE OF PerRecipientReport }

PerRecipientReport ::= SEQUENCE {
    report-entry    [0] SequenceNumber,
    position        [1] INTEGER (1..ub-recipients) DEFAULT 1 }
```

The components of **report-location** have the following meaning:

a) **No-correlated-reports** (C): No reports have been received which can be correlated with this originally specified recipient of the submitted message or probe. This is the initial value of the attribute.

b) **Location** (C): Identifies the sequence-numbers of the delivered-reports and the positions of the values within their per-recipient-report-delivery-fields attributes which relate to this originally specified recipient.

The attribute is created when the message or probe is submitted and is updated as reports are delivered. The content of this attribute is unaffected by any subsequent deletion of the delivered-report entries to which it refers.

11.2.2 AC-report-summary

This general-attribute, which is multi-valued, contains a summary of the reports requested and reports received from each originally specified recipient of a submitted message or probe. The attribute contains one value for each value of the *recipient-name* argument of the Message-submission or Probe-submission abstract-operation. Values at corresponding positions in this attribute and in the recipient-names attribute refer to the same recipient. The initial values of the attribute are set according to the values of the originator-report-request specified for each recipient, and are amended as reports are delivered. The attribute is generated by the MS. Subscription to this attribute requires subscription to the Auto-correlate-reports auto-action (see 13.3).

```
ms-ac-report-summary ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX          ReportSummary,
    EQUALITY MATCHING-RULE         integerMatch,
    ORDERING MATCHING-RULE         integerOrderingMatch,
    NUMERATION                     multi-valued,
    ID                             id-att-ac-report-summary }

ReportSummary ::= ENUMERATED {
    no-report-requested                                (0) -- non-delivery report suppressed --,
    no-report-received                                 (1) -- non-delivery report requested --,
    report-outstanding                                 (2) -- delivery report requested --,
    delivery-cancelled                                 (3),
    delivery-report-from-another-recipient             (4),
    non-delivery-report-from-another-recipient         (5),
    delivery-report-from-intended-recipient            (6),
    non-delivery-report-from-intended-recipient        (7) }
```

The values of **report-summary** have the following meaning:

a) *no-report-requested:* The suppression of non-delivery-reports was requested for this originally specified recipient and no reports have been received. This is a possible initial value of the attribute.

b) *no-report-received:* A request for the reporting of non-delivery was made of this originally specified recipient and no reports have been received. This is a possible initial value of the attribute.

c) *report-outstanding:* A request for the reporting of delivery and non-delivery was made of this originally specified recipient and no reports have been received. This is a possible initial value of the attribute.

d) *delivery-cancelled:* No reports shall be received because the delivery of the message was cancelled by means of the MS-cancel-deferred-delivery abstract-operation.

e) *delivery-report-from-another-recipient:* A delivery-report has been received concerning this originally specified recipient which indicates successful delivery of the message (or the potentially successful delivery of the probe's subject-message) to another recipient (possibly following redirection or DL-expansion).

f) *non-delivery-report-from-another-recipient:* A non-delivery-report has been received concerning this originally specified recipient which indicates a failure to deliver the message (or, potentially, the probe's subject message) to another recipient (possibly following redirection or DL-expansion).

g) *delivery-report-from-intended-recipient:* A report has been delivered confirming delivery to this originally specified recipient.

h) *non-delivery-report-from-intended-recipient:* A report has been delivered indicating non-delivery to this originally specified recipient.

The attribute is created when the message or probe is submitted. When reports are delivered to the MS, the value of AC-report-summary is updated if the resulting value is greater than that already recorded. This is in recognition that more than one report may be generated as result of PDS delivery or DL expansion. The method by which an MS determines the intended recipient to whom a report refers is not prescribed and is therefore implementation dependent. The content of this attribute is unaffected by any subsequent deletion of the delivered-report entries to which it refers.

NOTE – An MS which is colocated with an MTA may make use of the originally-specified-recipient-number component of per-recipient-report-transfer-fields to determine the intended recipient who is the subject of an incoming report.

11.2.3 AC-uncorrelated-report-list

This general-attribute, which is multi-valued, identifies each of the per-recipient-report-delivery-fields which have been correlated with a submitted message or probe but which have not been correlated with any of its originally specified recipients. It is generated by the MS. Subscription to this attribute requires subscription to the Auto-correlate-reports auto-action (see 13.3).

```
ms-ac-uncorrelated-report-list  ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       PerRecipientReport,
    NUMERATION                  multi-valued,
    ID                          id-att-ac-uncorrelated-report-list }
```

The attribute is created when the message or probe is submitted, and is updated as reports of the kind described are delivered. The content of this attribute is unaffected by any subsequent deletion of the delivered-report entries to which it refers.

11.2.4 Auto-action-error

This general-attribute identifies the auto-action-error which occurred when a registered auto-action was applied to an entry. The attribute may be present only in entries of the Auto-action-log entry-class (see 6.5.3). It is generated by the MS.

```
ms-auto-action-error ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       AutoActionError,
    NUMERATION                  single-valued,
    ID                          id-att-auto-action-error }
```

11.2.5 Auto-action-registration-identifier

This general-attribute indicates the registration-identifier of the auto-action whose execution is the subject of this entry in the Auto-action-log entry-class. The registered auto-action is identified by this attribute combined with the corresponding value of the auto-action-type general-attribute. The attribute may be present only in entries of the Auto-action-log entry-class (see 6.5.2). It is generated by the MS.

```
ms-auto-action-registration-identifier ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       INTEGER,
    EQUALITY MATCHING-RULE      integerMatch,
    ORDERING MATCHING-RULE      integerOrderingMatch,
    NUMERATION                  single-valued,
    ID                          id-att-auto-action-registration-identifier }
```

11.2.6 Auto-action-subject-entry

This general-attribute indicates the sequence-number of the Stored-message entry which was the subject of the auto-action processing recorded by the present entry. The attribute may be present only in entries of the Auto-action-log entry-class. It is generated by the MS.

```
ms-auto-action-subject-entry ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       SequenceNumber,
    EQUALITY MATCHING-RULE      integerMatch,
    ORDERING MATCHING-RULE      integerOrderingMatch,
    NUMERATION                  single-valued,
    ID                          id-att-auto-action-subject-entry }
```

11.2.7 Auto-action-type

This general-attribute identifies the type of the auto-action whose execution is the subject of this entry. The registered auto-action is identified by this attribute combined with the corresponding value of the auto-action-registration-identifier general-attribute. The attribute may be present only in entries of the Auto-action-log entry-class (see 6.5.1). It is generated by the MS

```
ms-auto-action-type ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       AutoActionType,
    EQUALITY MATCHING-RULE      objectIdentifierMatch,
    NUMERATION                  single-valued,
    ID                          id-att-auto-action-type }
```

11.2.8 Child-sequence-numbers

This general-attribute, which is multi-valued, identifies the child-entries of a parent-entry, if such exist. It is generated by the MS. It is present in a entry that has one or more child-entries associated with it. It is absent in an entry without child-entries (see 6.3.4).

```
ms-child-sequence-numbers ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       SequenceNumber,
    NUMERATION                  multi-valued,
    ID                          id-att-child-sequence-numbers }
```

11.2.9 Content

This general-attribute contains the complete *content* of a message as submitted by the MS-message-submission abstract-operation, or as delivered by the Message-delivery abstract-operation, or as presented in the *returned-content* of the Report-delivery abstract-operation. In this last case, the content general-attribute is created in the returned-content child-entry, and not in the delivered-report entry itself. See 8.2.1.1.1.37 and 8.3.1.2.1.14 of ITU-T Rec. X.411 | ISO/IEC 10021-4. The encoding of the *content* contained in the Octet String that constitutes this attribute shall be identical, when retrieved, to that delivered to the MS; there is no constraint on the encoding employed for the containing Octet String itself.

```
mt-content ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       Content,
    NUMERATION                  single-valued,
    ID                          id-att-content }
```

11.2.10 Content-confidentiality-algorithm-identifier

This general-attribute contains the *content-confidentiality-algorithm-identifier* argument of the Message-submission and Message-delivery abstract-operations. See 8.2.1.1.1.27 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-content-confidentiality-algorithm-identifier ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       AlgorithmIdentifier,
    NUMERATION                  single-valued,
    ID                          id-att-content-confidentiality-algorithm-identifier }
```

11.2.11 Content-correlator

This general-attribute contains the *content-correlator* argument of the Message-submission, Probe-submission, and Report-delivery abstract-operations. See 8.2.1.1.1.36 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-content-correlator ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ContentCorrelator,
    EQUALITY MATCHING-RULE      contentCorrelatorMatch,
    NUMERATION                  single-valued,
    ID                          id-att-content-correlator }
```

11.2.12 Content-identifier

This general-attribute contains the *content-identifier* argument of the Message-submission, Probe-submission, Message-delivery, and Report-delivery abstract-operations. It may be generated by the originator. See 8.2.1.1.1.35 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-content-identifier ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ContentIdentifier,
    EQUALITY MATCHING-RULE      contentIdentifierMatch,
    NUMERATION                  single-valued,
    ID                          id-att-content-identifier }
```

11.2.13 Content-integrity-check

This general-attribute contains the *content-integrity-check* argument of the Message-delivery abstract-operation; it provides the MS-user with a means of verifying that the content of a delivered-message has not been modified. It may be generated by the originator of the message. In submitted-message entries, this information is contained in the per-recipient-message-submission-fields attribute. See 8.2.1.1.1.28 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-content-integrity-check ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ContentIntegrityCheck,
    NUMERATION                  single-valued,
    ID                          id-att-content-integrity-check }
```

11.2.14 Content-length

This general-attribute indicates the length of the content, in octets, of a submitted message, a submitted probe, a draft-message, a delivered message, or the returned-content of a delivery-report. For a submitted probe, the attribute indicates the length in octets of the content of the subject-message. It is generated by the MS.

```
ms-content-length ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ContentLength,
    ORDERING MATCHING-RULE      integerOrderingMatch,
    NUMERATION                  single-valued,
    ID                          id-att-content-length }
```

11.2.15 Content-returned

This general-attribute indicates whether a content has been returned in the Report-delivery abstract-operation. It is generated by the MS (see 6.3.4).

```
ms-content-returned ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       BOOLEAN,
    EQUALITY MATCHING-RULE      booleanMatch,
    NUMERATION                  single-valued,
    ID                          id-att-content-returned }
```

11.2.16 Content-type

This general-attribute identifies the type of the content of a message, and is generated from the *content-type* argument of the Message-submission, Probe-submission, Message-delivery, and Report-delivery abstract-operations. See 8.2.1.1.1.34 of ITU-T Rec. X.411 | ISO/IEC 10021-4. When present in an entry of the Auto-action-log entry-class, it identifies the content-type of the entry which was the subject of auto-action processing; where that entry is a delivered-report, the presence of the attribute is conditional.

```
mt-content-type ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       OBJECT IDENTIFIER,
    EQUALITY MATCHING-RULE      objectIdentifierMatch,
    NUMERATION                  single-valued,
    ID                          id-att-content-type }
```

11.2.17 Conversion-with-loss-prohibited

This general-attribute contains the *conversion-with-loss-prohibited* argument of the Message-submission, Probe-submission, and Message-delivery abstract-operations. It indicates whether conversion with loss of information was allowed or prohibited by the originator. See 8.2.1.1.1.10 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-conversion-with-loss-prohibited ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ConversionWithLossProhibited,
    EQUALITY MATCHING-RULE      integerMatch,
    NUMERATION                  single-valued,
    ID                          id-att-conversion-with-loss-prohibited }
```

11.2.18 Converted-EITs

This general-attribute, which is multi-valued, identifies the encoded-information-types of the content after conversion, as indicated by the Message-delivery abstract-operation. It is generated from the *converted-encoded-information-types* argument of the Message-delivery abstract-operation. It is absent if no conversion took place. See 8.3.1.1.1.8 and 8.3.1.2.1.5 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-converted-EITs ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       MS-EIT,
    EQUALITY MATCHING-RULE      objectIdentifierMatch,
    NUMERATION                  multi-valued,
    ID                          id-att-converted-EITs }
```

11.2.19 Creation-time

This general-attribute indicates the time at which an entry was created in the MS. It is generated by the MS. See 6.3.2.

NOTE – Two or more consecutive entries may have the same creation-time.

```
ms-creation-time ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       CreationTime,
    EQUALITY MATCHING-RULE      uTCTimeMatch,
    ORDERING MATCHING-RULE      uTCTimeOrderingMatch,
    NUMERATION                  single-valued,
    ID                          id-att-creation-time }
```

11.2.20 Deferred-delivery-cancellation-time

This general-attribute indicates the time at which the MTS confirmed the successful cancellation of the deferred-delivery of a previously submitted message (see 8.3.3). It is generated by the MS.

```
ms-deferred-delivery-cancellation-time ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       DeferredDeliveryCancellationTime,
    EQUALITY MATCHING-RULE      uTCTimeMatch,
    ORDERING MATCHING-RULE      uTCTimeOrderingMatch,
    NUMERATION                  single-valued,
    ID                          id-att-deferred-delivery-cancellation-time }

DeferredDeliveryCancellationTime ::= UTCTime
```

11.2.21 Deferred-delivery-time

This general-attribute contains the *deferred-delivery-time* argument of the Message-submission abstract-operation. It indicates the time before which a submitted message shall not be delivered to its recipient(s). See 8.2.1.1.1.12 of ITU-T Rec. X.411 | ISO/IEC 10021-4. Subscription to this attribute requires subscription to the Auto-correlate-reports auto-action (see 13.3).

```
mt-deferred-delivery-time ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       DeferredDeliveryTime,
    EQUALITY MATCHING-RULE      uTCTimeMatch,
    ORDERING MATCHING-RULE      uTCTimeOrderingMatch,
    NUMERATION                  single-valued,
    ID                          id-att-deferred-delivery-time }
```

11.2.22 Deletion-time

This general-attribute may be present in entries of the Message-log entry-class, and indicates the time at which the corresponding entry in the Stored-message entry-class was deleted. In the case of a Submission-log entry for which no Submission entry is created, the deletion-time attribute is generated when the Submission-log entry is created and is assigned the same value as the entry's creation-time. It is generated by the MS. See 16.1.4.

```
ms-deletion-time ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       DeletionTime,
    EQUALITY MATCHING-RULE      uTCTimeMatch,
    ORDERING MATCHING-RULE      uTCTimeOrderingMatch,
    NUMERATION                  single-valued,
    ID                          id-att-deletion-time }

DeletionTime ::= UTCTime
```

11.2.23 Delivered-EITs

This general-attribute, which is multi-valued, identifies the encoded-information-types in the content of a message as delivered. It is generated by the MS, and is derived from the *original-encoded-information-types* and the *converted-encoded-information-types* arguments of the Message-delivery abstract-operation.

```
ms-delivered-EITs ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       MS-EIT,
    EQUALITY MATCHING-RULE      objectIdentifierMatch,
    NUMERATION                  multi-valued,
    ID                          id-att-delivered-EITs }
```

NOTE – This attribute has the same value as *converted-encoded-information-types* (if present); otherwise, the same value as *original-encoded-information-types*. It provides a convenient way of determining, in a single operation, the EITs present in the content.

11.2.24 Delivery-flags

This general-attribute contains information derived from the arguments of the Message-delivery abstract-operation. It indicates whether implicit-conversion of the content is prohibited. See 8.2.1.1.1.9 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-delivery-flags ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       DeliveryFlags,
    EQUALITY MATCHING-RULE      bitStringMatch,
    NUMERATION                  single-valued,
    ID                          id-att-delivery-flags }
```

11.2.25 DL-expansion-history

This general-attribute, which is multi-valued, contains a history of distribution-list expansion. If present in a delivered-message, it contains one or more distribution-list names used during the expansion process. It is absent if delivery to the MS-user did not involve any expansion of a distribution-list. If present in a delivered-report, it contains the originator name and one or more distribution-list names used during the expansion process. It is absent if the message, or the report's subject message, did not involve any expansion of a distribution-list. See 8.3.1.1.1.7 and 8.3.1.2.1.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-dl-expansion-history ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       DLExpansion,
    OTHER MATCHING-RULES        {redirectionOrDLExpansionMatch |
                                 redirectionOrDLExpansionElementsMatch |
                                 redirectionOrDLExpansionSubstringElementsMatch},
    NUMERATION                  multi-valued,
    ID                          id-att-dl-expansion-history }
```

11.2.26 DL-expansion-prohibited

This general-attribute contains the *DL-expansion-prohibited* argument of the Message-submission and Probe-submission abstract-operations; it indicates whether DL-expansion within the MTS is prohibited for a recipient-name which denotes a DL. See 8.2.1.1.1.6 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-dl-expansion-prohibited ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       DLExpansionProhibited,
    EQUALITY MATCHING-RULE      integerMatch,
    NUMERATION                  single-valued,
    ID                          id-att-dl-expansion-prohibited }
```

11.2.27 Entry-type

This general-attribute identifies the type of an entry (see 6.3.6). It is generated by the MS.

```
ms-entry-type ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       EntryType,
    EQUALITY MATCHING-RULE      integerMatch,
    ORDERING MATCHING-RULE      integerOrderingMatch, -- rule not defined in 1988 Application Contexts --
    NUMERATION                  single-valued,
    ID                          id-att-entry-type }
```

11.2.28 Internal-trace-information

This general-attribute, which is multi-valued, documents the actions taken on the message (or probe or report) by each MTA through which it passed as it transferred within an MD. It is generated from the Message-delivery-envelope and the Report-delivery-envelope. See 12.2.1.1.1.4 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-internal-trace-information ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       InternalTraceInformationElement,
    NUMERATION                  multi-valued,
    ID                          id-att-internal-trace-information }
```

11.2.29 Latest-delivery-time

This general-attribute contains the *latest-delivery-time* argument of the Message-submission abstract-operation; it indicates the time after which a message shall not be delivered to its recipient(s). See 8.2.1.1.1.13 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-latest-delivery-time ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       LatestDeliveryTime,
    EQUALITY MATCHING-RULE      uTCTimeMatch,
    ORDERING MATCHING-RULE      uTCTimeOrderingMatch,
    NUMERATION                  single-valued,
    ID                          id-att-latest-delivery-time }
```

11.2.30 Marked-for-deletion

This general-attribute indicates (by its presence or absence) whether the MS-user has marked the entry for subsequent deletion. Support for this attribute implies that the MS-user shall be able to create it or amend it by means of the Modify abstract-operation and the Auto-modify auto-action.

```
ms-marked-for-deletion ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX      NULL,
    NUMERATION                 single-valued,
    ID                         id-att-marked-for-deletion }
```

NOTE – The MS-user may use this attribute-type to support a two-stage entry deletion function, i.e. a 'wastebasket'.

11.2.31 Message-delivery-envelope

This general-attribute contains the *message-delivery-envelope* component of the argument of the Message-delivery abstract-operation. See Figure 2 (Part 5) of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-message-delivery-envelope ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX      MessageDeliveryEnvelope,
    NUMERATION                 single-valued,
    ID                         id-att-message-delivery-envelope }
```

11.2.32 Message-delivery-time

This general-attribute contains the *message-delivery-time* argument of the Message-delivery abstract-operation. See 8.3.1.1.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

NOTE – There is no general-attribute corresponding to the delivery-time parameter of the Report-delivery abstract-operation, because in order to be useful, this delivery-time must be correlated with the name of the recipient the message was delivered to. Both items of information are present in the per-recipient-report-delivery-fields general-attribute.

```
mt-message-delivery-time ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX      MessageDeliveryTime,
    EQUALITY MATCHING-RULE     uTCTimeMatch,
    ORDERING MATCHING-RULE     uTCTimeOrderingMatch,
    NUMERATION                 single-valued,
    ID                         id-att-message-delivery-time }
```

11.2.33 Message-group-name

This general-attribute, which is multi-valued, identifies the message-group-names which the MS-user has attached to an entry (see 6.4). Support for this general-attribute implies that the MS-user shall be able to create or amend it by use of the Modify abstract-operation and the Auto-modify auto-action. An attempt to assign a message-group-name attribute-value which has not been registered previously by means of the Register-MS abstract-operation shall result in the generation of a message-group-error. Each value of the attribute shall be distinct.

```
ms-message-group-name ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX      MessageGroupName,
    EQUALITY MATCHING-RULE     mSStringListMatch,
    OTHER MATCHING-RULES       {mSSingleSubstringListMatch |
                                mSStringListElementsMatch |
                                mSSingleSubstringListElementsMatch |
                                valueCountMatch},
    NUMERATION                 multi-valued,
    ID                         id-att-message-group-name }
```

NOTE – The maximum number of group-name-parts which will be present in a message-group-name may be determined from the message-group-depth parameter of MS-bind-result (see 7.1.2, item h).

11.2.34 Message-identifier

This general-attribute contains an MTS-identifier that distinguishes this message or probe from all other submitted messages and probes, and delivered messages. It contains the *message-submission-identifier* or *probe-submission-identifier* result of the Message-submission or Probe-submission abstract-operation, or the *message-delivery-identifier* argument of the Message-delivery abstract-operation. It is generated by the MTS. See 8.2.1.1.2.1, 8.2.1.2.2.1, and 8.3.1.1.1.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-message-identifier ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX      MTSIdentifier,
    EQUALITY MATCHING-RULE     mTSIdentifierMatch, -- rule not defined in 1988 Application Contexts --
    NUMERATION                 single-valued,
    ID                         id-att-message-identifier }
```

11.2.35 Message-notes

This general-attribute, which is multi-valued, contains textual annotations specified by the MS-user. The meaning attached to message-notes is of concern only to the MS-user. Support for this attribute implies that the MS-user shall be able to create or amend it by means of the Modify abstract-operation and the Auto-modify auto-action.

NOTE – The attribute may contain keywords, annotations, or reminders chosen by the MS-user, but does not represent information conveyed by the MTS.

```
ms-message-notes ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX        GeneralString (SIZE (1..ub-message-notes-length)),
    EQUALITY MATCHING-RULE       mSStringMatch,
    SUBSTRINGS MATCHING-RULE     mSSubstringsMatch,
    NUMERATION                   multi-valued,
    ID                           id-att-message-notes }
```

11.2.36 Message-origin-authentication-check

This general-attribute contains the *message-origin-authentication-check* argument of the Message-submission and Message-delivery abstract-operations. It provides the recipient(s) of a message with a means of authenticating its origin and may be generated by the originator of the message. See 8.2.1.1.1.29 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-message-origin-authentication-check ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX        MessageOriginAuthenticationCheck,
    NUMERATION                   single-valued,
    ID                           id-att-message-origin-authentication-check }
```

11.2.37 Message-security-label

This general-attribute contains the *message-security-label* argument of the Message-submission, Probe-submission, Message-delivery, and Report-delivery abstract-operations. It comprises a set of security attributes which may include a security-policy-identifier, a security-classification, a privacy-mark, and a set of security-categories. See 8.2.1.1.1.30 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-message-security-label ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX        MessageSecurityLabel,
    NUMERATION                   single-valued,
    ID                           id-att-message-security-label }
```

11.2.38 Message-submission-envelope

This general-attribute contains the *envelope* component of the argument of the Message-submission abstract-operation. See Figure 2 (Part 3) of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-message-submission-envelope ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX        MessageSubmissionEnvelope,
    NUMERATION                   single-valued,
    ID                           id-att-message-submission-envelope }
```

11.2.39 Message-submission-time

This general-attribute indicates the time at which the MTS accepted responsibility for the message or probe. It contains the *message-submission-time* result of Message-submission, or the *probe-submission-time* result of Probe-submission, or the *message-submission-time* argument of Message-delivery. It is generated by the MTS. See 8.2.1.1.2.2 and 8.2.1.2.2.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-message-submission-time ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX        MessageSubmissionTime,
    EQUALITY MATCHING-RULE       uTCTimeMatch,
    ORDERING MATCHING-RULE       uTCTimeOrderingMatch,
    NUMERATION                   single-valued,
    ID                           id-att-message-submission-time }
```

11.2.40 Message-token

This general-attribute contains the *message-token* argument of the Message-delivery abstract-operation. It is generated by the originator of the message. See 8.2.1.1.1.26 of ITU-T Rec. X.411 | ISO/IEC 10021-4. In submitted-message entries, this information is contained in the per-recipient-message-submission-fields attribute.

```
mt-message-token ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       MessageToken,
    NUMERATION                  single-valued,
    ID                          id-att-message-token }
```

11.2.41 MS-originated

This general-attribute indicates whether the message represented by a submitted-message entry was submitted by the MS as a consequence of the performance of an auto-action, or whether it was submitted by the MS-user. The attribute is present in the former case and absent in the latter. It is generated by the MS.

```
ms-originated ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       NULL,
    NUMERATION                  single-valued,
    ID                          id-att-ms-originated }
```

11.2.42 MS-submission-error

This general-attribute may be present in entries of the Submission-log entry-class and indicates the error returned by the MS-message-submission or MS-probe-submission abstract-operation that caused the creation of the entry. It is generated by the MS.

```
ms-submission-error ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       SubmissionError,
    NUMERATION                  single-valued,
    ID                          id-att-ms-submission-error }

SubmissionError ::= CHOICE {
    submission-control-violated             [1]  NULL,
    originator-invalid                      [2]  NULL,
    recipient-improperly-specified          [3]  ImproperlySpecifiedRecipients,
    element-of-service-not-subscribed       [4]  NULL,
    inconsistent-request                    [11] NULL,
    security-error                          [12] SecurityProblem,
    unsupported-critical-function           [13] NULL,
    remote-bind-error                       [15] NULL,
    service-error                           [27] ServiceErrorParameter,
    message-group-error                     [30] MessageGroupErrorParameter,
    ms-extension-error                      [31] MSExtensionErrorParameter,
    entry-class-error                       [34] EntryClassErrorParameter }
```

Each component of submission-error corresponds to one of the errors defined for the MS-message-submission and MS-probe-submission abstract-operations. The tag numbers correspond to the remote error codes (see 8.3.1 and 8.3.2).

11.2.43 Original-EITs

This general-attribute, which is multi-valued, identifies the encoded-information-types in the content of the message as submitted, or specified in the envelope of a submitted probe. It is generated from the *original-encoded-information-types* argument of the Message-delivery, Report-delivery, Message-submission, and Probe-submission abstract-operations. See 8.2.1.1.1.33 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-original-EITs ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       MS-EIT,
    EQUALITY MATCHING-RULE      objectIdentifierMatch,
    NUMERATION                  multi-valued,
    ID                          id-att-original-EITs }
```

11.2.44 Originally-intended-recipient-name

This general-attribute contains the *originally-intended-recipient-name* argument of the Message-delivery abstract-operation, and is present if the message has been redirected. See 8.3.1.1.1.4 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-originally-intended-recipient-name ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ORName,
    EQUALITY MATCHING-RULE      oRNameMatch,
    OTHER MATCHING-RULES        {oRNameElementsMatch | oRNameSubstringElementsMatch |
                                oRNameSingleElementMatch},
    NUMERATION                  single-valued,
    ID                          id-att-originally-intended-recipient-name }
```

11.2.45 Originating-MTA-certificate

This general-attribute contains the *originating-MTA-certificate* result of the Message-submission abstract-operation. It is generated by the MTA to which the message was submitted. See 8.2.1.1.2.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-originating-MTA-certificate ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX      OriginatingMTACertificate,
    NUMERATION                 single-valued,
    ID                         id-att-originating-MTA-certificate }
```

11.2.46 Originator-certificate

This general-attribute contains the *originator-certificate* argument of the Message-submission, Probe-submission, and Message-delivery abstract-operations. It is generated by a trusted source (e.g. a certification-authority), and may be supplied by the originator of the message. See 8.2.1.1.1.25 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-originator-certificate ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX      OriginatorCertificate,
    NUMERATION                 single-valued,
    ID                         id-att-originator-certificate }
```

11.2.47 Originator-name

This general-attribute contains the *originator-name* argument of the Message-submission, Probe-submission, and Message-delivery abstract-operations. See 8.2.1.1.1.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-originator-name ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX      ORName,
    EQUALITY MATCHING-RULE     oRNameMatch,
    OTHER MATCHING-RULES       {oRNameElementsMatch | oRNameSubstringElementsMatch |
                               oRNameSingleElementMatch},
    NUMERATION                 single-valued,
    ID                         id-att-originator-name }
```

11.2.48 Originator-report-request

This general-attribute, which is multi-valued, contains the *originator-report-request* argument of the Message-submission and Probe-submission abstract-operations. See 8.2.1.1.1.22 of ITU-T Rec. X.411 | ISO/IEC 10021-4. The ordering of the values of this attribute matches the ordering of the values of the per-recipient-message-submission-fields attribute.

```
mt-originator-report-request ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX      OriginatorReportRequest,
    NUMERATION                 multi-valued,
    ID                         id-att-originator-report-request }
```

11.2.49 Originator-return-address

This general-attribute contains the *originator-return-address* argument of the Message-submission and Message-delivery abstract-operations; it indicates the postal-OR-address of the originator of the message. See 8.2.1.1.1.21 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-originator-return-address ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX      OriginatorReturnAddress,
    NUMERATION                 single-valued,
    ID                         id-att-originator-return-address }
```

11.2.50 Other-recipient-names

This general-attribute, which is multi-valued, contains the *other-recipient-names* argument of the Message-delivery abstract-operation. See 8.3.1.1.1.6 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-other-recipient-names ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX      ORName,
    EQUALITY MATCHING-RULE     oRNameMatch,
    OTHER MATCHING-RULES       {oRNameElementsMatch | oRNameSubstringElementsMatch |
                               oRNameSingleElementMatch},
    NUMERATION                 multi-valued,
    ID                         id-att-other-recipient-names }
```

11.2.51 Parent-sequence-number

This general-*attribute identifies the entry's parent-entry. It is generated by the MS. It is always present in a child-entry and is absent in a main-entry (see 6.3.4).

```
ms-parent-sequence-number ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       SequenceNumber,
    EQUALITY MATCHING-RULE      integerMatch,
    ORDERING MATCHING-RULE      integerOrderingMatch,
    NUMERATION                  single-valued,
    ID                          id-att-parent-sequence-number }
```

11.2.52 Per-message-indicators

This general-attribute comprises various arguments of the Message-submission and (for the first two arguments) the Probe-submission abstract-operations. These include the *implicit-conversion-prohibited, alternate-recipient-allowed, disclosure-of-other-recipients,* and *content-return-request* arguments. See Figure 2 (Part 10) of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-per-message-indicators ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       PerMessageIndicators,
    EQUALITY MATCHING-RULE      bitStringMatch,
    NUMERATION                  single-valued,
    ID                          id-att-per-message-indicators }
```

11.2.53 Per-recipient-message-submission-fields

This general-attribute, which is multi-valued, contains the *per-recipient-fields* component of the Message-submission-envelope. See Figure 2 (Part 7) of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-per-recipient-message-submission-fields ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       PerRecipientMessageSubmissionFields,
    NUMERATION                  multi-valued,
    ID                          id-att-per-recipient-message-submission-fields }
```

11.2.54 Per-recipient-probe-submission-fields

This general-attribute, which is multi-valued, contains the *per-recipient-fields* component of the Probe-submission-envelope. See Figure 2 (Part 8) of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-per-recipient-probe-submission-fields ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       PerRecipientProbeSubmissionFields,
    NUMERATION                  multi-valued,
    ID                          id-att-per-recipient-probe-submission-fields }
```

11.2.55 Per-recipient-report-delivery-fields

This general-attribute, which is multi-valued, contains the *per-recipient-fields* component of the Report-delivery-envelope. See Figure 2 (Part 9) of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-per-recipient-report-delivery-fields ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       PerRecipientReportDeliveryFields,
    NUMERATION                  multi-valued,
    ID                          id-att-per-recipient-report-delivery-fields }
```

11.2.56 Priority

This general-attribute contains the *priority* argument of the Message-submission and Message-delivery abstract-operations. If no value for the parameter is supplied in the Message-submission or Message-delivery abstract-operation then the attribute is generated with the default value for priority. See 8.2.1.1.1.8 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-priority ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       Priority,
    EQUALITY MATCHING-RULE      integerMatch,
    ORDERING MATCHING-RULE      integerOrderingMatch, -- rule not defined in 1988 Application Contexts --
    NUMERATION                  single-valued,
    ID                          id-att-priority }
```

11.2.57 Probe-origin-authentication-check

This general-attribute contains the *probe-origin-authentication-check* argument of the Probe-submission abstract-operation. It provides any MTA through which the probe is transferred with a means of authenticating its origin. It may be generated by the MS-user when originating the probe. See 8.2.1.2.1.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-probe-origin-authentication-check ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ProbeOriginAuthenticationCheck,
    NUMERATION                  single-valued,
    ID                          id-att-probe-origin-authentication-check }
```

11.2.58 Probe-submission-envelope

This general-attribute contains the *envelope* component of the argument of the Probe-submission abstract-operation. See Figure 2 (Part 3) of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-probe-submission-envelope ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ProbeSubmissionEnvelope,
    NUMERATION                  single-valued,
    ID                          id-att-probe-submission-envelope }
```

11.2.59 Proof-of-delivery-request

This general-attribute contains the *proof-of-delivery-request* argument of the Message-delivery abstract-operation. It may be generated by the originator of the message. In submitted-message entries, this information may be found in the per-recipient-message-submission-fields attribute. See 8.2.1.1.1.32 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-proof-of-delivery-request ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ProofOfDeliveryRequest,
    EQUALITY MATCHING-RULE      integerMatch, -- rule not defined in 1988 Application Contexts --
    NUMERATION                  single-valued,
    ID                          id-att-proof-of-delivery-request }
```

11.2.60 Proof-of-submission

This general-attribute contains the *proof-of-submission* result of the Message-submission abstract-operation. It provides the MS-user with proof of submission of a message to the MTS. See 8.2.1.1.2.4 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-proof-of-submission ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ProofOfSubmission,
    NUMERATION                  single-valued,
    ID                          id-att-proof-of-submission }
```

11.2.61 Recipient-names

This general-attribute, which is multi-valued, contains the *recipient-name* components specified in the *per-recipient-fields* components of the Message-submission-envelope and Probe-submission-envelope. It is generated by the MS. The ordering of the values of this attribute matches the ordering of the values of the per-recipient-message-submission-fields (or per-recipient-probe-submission-fields) attribute. See 8.2.1.1.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
ms-recipient-names ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ORName,
    EQUALITY MATCHING-RULE      oRNameMatch,
    OTHER MATCHING-RULES        {oRNameElementsMatch | oRNameSubstringElementsMatch |
                                oRNameSingleElementMatch},
    NUMERATION                  multi-valued,
    ID                          id-att-recipient-names }
```

11.2.62 Recipient-reassignment-prohibited

This general-attribute contains the *recipient-reassignment-prohibited* argument of the Message-submission and Probe-submission abstract-operations. It indicates whether a message or probe submitted by the MS-user may be redirected to a recipient-assigned-alternate-recipient registered by the intended-recipient. See 8.2.1.1.1.4 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-recipient-reassignment-prohibited ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       RecipientReassignmentProhibited,
    EQUALITY MATCHING-RULE      integerMatch,
    NUMERATION                  single-valued,
    ID                          id-att-recipient-reassignment-prohibited }
```

11.2.63 Redirection-history

This general-attribute, which is multi-valued, contains the history of recipient redirection(s) with reason(s) from the Message-delivery or Report-delivery abstract-operations. In the latter case, the redirections concern the delivery report itself rather than the report's subject message. See 8.3.1.1.1.5 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-redirection-history ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       Redirection,
    OTHER MATCHING-RULES        {redirectionOrDLExpansionMatch |
                                 redirectionOrDLExpansionElementsMatch |
                                 redirectionOrDLExpansionSubstringElementsMatch |
                                 redirectionReasonMatch},
    NUMERATION                  multi-valued,
    ID                          id-att-redirection-history }
```

11.2.64 Report-delivery-envelope

This general-attribute contains the *report-delivery-envelope* component of the argument of the Report-delivery abstract-operation. See Figure 2 (Part 9) of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-report-delivery-envelope ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ReportDeliveryEnvelope,
    NUMERATION                  single-valued,
    ID                          id-att-report-delivery-envelope }
```

11.2.65 Reporting-DL-name

This general-attribute contains the *reporting-DL-name* argument of the Report-delivery abstract-operation. It identifies the DL that forwarded the report to the owner of the DL. See 8.3.1.2.1.4 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-reporting-DL-name ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ReportingDLName,
    EQUALITY MATCHING-RULE      oRNameMatch, -- rule not defined in 1988 Application Contexts --
    OTHER MATCHING-RULES        {oRNameElementsMatch | oRNameSubstringElementsMatch |
                                 oRNameSingleElementMatch},
    NUMERATION                  single-valued,
    ID                          id-att-reporting-DL-name }
```

11.2.66 Reporting-MTA-certificate

This general-attribute contains the *reporting-MTA-certificate* argument of the Report-delivery abstract-operation. It contains the certificate of the MTA that generated the report. See 8.3.1.2.1.12 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-reporting-MTA-certificate ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ReportingMTACertificate,
    NUMERATION                  single-valued,
    ID                          id-att-reporting-MTA-certificate }
```

11.2.67 Report-origin-authentication-check

This general-attribute contains the *report-origin-authentication-check* argument of the Report-delivery abstract-operation. It provides a means of authenticating the origin of the report. See 8.3.1.2.1.13 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-report-origin-authentication-check ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ReportOriginAuthenticationCheck,
    NUMERATION                  single-valued,
    ID                          id-att-report-origin-authentication-check }
```

11.2.68 Retrieval-status

This general-attribute indicates the retrieval-status of an entry (see 6.3.8). If the Modify abstract-operation is available, the MS-user shall be able to amend the attribute, provided that its existing value is *listed* and the replacement value is *processed*. The attribute is not subject to modification by means of the Auto-modify auto-action. In an entry of the Message-log entry-class, the attribute reflects the retrieval-status of the corresponding Stored-message entry. In an entry of the Auto-action-log entry-class, the attribute indicates the retrieval-status of the entry itself (not that of the auto-action-subject-entry). The attribute is created and maintained by the MS.

```
ms-retrieval-status ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX        RetrievalStatus,
    EQUALITY MATCHING-RULE       integerMatch,
    NUMERATION                   single-valued,
    ID                           id-att-retrieval-status }
```

11.2.69 Security-classification

This general-attribute comprises the *security-classification* component of the message-security-label. It is defined as a separate attribute to allow its use in the Summarize abstract-operation. See 8.5.9 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-security-classification ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX        SecurityClassification,
    EQUALITY MATCHING-RULE       integerMatch,
    NUMERATION                   single-valued,
    ID                           id-att-security-classification }
```

11.2.70 Sequence-number

This general-attribute identifies the entry containing the attribute. It is allocated by the MS when the entry is created (see 6.3.2).

```
ms-sequence-number ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX        SequenceNumber,
    EQUALITY MATCHING-RULE       integerMatch,
    ORDERING MATCHING-RULE       integerOrderingMatch,
    NUMERATION                   single-valued,
    ID                           id-att-sequence-number }
```

11.2.71 Storage-period

This general-attribute indicates the period, in seconds, for which the MS-user anticipates that storage of the entry will be required, relative to its creation-time. An entry becomes subject to deletion by the Auto-delete auto-action when its storage-period expires (see 13.4). An MS that supports this general-attribute shall support its creation and amendment by the Modify abstract-operation and the Auto-modify auto-action; it shall also support the storage-time general-attribute to which this general-attribute is semantically related. A given value of the storage-period attribute is equal to the value of the corresponding storage-time minus the creation-time, expressed in seconds. The creation, modification, or deletion of the storage-period general-attribute is reflected in the creation, modification, or deletion of the storage-time general-attribute; i.e. the semantic link between the two is always preserved.

```
ms-storage-period ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX        StoragePeriod,
    EQUALITY MATCHING-RULE       integerMatch,
    ORDERING MATCHING-RULE       integerOrderingMatch,
    NUMERATION                   single-valued,
    ID                           id-att-storage-period }

StoragePeriod ::= INTEGER        -- seconds --
```

NOTE – This attribute is intended for use with the Auto-modify auto-action, to enable the MS-user to add a storage-time equal to a fixed period after entry creation to selected entries. This is not possible using storage-time directly as that attribute contains an absolute time.

11.2.72 Storage-time

This general-attribute indicates the date and time at which the MS-user estimates that storage for the entry will no longer be required. An entry becomes subject to deletion by the Auto-delete auto-action when its storage-time is reached (see 13.4). An MS that supports this general-attribute shall support its creation and amendment by the Modify abstract-operation and the Auto-modify auto-action; it shall also support the storage-period general-attribute to which this general-attribute is semantically related. The creation, modification, or deletion of the storage-time general-attribute is reflected in the creation, modification, or deletion of the storage-period general-attribute; i.e. the semantic link between the two is always preserved.

```
ms-storage-time ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX        StorageTime,
    EQUALITY MATCHING-RULE       uTCTimeMatch,
    ORDERING MATCHING-RULE       uTCTimeOrderingMatch,
    NUMERATION                   single-valued,
    ID                           id-att-storage-time }

StorageTime ::= UTCTime
```

11.2.73 Subject-submission-identifier

This general-attribute contains the *subject-submission-identifier* argument of the Report-delivery abstract-operation. It indicates the *message-submission-identifier* or the *probe-submission-identifier* of the subject of the report. See 8.3.1.2.1.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-subject-submission-identifier ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       SubjectSubmissionIdentifier,
    EQUALITY MATCHING-RULE      mTSIdentifierMatch, -- rule not defined in 1988 Application Contexts --
    NUMERATION                  single-valued,
    ID                          id-att-subject-submission-identifier }
```

11.2.74 This-recipient-name

This general-attribute contains the *this-recipient-name* argument of the Message-delivery abstract-operation, and identifies the MS-user. See 8.3.1.1.1.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-this-recipient-name ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       ORName,
    EQUALITY MATCHING-RULE      oRNameMatch,
    OTHER MATCHING-RULES        {oRNameElementsMatch | oRNameSubstringElementsMatch |
                                oRNameSingleElementMatch},
    NUMERATION                  single-valued,
    ID                          id-att-this-recipient-name }
```

11.2.75 Trace-information

This general-attribute, which is multi-valued, documents the actions taken on the message (or probe or report) by each MD through which it passed as it transferred through the MTS. It is generated from the Message-delivery-envelope and the Report-delivery-envelope. See 12.2.1.1.1.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

```
mt-trace-information ATTRIBUTE ::= {
    WITH ATTRIBUTE-SYNTAX       TraceInformationElement,
    NUMERATION                  multi-valued,
    ID                          id-att-trace-information }
```

11.3 The Attribute-table information object set

The **Attribute-table** information object set is used as a constraining set in this Service Definition where related fields of the ATTRIBUTE information object class are referenced in the MS abstract-syntax. It comprises two object sets:

```
AttributeTable ATTRIBUTE ::= {
    GeneralAttributes | ContentSpecificAttributes }
```

The **General-attributes** information object set contains the general-attributes defined in this Service Definition. It is defined in Annex C. An MS implementation is not constrained to support only those general-attributes present in the General-attributes object set.

The **Content-specific-attributes** information object set is an empty, extensible set. Where an MS offers support for a given content-type, the attribute-types associated with that content-type shall be regarded as objects populating the Content-specific-attributes object set.

```
ContentSpecificAttributes ATTRIBUTE ::= { ... }
```

11.4 Generation of the general-attributes

This subclause describes how the general-attributes are generated. The information is summarized in Table 4; definitive attribute descriptions are given in 11.2. For a description of the classifications used see 5.3.

11.5 Attribute-types subscription

Attribute-type subscription is a local matter. If the attribute-type subscription is changed so that fewer attribute-types are subscribed to, then the MS-user may continue to have access to all attributes originally subscribed to for messages present in the MS at the time the subscription was changed. The handling of attributes for which subscription has been cancelled is a local matter. Similarly, when a new attribute is subscribed to, the MS-user may not have access to this attribute for messages present in the MS at the time the subscription was changed.

Table 4 – Generation of the General-attribute-types

Attribute-type name	Single/ multi-valued	Source parameter	Source generated by	Generation rules
AC-correlated-report-list	M	–	MS	A value is generated for each originally specified recipient, identifying the per-recipient-report-delivery-fields which have been correlated with that recipient.
AC-report-summary	M	–	MS	The initial values are the values of originator-report-request. They are updated as reports are received. One value is generated for each originally specified recipient.
AC-uncorrelated-report-list	M	–	MS	A value is generated for each per-recipient-report-delivery-fields which cannot be correlated with an originally specified recipient.
Auto-action-error	S	–	MS	The attribute-value is the value of the error generated as the result of auto-action processing.
Auto-action-registration-identifier	S	–	MS	The attribute-value is the registration-identifier of a registered auto-action executed by the MS.
Auto-action-subject	S	–	MS	The attribute-value is the sequence-number of an entry to which auto-action processing was applied.
Auto-action-type	S	–	MS	The attribute-value is the type of a registered auto-action executed by the MS.
Child-sequence-numbers	M	–	MS	A value is generated for each child-entry belonging to a parent-entry.
Content	S	content returned-content	Ms, Md Rd	The attribute-value in the delivered or submitted message entry is the value of the source parameter. The attribute-value in the returned-content child-entry of the delivered-report entry is the value of the source parameter.
Content-confidentiality-algorithm-identifier	S	content-confidentiality-algorithm-identifier	Ms, Md	The attribute-value is the value of the source parameter.
Content-correlator	S	content-correlator	Ms, Ps, Rd	The attribute-value is the value of the source parameter.
Content-identifier	S	content-identifier	Ms, Ps, Md, Rd	The attribute-value is the value of the source parameter.
Content-integrity-check	S	content-integrity-check	Md	The attribute-value is the value of the source parameter.
Content-length	S	– content-length	MS Ps	The (approximate) size of the content in octets. The attribute-value is the value of the source parameter.
Content-returned	S	–	MS	The value is set to true if returned-content is present in a Report-delivery and to false if not present.
Content-type	S	content-type	Ms, Ps, Md, Rd	If represented by Object Identifier, the value of the source parameter. If represented by Integer, converted to the corresponding Object Identifier.
Conversion-with-loss-prohibited	S	conversion-with-loss-prohibited	Ms, Ps, Md	The attribute-value is the value of the source parameter.
Converted-EITs	M	converted-encoded-information-types	Md	A corresponding value is generated from each bit that is set to 1 in the built-in-encoded-information-types component of, and from each extended-encoded-information-type present in the converted-encoded-information-types parameter.
Creation-time	S	–	MS	The time of creation of the entry.
Deferred-delivery-cancellation-time	S	–	MS	The time at which the Cancel-deferred-delivery abstract-operation was performed.
Deferred-delivery-time	S	deferred-delivery-time	Ms	The attribute-value is the value of the source parameter.

Table 4 *(continued)* – Generation of the General-attribute-types

Attribute-type name	Single/ multi-valued	Source parameter	Source generated by	Generation rules
Deletion-time	S	–	MS	Generated when the entry to which this Message-log entry corresponds is deleted.
Delivered-EITs	M	converted-encoded-information-types, if present; original-encoded-information-types otherwise	MS	Converted-EITs if converted-encoded-information-types is present; Original-EITs otherwise.
Delivery-flags	S	delivery-flags	Md	The attribute-value is the value of the source parameter. If there are no delivery-flags in the Md, generate a default value with no flags set.
DL-expansion-history	M	dl-expansion-history originator-and-dl-expansion-history	Md Rd	A corresponding value is generated from each component of the SEQUENCE. A corresponding value is generated from each component of the SEQUENCE.
DL-expansion-prohibited	S	dl-expansion-prohibited	Ms, Ps	The attribute-value is the value of the source parameter.
Entry-type	S	Md Argument Rd Argument Ms Argument Ps Argument Ms Argument	MS MS MS MS MS MS	The value is "delivered-message". The value is "delivered-report". If a returned-content is present, a child-entry which contains the returned-content is created and given an entry-type of "returned-content". The value is "submitted-message". The value is "submitted-probe" The value is "draft-message" The value is "auto-action-event"
Internal-trace-information	M	internal-trace-information	Md Rd	A corresponding value is generated from each component of the SEQUENCE.
Latest-delivery-time	S	latest-delivery-time	Ms	The attribute-value is the value of the source parameter.
Marked-for-deletion	S	Modify Argument, Auto-modify Argument	Mod, Amod	Generated by the Modify abstract-operation or the Auto-modify auto-action.
Message-delivery-envelope	S	MessageDelivery Envelope	Md	The attribute-value is the value of the source parameter.
Message-delivery-time	S	message-delivery-time	Md	The attribute-value is the value of the source parameter.
Message-group-name	M	Modify Argument, Auto-modify Argument	Mod, Amod	Generated by the Modify abstract-operation or the Auto-modify auto-action.
Message-identifier	S	message-submission-identifier, probe-submission-identifier, message-delivery-identifier	Ms, Ps, Md	The attribute-value is the value of the source parameter.
Message-notes	M	Modify Argument, Auto-modify Argument	Mod, Amod	Generated by the Modify abstract-operation or the Auto-modify auto-action.
Message-origin-authentication-check	S	message-origin-authentication-check	Ms, Md	The attribute-value is the value of the source parameter.
Message-security-label	S	message-security-label	Ms, Ps, Md, Rd	The attribute-value is the value of the source parameter.

Table 4 *(continued)* – Generation of the General-attribute-types

Attribute-type name	Single/ multi-valued	Source parameter	Source generated by	Generation rules
Message-submission-envelope	S	Message-Submission-Envelope	Ms	The attribute-value is the value of the source parameter.
Message-submission-time	S	message-submission-time, probe-submission-time	Ms, Md Ps	The attribute-value is the value of the source parameter. The attribute-value is the value of the source parameter.
Message-token	S	message-token	Md	The attribute-value is the value of the source parameter.
MS-originated	S	–	MS	Set *true* if the MS originated this message in auto-action processing.
MS-submission-error	S	MS-message-submission error, MS-probe-submission error	MS	Generated from the ERROR returned by the MS-message-submission or MS-probe-submission abstract-operations.
Original-EITs	M	original-encoded-information-types	Ms, Ps, Md, Rd	A corresponding value is generated from each bit that is set to 1 in the built-in-encoded-information-types component of, and from each extended-encoded-information-type present in the original-encoded-information-types parameter.
Originally-intended-recipient-name	S	originally-intended-recipient-name	Md	The attribute-value is the value of the source parameter.
Originating-MTA-certificate	S	originating-mta-certificate	Ms	The attribute-value is the value of the source parameter
Originator-certificate	S	originator-certificate	Ms, Ps, Md	The attribute-value is the value of the source parameter.
Originator-name	S	originator-name	Ms, Ps, Md	The attribute-value is the value of the source parameter.
Originator-report-request	M	originator-report-request	Ms, Ps	The attribute-value is the value of the source parameter
Originator-return-address	S	originator-return-address	Ms, Md	The attribute-value is the value of the source parameter
Other-recipient-names	M	other-recipient-names	Md	A corresponding value is generated from each component of the SEQUENCE.
Parent-sequence-number	S	–	MS	When creating a child-entry, this attribute is generated with the corresponding parent-entry's sequence-number as value.
Per-message-indicators	S	per-message-indicators	Ms, Ps	The attribute-value is the value of the source parameter. If absent from Ms or Ps, generate a zeroed bit string by default.
Per-recipient-message-submission-fields	M	PerRecipientMessageSubmissionFields	Ms	The attribute-value is the value of the source parameter
Per-recipient-probe-submission-fields	M	PerRecipientProbeSubmissionFields	Ps	The attribute-value is the value of the source parameter
Per-recipient-report-delivery-fields	M	per-recipient-fields	Rd	A corresponding value is generated from each component of the SEQUENCE.
Priority	S	priority	Ms, Md	The attribute-value is the value of the source parameter.
Probe-origin-authentication-check	S	probe-origin-authentication-check	Ps	The attribute-value is the value of the source parameter.
Probe-submission-envelope	S	ProbeSubmission-Envelope	Ps	The attribute-value is the value of the source parameter
Proof-of-delivery-request	S	proof-of-delivery-request	Md	The attribute-value is the value of the source parameter.

Table 4 *(concluded)* – Generation of the General-attribute-types

Attribute-type name	Single/ multi-valued	Source parameter	Source generated by	Generation rules
Proof-of-submission	S	proof-of-submission	Ms	The attribute-value is the value of the source parameter
Recipient-names	M	recipient-name	Ms, Ps	The attribute-value is the value of the source parameter
Recipient-reassignment-prohibited	S	recipient-reassignment-prohibited	Ms, Ps	The attribute-value is the value of the source parameter
Redirection-history	M	redirection-history	Md, Rd	A corresponding value is generated from each component of the SEQUENCE.
Report-delivery-envelope	S	ReportDelivery Envelope	Rd	The attribute-value is the value of the source parameter.
Reporting-DL-name	S	reporting-dl-name	Rd	The attribute-value is the value of the source parameter.
Reporting-MTA-certificate	S	reporting-MTA-certificate	Rd	The attribute-value is the value of the source parameter.
Report-origin-authentication-check	S	report-origin-authentication-check	Rd	The attribute-value is the value of the source parameter.
Retrieval-status	S	–	MS	Generated when the entry is created; given the value 'new'.
Security-classification	S	security-classification	Md, Rd	The attribute-value is the value of the source parameter.
Storage-period	S	Modify Argument, Auto-modify Argument	Mod, Amod	Generated by the Modify abstract-operation or the Auto-modify auto-action. Storage-time is generated at the same time.
Storage-time	S	Modify Argument, Auto-modify Argument	Mod, Amod	Generated by the Modify abstract-operation or the Auto-modify auto-action. Storage-period is generated at the same time.
Sequence-number	S	–	MS	When creating an entry, the MS assigns a unique value for this attribute in ascending order.
Subject-submission-identifier	S	subject-submission-identifier	Rd	The attribute-value is the value of the source parameter.
This-recipient-name	S	this-recipient-name	Md	The attribute-value is the value of the source parameter.
Trace-information	M	trace-information	Md Rd	A corresponding value is generated from each component of the SEQUENCE.

11.6 General-attribute-types subject to modification

Of the general-attribute-types, only those listed below are subject to modification by the Modify abstract-operation, and, except for retrieval-status, by the Auto-modify auto-action. With the exception of retrieval-status, where an MS supports one of these attributes, it shall support its modification by the Modify abstract-operation.

a) Marked-for-deletion

b) Message-group-name

c) Message-notes

d) Retrieval-status

e) Storage-period

f) Storage-time

Details of content-specific attributes that are subject to modification are given in the Specification which defines the content-type concerned.

12 General matching-rules

The **general-matching-rules** may be applied to the attributes of an entry of any content-type. Other matching-rules, which are content-specific, are defined in the relevant Specification. Some of the rules are based on matching-rules defined in ITU-T Rec. X.501 | ISO/IEC 9594-2 and ITU-T Rec. X.520 | ISO/IEC 9594-6. The MATCHING-RULE information object class used in this Service Definition is described in 6.3.9.3.

12.1 MS-string syntax

The attribute-syntax of some attribute-types allows the same information to be represented in different forms:

- as either a Numeric String or Printable String;
- as either a Printable String or Teletex String.

For the standard attributes of OR-address, these alternative forms of representation are listed in Table 9 of ITU-T Rec. X.402 | ISO/IEC 10021-2. It is convenient to define an ASN.1 syntax for specifying these optional forms in matching-rule-assertions:

```
MSString { INTEGER : maxSize } ::= CHOICE {
    printable                   PrintableString (SIZE (1..maxSize)),
    teletex                     TeletexString (SIZE (1..maxSize)),
    general                     GeneralString (SIZE (1..maxSize)),
    universal                   UniversalString (SIZE (1..maxSize)),
    bmp                         BMPString (SIZE (1..maxSize))}
```

When the MS-string type is used as the assertion-syntax in a matching-rule-assertion, the choice of string types is confined to that represented in the attribute-syntax of the attribute under test.

Constraints on the set of Universal String characters that may be present in an MS-string will be laid down in Specifications which define attribute-types with an attribute-syntax of Universal String. No such attribute-types are defined in this Service Definition.

12.2 String matching-rules

In the matching-rules specified in 12.2.1 to 12.2.9, the following spaces are not regarded as significant:

- leading spaces (i.e. those preceding the first printing character);
- trailing spaces (i.e. those following the last printing character);
- multiple consecutive internal spaces (these are taken as equivalent to a single space character).

In the matching-rules to which these apply, the strings to be matched shall be matched as if the insignificant spaces were not present in either string. A value which contains only space characters is regarded as containing a single significant space.

The following rules also apply:

- the Non-spacing underline graphic character shall be considered insignificant, as shall all control functions except Space and those used for code extension procedures;
- the choice between different encodings of the same character shall be considered insignificant.

Where the strings being matched are of different ASN.1 character string types (e.g. a presented Printable String and a stored Teletex String), the comparison proceeds as normal so long as the corresponding characters are in both character sets; the comparison is based on the equivalence of the characters and the choice of encoding is ignored. Otherwise matching fails.

12.2.1 MS-string-match

The **MS-string-match** compares for equality a presented string with attribute-values of type MS-string without regard to the case (upper or lower) of the strings (e.g. "Dundee" and "DUNDEE" match).

NOTE – In some languages, the concept of distinct upper-case and lower-case letters does not exist.

```
mSStringMatch MATCHING-RULE ::= {
    SYNTAX                  MSString {ub-msstring-match}
    ID                      id-mr-msstring-match }
```

The rule returns *true* if, and only if, the strings have the same number of characters and corresponding characters are identical (except that the case shall be ignored). Where the attribute-type allows the choice between different ASN.1 character string types, the comparison is based on the equivalence of the characters and the choice of encoding is ignored.

12.2.2 MS-string-ordering-match

The **MS-string-ordering-match** compares the collation order of a presented string with attribute-values of type MS-string without regard to the case (upper or lower) of the strings.

```
mSStringOrderingMatch MATCHING-RULE ::= {
    SYNTAX                  MSString {ub-msstring-match}
    ID                      id-mr-msstring-ordering-match }
```

The rule returns *true* if, and only if, some value of the attribute is 'less' or appears earlier than the presented value, when the strings are compared using the normal collation order for their syntax after lower-case letters in both strings have been replaced by their upper-case equivalents.

NOTES

1 For example, the rule returns *true* where a presented value of "Falkirk" is compared with an attribute containing the values "Glasgow" and "Edinburgh", since the value "Edinburgh" is 'less than' the presented value.

2 The definition of normal collation order may be the subject of National Decision.

12.2.3 MS-substrings-match

The **MS-substrings-match** determines whether a presented value is a substring of some value of an attribute of type MS-string without regard to the case (upper or lower) of the strings.

```
mSSubstringsMatch MATCHING-RULE ::= {
    SYNTAX      SubstringAssertion
    ID          id-mr-ms-substrings-match }

SubstringAssertion ::= SEQUENCE OF CHOICE {
    initial     [0] MSString {ub-msstring-match},
    any         [1] MSString {ub-msstring-match},
    final       [2] MSString {ub-msstring-match} }
    -- at most one initial and one final component --
```

For a substring value to match, the specified substrings must appear in the asserted order. The substrings shall be non-overlapping, and may (but need not) be separated from the ends of the attribute-value and from one another by zero or more string elements.

If **initial** is present, the substring shall match the initial string of the value; if **final** is present, the substring shall match the final substring of the value; if **any** is present, the substring shall match any substring in the value. Each substring is matched according to the MS-string-match rule.

For a component of substring-assertion to match a portion of the attribute-value, corresponding characters must be identical, except in regard to case. Where the attribute-type allows the choice between different ASN.1 character string types, the comparison is based on the equivalence of the characters and the choice of encoding is ignored.

NOTE – For example, the rule returns *true* where a presented value containing initial "mes", any "age" and final "orc" is compared with an attribute containing the value "Message Store".

12.2.4 MS-single-substring-match

The **MS-single-substring-match** determines whether a presented string is a single substring of some value of an attribute of type MS-string without regard to the case (upper or lower) of the strings.

```
mSSingleSubstringMatch MATCHING-RULE ::= {
    SYNTAX    MSString {ub-msstring-match}
    ID        id-mr-ms-single-substring-match }
```

The rule is identical to the MS-substrings-match rule, except that the presented value is treated as the sole **any** component of the substring-assertion; the **initial** and **final** components are absent.

NOTE – For example, the rule returns *true* when a presented value of "Age" is compared with an attribute containing the value "Message Store".

12.2.5 MS-string-case-sensitive-match

The **MS-string-case-sensitive-match** compares for equality a presented string with attribute-values of type MS-string.

```
mSStringCaseSensitiveMatch MATCHING-RULE ::= {
    SYNTAX    MSString {ub-msstring-match}
    ID        id-mr-msstring-case-sensitive-match }
```

The rule is identical to the MS-string-match rule except that case is not ignored.

NOTE – For example, the rule returns *true* for a presented value of "CaSe" if and only if that value is compared with an attribute containing the value "CaSe".

12.2.6 MS-string-list-match

The **MS-string-list-match** compares for equality a presented sequence of strings with attribute-values which are sequences of MS-string without regard to the case (upper or lower) of the strings.

```
mSStringListMatch MATCHING-RULE ::= {
    SYNTAX    SEQUENCE OF MSString {ub-msstring-match}
    ID        id-mr-msstring-list-match }
```

The rule returns *true* if, and only if, the number of strings in each is the same, and corresponding strings match using the MS-string-match matching-rule.

NOTE – For example, the rule returns *true* when a presented value of "ms projects+milestones+announcements" is compared with an attribute containing the value "MS Projects+Milestones+Announcements" (where "+" separates the strings in the sequence).

12.2.7 MS-string-list-elements-match

The **MS-string-list-elements-match** determines whether a presented sequence of strings is a leading subset of the sequence of strings present in some value of an attribute containing sequences of MS-string without regard to the case (upper or lower) of the strings.

```
mSStringListElementsMatch MATCHING-RULE ::= {
    SYNTAX    SEQUENCE OF MSString {ub-msstring-match}
    ID        id-mr-msstring-list-elements-match }
```

The rule is identical to the MS-string-list-match rule except that strings are matched in the order presented in the sequence (beginning with the first string in each sequence), and the rule returns *true* when all strings in the presented sequence match (i.e. the number of strings in the presented sequence shall be less than or equal to the number of strings in the stored value).

NOTE – For example, the rule returns *true* when a presented value of "ms projects+milestones" is compared with an attribute containing the value "MS Projects+Milestones+Announcements" (where "+" separates the strings in the sequence).

12.2.8 MS-single-substring-list-match

The **MS-single-substring-list-match** determines whether each string in a presented sequence is a single substring of each corresponding string in a sequence that constitutes some value of an attribute containing sequences of MS-string, without regard to the case (upper or lower) of the strings.

```
mSSingleSubstringListMatch MATCHING-RULE ::= {
    SYNTAX    SEQUENCE OF MSString {ub-msstring-match}
    ID        id-mr-ms-single-substring-list-match }
```

The rule returns *true* if, and only if, the number of strings in each is the same, and corresponding strings match using the MS-single-substring-match rule.

NOTE – For example, the rule returns *true* when a presented value of "projects+miles+ounce" is compared with an attribute containing the value "MS Projects+Milestones+Announcements" (where "+" separates the strings in the sequence).

12.2.9 MS-single-substring-list-elements-match

The **MS-single-substring-list-elements-match** determines whether a presented sequence of single substrings is a subset of the sequence of strings present in some value of an attribute containing sequences of MS-string, without regard to the case (upper or lower) of the strings.

```
mSSingleSubstringListElementsMatch MATCHING-RULE ::= {
    SYNTAX    SEQUENCE OF MSString {ub-msstring-match}
    ID        id-mr-ms-single-substring-list-elements-match }
```

The rule is identical to the MS-single-substring-list-match rule except that strings are matched in the order presented in the sequence (beginning with the first string in each sequence), and the rule returns *true* when all strings in the presented sequence match (i.e. the number of strings in the presented sequence shall be less than or equal to the number of strings in the stored value).

NOTE – For example, the rule returns *true* when a presented value of "ms+stones" is compared with an attribute containing the value "MS Projects+Milestones+Announcements" (where "+" separates the strings in the sequence).

12.3 Syntax-based matching-rules

This Service Definition makes use of the following syntax-based matching-rules defined in ITU-T Rec. X.501 | ISO/IEC 9594-2 and ITU-T Rec. X.520 | ISO/IEC 9594-6: Bit String Match, Object Identifier Match, Distinguished Name Match, Boolean Match, Integer Match, Integer Ordering Match, Octet String Match, UTC Time Match, UTC Time Ordering Match, and Presentation Address Match. In addition to these a Value-count matching-rule is defined.

NOTE 1 – The Integer Match rule defined in ITU-T Rec. X.520 | ISO/IEC 9594-6 is also applied here to Enumerated types.

The **value-count-match** determines whether the number of values present in an attribute of an entry is equal to a presented Integer.

```
valueCountMatch MATCHING-RULE ::=
    SYNTAX    INTEGER (1..ub-attribute-values)
    ID        id-mr-value-count-match }
```

The rule returns *true* if, and only if, the nominated attribute contains a number of values equal to the presented Integer.

NOTE 2 – This rule could be used in a filter that selects entries which belong to a given message-group to identify those entries which belong only to that message-group.

12.4 Matching-rules for complex Message Store attributes

The components of an OR-address are defined in clause 18 of ITU-T Rec. X.402 | ISO/IEC 10021-2 as **standard attribute types** and **domain defined attribute types** (but are not formally defined as instances of the ATTRIBUTE information object class). Many standard attribute types are complex data-types (e.g. personal-name) and are not always suitable for matching as complete units. For this reason, the standard attribute types are broken into their primitive components which are referred to as **OR-address elements** in this Service Definition.

NOTE – Thus the standard attribute type *personal-name* comprises the four OR-address elements *surname, given-name, initials,* and *generation-qualifier.*

12.4.1 OR-address-match

The **OR-address-match** rule compares for equality a presented value with attribute-values of type OR-address.

```
oRAddressMatch MATCHING-RULE ::= {
    SYNTAX    ORAddress
    ID        id-mr-oraddress-match }
```

The rule returns *true* if the presented value and at least one value of the stored attribute both contain the same number of OR-address elements of the same type, and corresponding elements match. Otherwise matching fails.

Matching of elements is performed according to the following rules, which reflect the rules for Attribute List Equivalence defined in ITU-T Rec. X.402 | ISO/IEC 10021-2:

a) All elements are matched using the MS-string-match rule, except for:

   1) Terminal-type, which is matched using Integer Match.

   2) The extended form of network-address. The E163-4-address alternative is matched using the MS string-list-match rule. The PSAP-address alternative is matched using the Presentation Address Match.

b) Where the value of an element may be a Numeric String or an equivalent Printable String, either form of presented value shall match either form of stored value.

   NOTE 1 – For administration-domain-name, private-domain-name, and postal-code the same numeric value may be represented as either a Numeric or Printable String.

c) The equivalence rule between the X.121 and ISO 3166 forms of the country-name element shall apply (see ITU-T Rec. X.402 | ISO/IEC 10021-2).

   NOTE 2 – As a local matter, where X.121 allocates more than one number to a country, these may be regarded as equivalent.

d) Where the value of an element may be a Printable String or an equivalent Teletex String, or both, either form of presented value shall match either form of stored value. Where both forms are present (in the presented value, or stored value, or both) then matching of the element succeeds if the matching of at least one form succeeds.

   NOTE 3 – For example, the organization-name element may be present as a built-in-standard-attribute (a Printable String), or as an extension-attribute (a Teletex String), or as both. Similarly, the common-name element may be present as either, or both, of two extension-attributes: common-name (a Printable String), and teletex-common-name (a Teletex String).

e) The type and value components of a domain-defined-attribute element may be a Printable String or an equivalent Teletex String value pair; either form of presented value pair shall match either form of stored value pair.

f) The order of elements is not significant, except for the elements derived from organizational-unit-names, which must appear in the same order in the presented value and stored value.

NOTE 4 – As a local matter, where an MS has specific knowledge of certain domain-defined-attributes, it may apply additional rules which attach significance to the punctuation characters, the case of letters, or the relative order of those domain-defined-attributes.

NOTE 5 – An MD may impose additional rules for matching based on local equivalence rules upon the attributes it assigns to its own users and DLs. The MS may utilize local information about the equivalence of attribute-values or combinations of values to determine whether a supplied OR-address unambiguously identifies a stored OR-address.

12.4.2 OR-address-elements-match

The **OR-address-elements-match** rule determines whether a presented value is a subset of the elements present in some value of an attribute of type OR-address.

```
oRAddressElementsMatch MATCHING-RULE ::= {
    SYNTAX                          ORAddress
    ID                              id-mr-oraddress-elements-match }
```

The rule returns *true* if, and only if, at least one value of the attribute includes those elements which comprise the presented value, and the values of the presented elements match those of the corresponding elements in the stored value. Matching of elements is performed according to the rules defined for the OR-address-match rule.

NOTE 1 – Care should be taken when constructing a complex filter. Where a filter consists of two filter-items, each containing a subset of OR-address elements, and the **and** operator (e.g. "C=xx **and** O=zz") each filter-item might match on different attribute-values. Consequently an entry could be selected even where no single attribute-value contains both elements; this occurs if one attribute-value contains C=xx and another O=zz. This is avoided if both elements are presented in the same OR-address-elements-match.

For the complex components (personal-name, organizational-unit-names, domain-defined-attributes, and the extended form of network-address) only those elements presented are considered.

NOTE 2 – This allows a match to proceed on a part of a complex component, e.g. "PersonalName.surname=Maruba".

12.4.3 OR-address-substring-elements-match

The **OR-address-substring-elements-match** rule determines whether a presented value is a subset of the elements present in some value of an attribute of type OR-address, where each presented string value is a substring of the corresponding stored value.

```
oRAddressSubstringElementsMatch MATCHING-RULE ::= {
    SYNTAX    ORAddress
    ID        id-mr-oraddress-substring-elements-match }
```

This rule is identical to the OR-address-elements-match rule except that for those elements which are matched using the MS-string-match rule, the MS-single-substring-match rule is applied.

NOTE – For example, the rule returns *true* when a presented value of "OrganizationName=rc", "PersonalName.surname=arthur" is compared with an attribute which includes the value "OrganizationName=RCC", "PersonalName.surname=McArthur".

12.4.4 OR-name-match

The **OR-name-match** rule compares for equality a presented value with attribute-values of type OR-name.

```
oRNameMatch MATCHING-RULE ::= {
    SYNTAX    ORName
    ID        id-mr-orname-match }
```

The rule returns *true* if, and only if, the presented value and at least one value of the attribute contains an OR-name which matches according to the following rules:

a) If the presented OR-name contains only an OR-address, the rule matches if the presented value matches an OR-address in the stored value according to the OR-address-match rule.

b) If the presented OR-name contains only a directory-name, the rule matches if the presented value matches a directory-name in the stored value according to the Distinguished Name Match rule.

c) If the presented OR-name contains both an OR-address and a directory-name, the rule matches if either of the rules described above matches.

Otherwise matching fails.

NOTE – It will not always be possible for the MS to perform a Distinguished Name Match, where the directory-name contains an attribute-value whose type is unknown to the MS.

12.4.5 OR-name-elements-match

The **OR-name-elements-match** rule determines whether a presented value is a subset of the elements present in some value of an attribute of type OR-name.

```
oRNameElementsMatch MATCHING-RULE ::= {
    SYNTAX    ORName
    ID        id-mr-orname-elements-match }
```

The rule returns *true* if, and only if, at least one value of the attribute includes those elements which comprise the presented value, and the values of the presented elements match those of the corresponding elements in the stored value.

This rule is identical to the OR-name-match rule except that the OR-address-elements-match rule is used instead of the OR-address-match rule.

NOTE – As a local matter an MS may relax the definition of the Distinguished Name Match such that only those relative-distinguished-names which appear in the presented value are required to match.

12.4.6 OR-name-substring-elements-match

The **OR-name-substring-elements-match** rule determines whether a presented value is a subset of the elements present in some value of an attribute of type OR-name, where each presented string value is a substring of the corresponding stored value.

```
oRNameSubstringElementsMatch MATCHING-RULE ::= {
    SYNTAX    ORName
    ID        id-mr-orname-substring-elements-match }
```

This rule is identical to the OR-name-elements-match rule except that for those elements which are matched using the MS-string-match rule, the MS-single-substring-match rule is applied.

NOTE – As a local matter an MS may relax the definition of the Distinguished Name Match such that only those relative-distinguished-names which appear in the presented value are required to match as substrings of the stored value.

12.4.7 OR-name-single-element-match

The **OR-name-single-element-match** rule determines whether a presented string and some OR-address element present in the OR-address component of some value of an attribute of type OR-name match for equality.

```
oRNameSingleElementMatch MATCHING-RULE ::= {
    SYNTAX    MSString {ub-msstring-match}
    ID        id-mr-orname-single-element-match }
```

The rule returns *true* if, and only if, the stored OR-name contains an OR-address component in which at least one OR-address element matches the presented value according to the MS-string-match rule. The terminal-type and extended form of network address elements are not considered when evaluating the OR-name-single-element-match rule.

12.4.8 Redirection-or-DL-expansion-match

The **Redirection-or-DL-expansion-match** rule compares for equality a presented value with the OR-address-and-optional-directory-name component of attribute-values of type Redirection-history or DL-expansion-history.

```
redirectionOrDLExpansionMatch MATCHING-RULE ::= {
    SYNTAX    ORAddressAndOptionalDirectoryName
    ID        id-mr-redirection-or-dl-expansion-match }
```

The rule returns *true* if, and only if, the presented value and at least one value of the attribute contains an OR-address-and-optional-directory-name which matches according to the OR-name-match rule.

12.4.9 Redirection-or-DL-expansion-elements-match

The **Redirection-or-DL-expansion-elements-match** rule determines whether a presented value is a subset of the elements present in the OR-address-and-optional-directory-name component of some value of an attribute of type Redirection-history or DL-expansion-history.

```
redirectionOrDLExpansionElementsMatch MATCHING-RULE ::= {
    SYNTAX    ORAddressAndOptionalDirectoryName
    ID        id-mr-redirection-or-dl-expansion-elements-match }
```

The rule returns *true* if, and only if, the presented value and at least one value of the attribute contains an OR-address-and-optional-directory-name which matches according to the OR-name-elements-match rule.

12.4.10 Redirection-or-DL-expansion-substring-elements-match

The **Redirection-or-DL-expansion-substring-elements-match** rule determines whether a presented value of element substrings is a matching subset of the elements present in the OR-address-and-optional-directory-name component of some value of an attribute of type Redirection-history or DL-expansion-history.

```
redirectionOrDLExpansionSubstringElementsMatch MATCHING-RULE ::= {
    SYNTAX    ORAddressAndOptionalDirectoryName
    ID        id-mr-redirection-or-dl-expansion-substring-elements-match }
```

This rule is identical to the Redirection-or-DL-expansion-elements-match rule except that for those elements which are matched using the MS-string-match rule, the MS-single-substring-match rule is applied.

12.4.11 Redirection-reason-match

The **Redirection-reason-match** rule compares for equality a presented value with the Redirection-reason component of attribute-values of type Redirection-history.

```
redirectionReasonMatch MATCHING-RULE ::= {
    SYNTAX    RedirectionReason
    ID        id-mr-redirection-reason-match }
```

The rule returns *true* if, and only if, the presented value matches the redirection-reason component of at least one value of the attribute according to the Integer Match rule.

12.4.12 MTS-identifier-match

The **MTS-identifier-match** rule compares for equality a presented value with attribute-values of type MTS-identifier.

```
mTSIdentifierMatch MATCHING-RULE ::= {
    SYNTAX    MTSIdentifier
    ID        id-mr-mts-identifier-match }
```

Each element of MTS-identifier (country-name, administration-domain-name, private-domain-identifier, and local-identifier) is matched using the MS-string-match rule. The rule returns *true* if, and only if, at least one value of the attribute contains elements which match those of the presented value. The equivalence rule between the X.121 and ISO 3166 forms of the country-name element may be applied (see ITU-T Rec. X.402 | ISO/IEC 10021-2).

12.4.13 Content-correlator-match

The **Content-correlator-match** rule compares for equality the presented value with attribute-values of type content-correlator.

```
contentCorrelatorMatch MATCHING-RULE ::= {
    SYNTAX    ContentCorrelator
    ID        id-mr-content-correlator-match }
```

If the presented and stored values are encoded as Octet Strings, then the rule matches according to the Octet String Match rule. If both are encoded as character string types, the rule matches according to the MS-string-case-sensitive-match rule. Otherwise matching fails.

12.4.14 Content-identifier-match

The **Content-identifier-match** rule compares for equality the presented value with attribute-values of type content-identifier.

```
contentIdentifierMatch MATCHING-RULE ::= {
    SYNTAX    ContentIdentifier
    ID        id-mr-content-identifier-match }
```

The rule returns *true* if, and only if, the presented value matches at least one value of the attribute according to the MS-string-case-sensitive-match rule.

12.5 Matching-rule support

Support for the following general-matching-rules is mandatory: Integer Match, Integer Ordering Match, Object Identifier Match, UTC Time Match, and UTC Ordering Match. An MS that claims support for a matching rule shall support its use in a Filter for any attribute-type for which it also claims support, where the attribute-type includes that matching-rule in its attribute definition (see 11.1.1).

The present match is supported for all supported attribute-types.

12.6 The Matching-rule-table information object set

The **Matching-rule-table** information object set is used as a constraining set in this Service Definition where related fields of the MATCHING-RULE information object class are referenced in the MS abstract-syntax. It comprises two object sets:

```
MatchingRuleTable MATCHING-RULE ::= {
    GeneralMatchingRules | ContentSpecificMatchingRules }
```

The **General-matching-rules** object set contains the general-attributes defined in this Service Definition. It is defined as follows:

```
GeneralMatchingRules MATCHING-RULE ::= {
    bitStringMatch | booleanMatch | contentIdentifierMatch | integerMatch | integerOrderingMatch |
    mSStringCaseSensitiveMatch | objectIdentifierMatch | oRNameMatch | uTCTimeMatch |
    uTCTimeOrderingMatch,
    ... -- 1994 extension additions --,
    contentCorrelatorMatch | mSSingleSubstringMatch | mSStringCaseSensitiveMatch |
    mSStringListElementsMatch | mSStringListMatch | mSStringMatch | mSStringOrderingMatch |
    mSSingleSubstringListElementsMatch | mSSingleSubstringListMatch | mSSubstringsMatch |
    mTSIdentifierMatch | oRAddressElementsMatch | oRAddressMatch | oRAddressSubstringElementsMatch |
    oRNameElementsMatch | oRNameMatch | oRNameSingleElementMatch | oRNameSubstringElementsMatch |
    redirectionOrDLExpansionElementsMatch | redirectionOrDLExpansionMatch |
    redirectionOrDLExpansionSubstringElementsMatch |redirectionReasonMatch | valueCountMatch }
```

The **Content-specific-matching-rules** object set is an empty, extensible set. Where an MS offers support for a given content-type, the matching-rules associated with that content-type shall be regarded as objects populating the Content-specific-matching-rules object set.

```
ContentSpecificMatchingRules MATCHING-RULE ::= { ... }
```

13 General-auto-actions

The **general-auto-actions** are applicable to entries of all content-types. Other auto-actions, specific to a content-type are defined in the relevant Specification.

Auto-actions are introduced in 6.5. The registration and deregistration of auto-actions by means of the Register-MS abstract-operation is described in 8.2.5. Alternatively, registration information may be conveyed to the MS by means of subscription. However, some auto-action-types may require that the MS supports registration by means of the Register-MS abstract-operation.

The following general-auto-action-types are defined:

a) Auto-alert;

b) Auto-modify;

c) Auto-correlate-reports;

d) Auto-delete.

NOTE – The Auto-forward auto-action defined in previous editions of this Specification is now defined as the IPM auto-forward auto-action in ITU-T Rec. X.420 | ISO/IEC 10021-7.

Each general-auto-action-type is defined as an instance of the AUTO-ACTION information object class (see 6.5.1). The **Auto-action-table** information object set is used as a constraining set in this Service Definition where related fields of the AUTO-ACTION information object class are referenced in the MS abstract-syntax. It comprises two object sets:

```
AutoActionTable AUTO-ACTION ::= {
    GeneralAutoActions | ContentSpecificAutoActions }
```

The **General-auto-actions** object set contains the general-auto-actions defined in this Service Definition. It is defined as follows:

```
GeneralAutoActions AUTO-ACTION ::= {
    auto-alert,
    ... -- 1994 extension additions --,
    auto-modify | auto-correlate-reports | auto-delete }
```

The **Content-specific-auto-actions** object set is an empty, extensible set. Where an MS offers support for a given content-type, the auto-actions associated with that content-type shall be regarded as objects populating the Content-specific-auto-actions object set.

```
ContentSpecificAutoActions AUTO-ACTION ::= { ... }
```

Each auto-action-error that shall be generated by the general-auto-action-types is defined as an instance of the AUTO-ACTION-ERROR or ABSTRACT-ERROR information object classes (the two are identical); see 6.5.3. The **Auto-action-error-table** information object set is defined below. It comprises the General-auto-action-errors and the Content-specific-auto-action-errors information object sets:

```
AutoActionErrorTable AUTO-ACTION-ERROR ::= {
    GeneralAutoActionErrors | ContentSpecificAutoActionErrors }

GeneralAutoActionErrors AUTO-ACTION-ERROR ::= {
    autoAlertError | modifyError | serviceError | securityError | messageGroupError,
    ... -- For future extension additions -- }

ContentSpecificAutoActionErrors AUTO-ACTION-ERROR ::= { ... }
```

The operation of auto-actions may be affected by the implementation of a security-policy.

Unless explicitly stated, the performance of an auto-action has no effect on the value of an entry's retrieval-status general-attribute. None of the auto-actions defined in this Service Definition affect the value of the retrieval-status general-attribute.

13.1 Auto-alert

The **Auto-alert** auto-action enables the MS-user to instruct the MS to alert the user automatically when the delivery of selected messages or reports causes the creation of an entry in the Delivery entry-class.

The alert may be conveyed to the MS-user by means of the Alert abstract-operation (see 8.2.6). Alternatively, (or in addition), the alert may be conveyed to the entity utilizing the services of the MS-user (e.g. a person) by means of some other alert mechanism, e.g. a pager, a light on a telephone, or other terminal equipment. In this case, the device which receives the alert is called the **alert-destination**, and is identified by an **alert-address**.

The **Auto-alert** auto-action-type allows one or more auto-alert-registration-parameters to be registered with the MS, each identified by its auto-alert registration-identifier. Registration information may be conveyed to the MS either by use of the Register-MS abstract-operation or by subscription. Each auto-alert-registration-parameter specifies criteria to determine whether it applies to a particular delivered-message or delivered-report. It may also specify one or more alert-destinations, and the attributes of the delivered-message or delivered-report which are to be conveyed in the Alert abstract-operation.

```
auto-alert AUTO-ACTION ::= {
    REGISTRATION PARAMETER IS AutoAlertRegistrationParameter
    ERRORS                    {auto-alert-error}
    IDENTIFIED BY             id-act-auto-alert }

AutoAlertRegistrationParameter ::= SET {
    filter                    [0] Filter OPTIONAL,
    alert-destinations        [1] SEQUENCE SIZE (1..ub-alert-addresses) OF
                                      AlertDestination OPTIONAL,
    requested-attributes      [2] EntryInformationSelection OPTIONAL,
                              -- 1994 extension --
    suppress-alert-destinations [3] BOOLEAN DEFAULT TRUE }
```

The components of the **auto-alert-registration-parameter** have the following meaning:

a) **Filter** (O): This specifies a set of criteria which a new entry representing a delivered-message or delivered-report shall satisfy before the MS will perform this registered Auto-alert auto-action. In the absence of this component, this Auto-alert auto-action will be performed for all new delivered-messages and delivered-reports.

b) **Alert-destinations** (O): This component identifies one or more alert-destinations, each identified by an alert-address, together with any information required to control the alert and qualify the information it is to convey to the alert-destination. The MS shall attempt to alert the alert-destinations subject to the qualification specified in item d).

```
AlertDestination ::= SEQUENCE {
    alert-address             EXTERNAL,
    alert-qualifier           OCTET STRING OPTIONAL }
```

The **alert-address** identifies the type of alert to be performed (e.g. the paging system to be used), and the address of an entity reachable by an alert of that type. The **alert-qualifier** specifies any additional information required to control the alert, and other information to be conveyed to the alert-destination. In the absence of this component, no control information is specified, and no additional information will be conveyed to the alert-destination (unless the required information has been registered by subscription).

Some types of alert may be internationally standardized. Others may be defined by national administrative authorities and private organizations. This implies that a number of separate authorities will be responsible for assigning identifiers for each type of alert in a way that ensures that each is unique. Each alert type is identified by an Object Identifier which is assigned when the alert type is defined. The Object Identifier is contained in the alert-address component. The ASN.1 data-type of the alert-address, and any further rules for processing the alert-qualifier are defined when the alert type is defined.

c) **Requested-attributes** (O): This indicates what information from the delivered-message or delivered-report is to be included in the argument of the Alert abstract-operation (see 8.1.4).

The Alert abstract-operation shall be invoked if this component is present or if alert-destinations is absent.

d) **Suppress-alert-destinations** (O): This component is meaningful only if both the alert-destinations and requested-attributes components are present; it is disregarded otherwise. If this component is *true*, and the Alert abstract-operation has been invoked successfully, then no attempt shall be made to convey the alert to the alert-destinations.

When a message or report is delivered, each of the registered Auto-alert auto-actions is processed in turn, in order of their registration-identifiers, until one is completed successfully. This occurs when the delivered-message or delivered-report satisfies the selection criteria of the registered Auto-alert, and the MS-user or one of the alert-destinations is alerted as described above. If the requested-attributes component is present or the alert-destinations component is absent, the Alert abstract-operation shall be invoked. This is possible only if an abstract-association already exists between the MS and MS-user. If the alert-destinations component is present (and the Alert was not performed successfully, or the suppress-alert-destinations component is *false*), the MS attempts to issue the alert to each alert-destination in turn, until one attempt succeeds. If, after processing all Auto-alert registrations, the entry has been found to satisfy the selection criteria of at least one registration, but none of the attempts to convey the alert to an alert-destination, or to invoke the Alert abstract-operation has been successful, then the MS shall set the alert-indication which is reported to the MS-user when an abstract-association is next established (see 7.1.2, item d)).

The following auto-action-error is generated if none of the alert-destinations can be alerted, or if some locally defined error condition arises:

```
auto-alert-error AUTO-ACTION-ERROR ::= {
    PARAMETER    SEQUENCE SIZE (1..ub-alert-addresses) OF AutoAlertErrorIndication
    CODE         global:id-aae-auto-alert-error }

AutoAlertErrorIndication ::= SET {
    failing-alert-destination      [0] AlertDestination OPTIONAL,
    supplementary-information      [1] GeneralString (SIZE (1..ub-supplementary-info-length)) OPTIONAL }
```

The parameter has the following meaning:

a) **Failing-alert-destination** (O): This identifies the alert-destination, if any, associated with the error condition described in supplementary-information.

b) **Supplementary-information** (O): A description of the error condition.

NOTE – As a local matter, the MS may generate an auto-action error concerning one alert-destination even if another alert-destination has been successfully alerted, e.g. where an alert-destination is registered with an erroneous address which may be detected only when an attempt is made to alert this destination.

The execution of the Auto-alert auto-action may cause the creation of an entry in the Auto-action-log entry-class.

13.2 Auto-modify

The **Auto-modify** auto-action enables the MS-user to instruct the MS to apply modifications automatically to the attributes of an entry provided that the entry satisfies given criteria. The auto-action is performed whenever an entry of the Stored-message or Message-log entry-classes is created, i.e. when a message or report is delivered, or when a draft message is stored, or when a message or probe is submitted and stored. The Auto-modify auto-action also applies to entries created as a consequence of the performance of content-specific auto-actions.

Only certain general-attribute-types are subject to modification using this auto-action (see 11.6). Attribute-types specific to a particular content-type, and subject to modification, are defined in the relevant Specification. The definition of an attribute-type which is subject to modification may specify additional rules for the performance of the Auto-modify auto-action.

The Auto-modify auto-action-type allows one or more auto-modify-registration-parameters to be registered with the MS, each identified by its registration-identifier. Each auto-modify-registration-parameter identifies the entry-class addressed by the auto-action, and specifies criteria which determine whether the auto-action shall apply to the particular entry, in which case the entry is subjected to the corresponding set of modifications. For every registered Auto-modify auto-action for which the entry satisfies the selection criteria, the corresponding set of modifications is applied.

```
auto-modify AUTO-ACTION ::= {
    REGISTRATION PARAMETER IS AutoModifyRegistrationParameter
    ERRORS                    {securityError | serviceError | modifyError |
                               messageGroupError}
    IDENTIFIED BY             id-act-auto-modify }

AutoModifyRegistrationParameter ::= SET {
    entry-class               [0] EntryClass DEFAULT delivery,
    filter                    [1] Filter OPTIONAL,
    modifications             [2] SEQUENCE SIZE (1..ub-modifications) OF EntryModification }
```

The components of the **auto-modify-registration-parameter** have the following meaning:

a) **Entry-class** (O): This specifies the entry-class addressed by the auto-action.

b) **Filter** (O): This specifies a set of criteria which a new entry shall satisfy before the MS will perform this registered Auto-modify auto-action. In the absence of this component, all entries created in the specified entry-class are subject to modification.

c) **Modifications** (M): This defines a sequence of modifications, which shall be applied in the order specified in the sequence (see 8.2.7.1).

The execution of the Auto-modify auto-action may cause the creation of an entry in the Auto-action-log entry-class.

Support of the Auto-modify auto-action by an MS, or UA accessing an MS, requires that it supports registration of the auto-modify-registration-parameter by means of the Register-MS abstract-operation. Registration information may also be conveyed to the MS by subscription.

13.3 Auto-correlate-reports

The **Auto-correlate-reports** auto-action enables the MS-user to instruct the MS to correlate, automatically, delivered-report entries of the Delivery entry-class with the submitted messages and probes to which they are related. The auto-action also records each successful invocation of the MS-cancel-deferred-delivery abstract-operation. The Auto-correlate-reports auto-action-type is available only by subscription, not by registration using the Register-MS abstract-operation.

```
auto-correlate-reports AUTO-ACTION ::= {
    IDENTIFIED BY id-act-auto-correlate-reports }
```

The MS supports the Auto-correlate-reports auto-action by means of the AC-correlated-report-list, AC-report-summary, AC-uncorrelated-report-list, and deferred-delivery-cancellation-time general-attributes (see 11.2.1, 11.2.2, 11.2.3, and 11.2.20).

No entry shall be created in the Auto-action-log entry-class as the result of performing the Auto-correlate-reports auto-action.

13.4 Auto-delete

The **Auto-delete** auto-action enables the MS-user to instruct the MS to delete an entry (and any child-entries associated with it) automatically at a predetermined interval after its creation. The MS shall subject to Auto-deletion every entry in the Stored-message entry-class which possesses a storage-time general-attribute whose value is less than the current date and time. Auto-deletion shall not be performed while an abstract-association exists between the MS and MS-user. All restrictions specified for the Delete abstract-operation apply equally to the Auto-delete auto-action. Consequently, Auto-delete shall not apply to child-entries nor to entries of the Delivery entry-class which have a retrieval-status value of *new* (see 8.2.4).

```
auto-delete AUTO-ACTION ::= {
    ERRORS          {securityError}
    IDENTIFIED BY   id-act-auto-delete }
```

No registration-parameter is defined for the Auto-delete auto-action.

NOTES

1 This auto-action differs from the others defined in this Service Definition in that its activity is initiated by the expiry of a timer rather than the creation of an entry. The period within which deletion shall occur after the storage-time is reached is not prescribed. In practice, implementations may choose to perform this action at fixed intervals, e.g. hourly, daily, or weekly.

2 A newly-created entry is subject to Auto-delete processing only after all other registered auto-actions have been performed.

3 An entry which contains the storage-period general-attribute also contains the storage-time general-attribute. Both attributes indicate, in different ways, the date and time after which the message is to be deleted, i.e. storage-period in combination with creation-time is equivalent to storage-time (see 11.2.71).

4 The marked-for-deletion attribute-type is not considered when performing the Auto-delete auto-action.

Support of the Auto-delete auto-action by an MS, or UA accessing an MS, requires that it supports registration of the auto-action with the MS by means of the Register-MS abstract-operation. The Auto-delete auto-action is also available by subscription.

The execution of the Auto-delete auto-action may cause the creation of an entry in the Auto-action-log entry-class.

SECTION 4 – PROCEDURES FOR MESSAGE STORE AND PORT REALIZATION

14 Overview

This section describes the procedures for the MS and the port realization. It contains a description of the consumption of the MTS abstract-service in clause 15. The provision of the MS abstract-service is described in clause 16. The port realization in the form of application service elements is described in clause 16.

The performance of the abstract-operations described in clauses 15 and 16 may be subject to the requirements of a security-policy, (if one is in force), which applies to the MTS abstract-services and to the MS abstract-services.

15 Consumption of the Message Transfer abstract-service

This clause specifies how the MS consumes the MTS abstract-service which is defined in clause 8 of ITU-T Rec. X.411 | ISO/IEC 10021-4. The consumption by the MS of the services of the MTS Delivery, Submission, and Administration Ports is defined.

15.1 Consumption of the Delivery Port abstract-services

This clause describes the performance of the Message-delivery and Report-delivery abstract-operations, and the invocation of the Delivery-control abstract-operation. The MS consumption of the Delivery Port abstract-services assumes that an abstract-association exists between the Delivery Port supplier (the MTS) and the Delivery Port consumer (the MS). The performance of the abstract-operations is in sequential order; no parallel processing takes place. Error cases are not described.

15.1.1 Performance of the Message-delivery abstract-operation

When the MTS invokes the Message-delivery abstract-operation, the MS performs the following actions:

a) The MS creates an entry (and conditionally one or more child-entries) with entry-type delivered-message in the Delivery entry-class and sets its retrieval-status to *new*. A corresponding entry with the same sequence-number is created in the Delivery-log entry-class. The mandatory and optional attribute-types for delivered-message entries of the Delivery and Delivery-log entry-classes are indicated in Table 2.

b) The MS returns a Message-delivery result to the MTS to indicate that the delivery was successful. The Message-delivery result shall contain proof-of-delivery information if the delivered-message contains a proof-of-delivery-request argument. This proof-of-delivery may be computed using the subject-MS secret key (see 8.5.7 and 8.3.1.1.2.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4).

c) The MS determines if any auto-actions are to be performed. In addition to the general-auto-action-types defined in this Service Definition, content-specific auto-action-types may be defined for individual content-types in the relevant Specifications. Except for content-specific auto-actions whose order of execution is specified in the relevant Specification, auto-actions are executed in the order indicated below:

   1) If Auto-modify auto-actions have been registered, the new entry is matched against the entry-class and any selection criteria specified in each Auto-modify registration-parameter. If the entry satisfies these criteria, then the sequence of entry-modifications is applied. This is repeated for each registered Auto-modify auto-action.

   2) If Auto-alert auto-actions have been registered, the new entry is matched against the selection criteria specified in each Auto-alert registration-parameter. If the entry satisfies the selection criteria of any of these, the MS shall attempt to invoke the Alert abstract-operation or issue the alert to an alert-destination (or perform both of these actions) as defined in 13.1.

      NOTE – If the delivered-message is deleted as a result of a content-specific auto-action, the Auto-alert is not performed.

   Certain auto-actions, when executed, cause the creation of an entry in the Auto-action-log entry-class. If one of these auto-actions causes an auto-action error, the MS shall attach an auto-action-error attribute indicating the nature of the error to the Auto-action-log entry, and shall set the auto-action-error-indication, which is reported to the MS-user when an abstract-association is next established. The MS then resumes processing of registered auto-actions.

d) After the actions described above have been performed, the new entry is made accessible to the MS-user over the Retrieval Port.

15.1.2 Performance of the Report-delivery abstract-operation

When the MTS invokes the Report-delivery abstract-operation, the MS performs the following actions:

a) The MS creates an entry with entry-type delivered-report in the Delivery entry-class and sets its retrieval-status to *new*. A corresponding entry with the same sequence-number is created in the Delivery-log entry-class. The mandatory and optional attribute-types for delivered-report entries of the Delivery and Delivery-log entry-classes are indicated in Table 2.

b) If the delivered-report contains the originally submitted message in its returned-content field, then the MS shall create a child-entry containing the returned-content-entry in the Delivery entry-class; no entry is created in the Delivery-log entry-class.

c) The MS returns a Report-delivery result to the MTS to indicate that the delivery was successful. The Report-delivery result has no parameters. For details, see 8.3.1.2.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

d) The MS determines if any auto-actions are to be performed. Auto-actions are executed in the order indicated below:

   1) If the Auto-correlate-reports auto-action is in effect then the MS attempts to correlate the delivered-report with the entry in the Submission or Submission-log entry-class which is the subject of the report. If an entry corresponding to the report's subject-message is found in the Submission or Submission-log entry-class, then the MS shall update the auto-correlation attributes indicated in 13.3.

   2) If Auto-modify auto-actions have been registered, the new entry is matched against the entry-class and any selection criteria specified in each Auto-modify registration-parameter. If the entry satisfies these criteria, then the sequence of entry-modifications is applied. This is repeated for each registered Auto-modify auto-action.

   3) If Auto-alert auto-actions have been registered, the new entry is matched against the selection criteria specified in each Auto-alert registration-parameter. If the entry satisfies the selection criteria of any of these, the MS shall attempt to invoke the Alert abstract-operation or issue the alert to an alert-destination (or perform both of these actions) as defined in 13.1.

   Certain auto-actions, when executed, cause the creation of an entry in the Auto-action-log entry-class. If one of these auto-actions causes an auto-action error, the MS shall attach an auto-action-error attribute indicating the nature of the error to the Auto-action-log entry, and shall set the auto-action-error-indication, which is reported to the MS-user when an abstract-association is next established. The MS then resumes processing of registered auto-actions.

e) After the actions described above have been performed, the new entry is made accessible to the MS-user over the Retrieval Port.

15.1.3 Invocation of the Delivery-control abstract-operation

If the MS requires a temporary restriction on the delivery abstract-operations that the MTS may invoke, or requires an alteration to the existing restrictions, it performs the following actions:

a) The MS invokes the Delivery-control abstract-operation, containing the parameters to be changed. See 8.3.1.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

b) The MTS returns a result to signify that the specified controls are now in force; alternatively, an error is returned. The result indicates any abstract-operations that the MTS would invoke, or any message types that it would deliver, were it not for the prevailing controls. See 8.3.1.3.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

c) When the MS is able to accept messages or reports held for delivery, it may invoke the Delivery-control abstract-operation to relax the restrictions. The effects of a Delivery-control abstract-operation are cancelled when either a new Delivery-control abstract-operation alters the restrictions or when the abstract-association between the MS and MTS is released.

15.1.4 Generation rules for general-attributes

Attributes shall be generated when a message or report is delivered, and may be generated when an auto-action is performed.

The entries of the Delivery and Delivery-log entry-classes created as a result of Message-delivery and Report-delivery shall contain the mandatory attributes indicated in Table 2, and such optional attributes as are supported by the MS implementation and subscribed to by the MS-user. The generated attributes form a new entry, or in some cases a parent-entry and one or more child-entries (see 6.3.4). Each attribute of an entry in the Delivery-log entry-class is identical with the attribute of the same name in the corresponding entry of the Delivery entry-class (except for the deletion-time attribute which may be present only in Delivery-log entries). The following kinds of general-attributes shall be generated:

a) general-attributes generated by the MS itself (e.g. sequence-number);

b) general-attributes generated from components of the message-delivery-envelope and report-delivery-envelope. For components which are not present, but for which default values are defined, a general-attribute containing the default value is generated.

See Table 4 and 11.4 for the rules on the generation of the general-attributes. The generation rules for content-specific attributes are defined in the relevant Specifications, e.g. the IPMS-specific attributes are described in 19.7 of ITU-T Rec. X.420 | ISO/IEC 10021-7.

15.2 Consumption of the Submission Port abstract-services

This subclause describes the invocation of the Message-submission, Probe-submission, and Cancel-deferred-delivery abstract-operations, and the consumption of the Submission-control abstract-operation. The MS consumption of the Submission Port abstract-services assumes that an abstract-association exists between the Submission Port supplier (the MTS) and the Submission Port consumer (the MS). The performance of the abstract-operations is in sequential order, no parallel processing takes place. Error cases are not described.

15.2.1 Invocation of the Message-submission abstract-operation

The invocation of the Message-submission abstract-association results from the performance of an auto-action by the MS or from the invocation of the MS-message-submission abstract-operation by the MS-user. The definition of the performance of MS-message-submission includes a description of the invocation of the Message-submission abstract-operation (see 16.2.1).

If the Message-submission is the consequence of an auto-action execution, the following additional actions are taken:

a) The submission-options parameter of the MS-message-submission abstract-operation is obtained as follows: if the auto-action's registration-parameter contains a submission-options parameter, that value is used directly. Otherwise, the general (non UA-specific) submission-defaults previously registered by means of the MS-register abstract-operation are used.

b) For auto-actions which cause the creation of an entry in the Auto-action-log entry-class, an error in the MS-message-submission abstract-operation shall cause the MS to attach an auto-action-error attribute indicating the nature of the error to the Auto-action-log entry. In addition, the MS shall present an auto-action-error-indication to the MS-user in MS-bind-result when an abstract-association is next established.

15.2.2 Invocation of the Probe-submission abstract-operation

The invocation of the Probe-submission abstract-operation results from the invocation of the MS-probe-submission abstract-operation by the MS-user. The definition of the performance of MS-probe-submission includes a description of the invocation of the Probe-submission abstract-operation (see 16.2.2).

15.2.3 Invocation of the Cancel-deferred-delivery abstract-operation

When the MS-user invokes the Cancel-deferred-delivery abstract-operation, the MS performs the following actions:

a) The MS invokes the Cancel-deferred-delivery abstract-operation over its abstract-association with the MTS, using the Cancel-deferred-delivery argument supplied by the MS-user. See 8.2.1.3.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

b) If the MTS returns a result (indicating success) and the Auto-correlate-reports auto-action is subscribed to, then the MS searches the Submission and Submission-log entry-classes for an entry corresponding to the submitted message for which deferred-delivery has been cancelled. If such an entry is found, then the MS attaches a deferred-delivery-cancellation-time attribute to it to record the date and time at which the cancellation occurred. The AC-report-summary attribute is also updated.

c) If the abstract-operation is performed successfully, the result is returned to the MS-user in the form of a Cancel-deferred-delivery result issued by the MS. Otherwise the error is returned to the MS-user.

15.2.4 Performance of the Submission-control abstract-operation

When the MTS invokes the Submission-control abstract-operation, the MS performs the following actions:

a) If no abstract-association exists between the MS and MS-user, a remote-bind-error is returned to the MTS and the procedure terminates.

b) The MS invokes a Submission-control abstract-operation over its abstract-association with the MS-user using the Submission-control argument supplied by the MTS.

c) The MS-user may return a result to signify that the specified controls are now in force; alternatively an error is returned. The result indicates any abstract-operations that the MS-user would invoke, or any message types that it would submit, were it not for the prevailing controls. See 8.2.1.4.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

d) The result or error returned by the MS-user is returned to the MTS.

e) When the MTS is able to accept the submission of messages or probes, it may invoke the Submission-control abstract-operation to relax the restrictions. The effects of a Submission-control abstract-operation are cancelled when either a new Submission-control abstract-operation alters the restrictions or when the abstract-association between the MS and MTS is released.

15.3 Consumption of the Administration Port abstract-services

This subclause describes the performance of the Register and Change-credentials abstract-operations. The consumption of the Administration Port abstract-services assumes that an abstract-association exists between the Administration Port supplier (the MTS) and the Administration Port consumer (the MS). The performance of the abstract-operations is in sequential order, no parallel processing takes place. Error cases are not described.

The MS use of the Administration Port is subject to the security-policy in force.

15.3.1 Invocation of the Register abstract-operation

When the MS-user invokes the Register abstract-operation, the MS performs the following actions:

a) The MS invokes the Register abstract-operation over its abstract-association with the MTS, using the Register argument supplied by the MS-user. See 8.4.1.1.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

b) The result or error returned by the MTS is returned to the MS-user.

NOTE – Some security-policies may permit the user-security-labels to be changed only if a secure link is employed.

15.3.2 Invocation of the Change-credentials abstract-operation

When the MS-user invokes the Change-credentials abstract-operation, the MS performs the following actions:

a) The MS invokes the Change-credentials abstract-operation over its abstract-association with the MTS, using the Change-credentials argument supplied by the MS-user. See 8.4.1.2.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

b) The MTS generates a Change-credentials result or error, which the MS returns to the MS-user. If the abstract-operation is successful, the MS stores the new credentials.

15.3.3 Performance of the Change-credentials abstract-operation

When the MTS invokes the Change-credentials abstract-operation, the MS performs the following actions:

a) The MS verifies that the supplied arguments are valid for the Change-credentials abstract-operation. See 8.4.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4. If the old credentials are incorrect or the new credentials are not acceptable, an error is returned and the procedure terminates.

b) The MS stores the new credentials for use on subsequent occasions when it binds to the MTS.

16 Supply of the Message Store abstract-service

This clause specifies how the MS supplies the MS abstract-service. This describes the supply of the abstract-services available at the Retrieval, MS-submission, and Administration Ports.

16.1 Supply of the Retrieval Port abstract-services

This subclause describes the supply of the Summarize, List, Fetch, Delete, Register-MS, Modify, and Alert abstract-operations. The MS abstract-service supply of the Retrieval Port abstract-services assumes that an abstract-association exists between the Retrieval Port supplier (the MS) and the Retrieval Port consumer (the MS-user). The performance of the abstract-operations is asynchronous, subject to the conditions stated in 8.2. Not all error cases are described.

16.1.1 Performance of the Summarize abstract-operation

When the MS-user invokes the Summarize abstract-operation, the MS performs the following actions:

a) The MS verifies that the supplied arguments are valid for the Summarize abstract-operation. Any attribute-types specified in summary-requests must be available for use with Summarize and subscribed to by the MS-user. See 8.2.1.1 and Table 2. If an error is found, the procedure terminates and the error is returned.

b) The MS establishes which entry-class is addressed by the abstract-operation and identifies the entry or entries specified in the argument. If no entries are selected, this is reported to the MS-user in the Summarize result and the procedure terminates.

c) If any entries are selected, the MS accumulates counts in accordance with the supplied summary-requests argument.

e) The MS returns the Summarize result to the MS-user (see 8.2.1.2).

f) If a security-policy is in force, then to ensure that such a security-policy is not violated during the Summarize abstract-operation, the message-security-label is checked against the security-context by the MS. If the requested operation is barred by the security-policy the Summarize abstract-operation shall be abandoned and a security error shall be indicated.

16.1.2 Performance of the List abstract-operation

When the MS-user invokes the List abstract-operation, the MS performs the following actions:

a) The MS verifies that the supplied arguments are valid for the List abstract-operation. Any attribute-types specified in requested-attributes must be available for use with List and subscribed to by the MS-user. See 8.2.2.1 and Table 2. If an error is found, the procedure terminates and the error is returned.

b) The MS establishes which entry-class is addressed by the abstract-operation and identifies the entry or entries specified in the argument. If no entries are selected, the MS returns a List result to the MS-user and the procedure terminates.

c) If a requested-attributes parameter is specified in the List argument, then for each selected entry these attributes (if present) are returned. If no requested-attributes are specified in the List argument, the MS determines whether the present abstract-association identified, in its MS-bind-argument, a UA-registration which specifies UA-list-attribute-defaults. If so, these attributes (if present) are returned for each selected entry. Otherwise, the MS returns the attributes (if present) specified in the general (non UA-specific) list-attribute-defaults if the MS-user has previously registered these using the Register-MS abstract-operation (see 8.2.5.1, item c). The retrieval-status of each Stored-message entry for which entry-information is returned is set to *listed* if its current value is *new*. Additional rules affecting the setting of retrieval-status may be defined in the Specifications for particular content-types.

d) The MS returns a List result to the MS-user (see 8.2.2.2).

e) If a security-policy is in force, then to ensure that such a security-policy is not violated during the List abstract-operation, the message-security-label is checked against the security-context by the MS. If the requested operation is barred either by the security-policy or by temporary security restrictions, the List abstract-operation shall be abandoned and a security error shall be indicated.

16.1.3 Performance of the Fetch abstract-operation

When the MS-user invokes the Fetch abstract-operation, the MS performs the following actions:

a) The MS verifies that the supplied arguments are valid for the Fetch abstract-operation. Any attribute-types specified in requested-attributes must be subscribed to by the MS-user. See 8.2.3.1 and Table 2. If an error is found, the procedure terminates and the error is returned.

b) The MS establishes which entry-class is addressed by the abstract-operation and identifies the entry or entries specified in the argument. If no entries are selected, the MS returns a Fetch result to the MS-user and the procedure terminates.

c) The fetch-restrictions on allowed-content-types established by the MS-bind abstract-operation (unless overridden) are applied to determine whether requested-attributes of the selected entry shall be returned or whether an error results (see 7.1.1, item d)).

d) If a requested-attributes parameter is specified in the Fetch argument, then these attributes are returned, if present, for the first selected entry. If no requested-attributes are specified in the Fetch argument, the MS determines whether the present abstract-association identified, in its MS-bind-argument, a UA-registration which specifies UA-fetch-attribute-defaults. If so, these attributes (if present) are returned. Otherwise, the MS returns the attributes (if present) specified in the general (non UA-specific) fetch-attribute-defaults if the MS-user has previously registered these using the Register-MS abstract-operation (see 8.2.5.1, item d)). The fetch-restrictions on allowed-EITs and maximum-attribute-length established by the MS-bind-argument may limit the information returned (see 7.1.1, item d)).

   If the first selected entry belongs to the Stored-message entry-class and has a retrieval-status of *new*, its retrieval-status is set to *listed*. Additional rules affecting the setting of retrieval-status may be defined in the Specification relevant to a given content-types. If several entries match the search criteria, the sequence-numbers of the second and subsequent entries are returned in the order specified by selector (see 8.1.3). If the number of matching entries exceeds the specified limit, the next sequence-number beyond the limit is also returned (see 8.2.3.2).

e) The MS returns the Fetch result to the MS-user.

f) If a security-policy is in force, then to ensure that such a security-policy is not violated during the Fetch abstract-operation, the message-security-label is checked against the security-context by the MS. If the requested operation is barred either by the security-policy or by temporary security restrictions, the Fetch abstract-operation shall be abandoned and a security error shall be indicated.

16.1.4 Performance of the Delete abstract-operation

When the MS-user invokes the Delete abstract-operation, the MS performs the following actions:

a) The MS verifies that the supplied arguments are valid for the Delete abstract-operation (see 8.2.4.1). If an error is found, the procedure terminates and the error is returned.

b) The MS establishes which entry-class is addressed by the abstract-operation and identifies the entry or entries specified in the argument. If no entries are selected, the MS returns a Delete result to the MS-user and the procedure terminates.

c) If an entry is encountered whose deletion is prohibited by a delete restriction (see 8.2.4), then an error is generated which includes an indication of the sequence-numbers of the entries successfully deleted (if any), and the procedure terminates. As a local matter, the MS may verify that no delete restrictions apply to any of the selected entries before attempting their deletion. If an entry in the Stored-message entry-class is deleted, then the corresponding entry in the Message-log entry-class has its deletion-time attribute set to the time at which deletion occurred.

d) The MS returns a Delete result to the MS-user (see 8.2.4.2). If the Delete argument contains a selector parameter to specify the entries to be deleted, then the sequence-numbers of those entries are reported in the Delete result.

16.1.5 Performance of the Register-MS abstract-operation

When the MS-user invokes the Register-MS abstract-operation, the MS performs the following actions:

a) The MS verifies that the supplied arguments are valid for the Register-MS abstract-operation (see 8.2.5.1). If an error is found, the procedure terminates and the error is returned.

b) The supplied arguments may cause registrations to be added, deleted, or changed. In general, the registration of an auto-action has no effect on existing entries; only entries created subsequent to the auto-action registration are subject to auto-action processing. However, auto-actions whose performance is not initiated by the creation of an entry (such as Auto-delete, which is initiated by the expiry of a timer) shall be effective for existing entries.

c) If the supplied arguments include a registration-status-request, then the requested information is returned to the MS-user. Otherwise, a Null result is returned.

d) If a security-policy is in force then the Register-MS abstract-operation shall be subject to such a policy. Some security-policies may only permit user-security-labels to be changed if a secure link is employed. Other local means of changing the user-security-labels in a secure manner may be provided.

16.1.6 Performance of the Modify abstract-operation

When the MS-user invokes the Modify abstract-operation, the MS performs the following actions:

a) The MS verifies that the supplied arguments are valid for a Modify abstract-operation. The MS attempts to discover all possible static errors before applying the modifications requested. For details see 8.2.7.1. If an error is found, the procedure terminates and the error is returned.

b) The MS establishes which entry-class is addressed by the abstract-operation and identifies the entry or set of entries specified in the argument.

c) The MS applies the sequence of modifications in the order specified to each selected entry in turn. If an error is encountered then the procedure terminates leaving the entry which caused the error unchanged. The modify-error indicates the sequence-numbers of those entries which were successfully modified, as well as that of the failing entry.

d) The result is returned to the MS-user; it contains a list of the sequence-numbers of the entries selected for modification (if any).

16.1.7 Invocation of the Alert abstract-operation

When the MTS invokes the Message-delivery or Report-delivery abstract-operations, the MS performs the following actions:

a) Each registered Auto-alert auto-action is processed until one succeeds in issuing the alert or all have been attempted. If the delivered-message or delivered-report satisfies the registered selection criteria, and the requested-attributes parameter is present (or the alert-destinations component is absent), then the MS invokes the Alert abstract-operation. (Note that this procedure, in common with the others defined in 16.1, assumes that an abstract-association exists between the MS and MS-user; if no abstract-association exists, Alert cannot be invoked.) If the Alert is performed successfully, the MS-user returns an alert-result.

b) If the alert-destinations component is present (and the Alert was not performed successfully, or the suppress-alert-destinations component is *false*), the MS attempts to issue the alert to each alert-destination in turn, until one attempt succeeds, as defined in 13.1.

c) If, after processing all Auto-alert registrations, the Alert abstract-operation has not been performed successfully, and no alert-destinations have been alerted, then the MS sets the alert-indication which is reported to the MS-user when an abstract-association is next established.

d) If a security-policy is in force, then to ensure that such a security-policy is not violated during the alert the message-security-label is checked against the security-context by the MS. If the Alert abstract-operation is barred either by the security-policy or by temporary security restrictions, the action taken shall be defined by the security policy in force.

16.2 Supply of the MS-submission Port abstract-services

This subclause describes the performance of the MS-message-submission, MS-probe-submission, and MS-cancel-deferred-delivery abstract-operations, and the invocation of the Submission-control abstract-operation. The MS abstract-service supply of the MS-submission Port abstract-services assumes that an abstract-association exists between the MS-submission Port supplier (the MS) and the MS-submission Port consumer (the MS-user). The performance of the abstract-operations is in sequential order, no parallel processing takes place. Not all error cases are described.

16.2.1 Performance of the MS-message-submission abstract-operation

When the MS-user invokes the MS-message-submission abstract-operation, the MS performs the following actions:

a) If a 1994 Application Context is in use, the submission-options parameter of the MS-message-submission-argument is extracted and the MTS Message-submission argument is constructed from the remainder. If the submission-options parameter is absent and the present abstract-association identified a UA-registration in its MS-bind-argument, the submission-options are drawn from the UA-submission-defaults of that UA-registration. If the present abstract-association did not identify a UA-registration, or the UA-registration did not contain UA-submission-defaults, the submission-options are drawn from the general (non UA-specific) submission-defaults registered by Register-MS. If the submission-options contain any MS-submission-extensions these are acted on according to the rules stated in the Specification which defines the content-type of the submitted message. If the MS is unable to progress the submission because the submission-options are incorrectly specified or include MS-submission-extensions which are not supported, then the MS shall create an entry in the Submission-log entry-class and attach an MS-submission-error to it; in this case, the procedure resumes at step g). Otherwise, the procedure resumes at step c).

   If a 1988 Application Context is in use, but the MS-user has registered general (non UA-specific) submission-defaults by means of Register-MS when using a 1994 Application Context, then the submission-options parameter of the MS-message-submission-argument is drawn from that registration.

b) If a 1988 Application Context is in use, and the MS-message-submission argument contains a forwarding-request parameter. the MS verifies that the entry to be forwarded is a delivered-message entry of the Delivery entry-class and incorporates the entry in the MTS Message-submission argument according to content-type specific rules.

c) If the submission-options contained a request for the creation of a draft-message entry (without submission to the MTS) then this entry is created and the procedure continues at step g) below. The mandatory and optional attribute-types for entries of the Draft entry-class are indicated in Table 2.

d) The MS attempts to establish an abstract-association with the MTS if one does not already exist. If an abstract-association cannot be established, a remote-bind-error is returned to the MS-user and the procedure terminates.

e) The MS creates an entry in the Submission-log entry-class and invokes the Message-submission abstract-operation over its abstract-association with the MTS. The mandatory and optional attribute-types for submitted-message entries of the Submission-log entry-class are indicated in Table 2. If Message-submission is unsuccessful the MS attaches an MS-submission-error attribute to the Submission-log entry to record the error and the procedure continues at step g).

f) If the Message-submission is successful and the submission-options parameter (or its registered default) requested the creation of an entry in the Submission entry-class then that entry is created. The mandatory and optional attribute-types for submitted-message entries of the Submission entry-class are indicated in Table 2. If the Auto-correlate-reports auto-action is subscribed to by the MS-user then the MS generates the correlation attributes indicated in 13.3.

g) The entry created in the Submission-log or Draft entry-class, or the entries created in both the Submission-log and Submission entry-classes are modified as follows. If the submission-options parameter (or its registered default) specified one or more message-group-names to be added to the new entry or entries, then these attribute-values are added. If Auto-modify auto-actions have been registered by the MS-user, and enabled in the submission-options parameter (or its registered default), then each is examined in turn. If the entry belongs to the entry-class specified for that registered auto-action, and satisfies any selection criteria specified, then the sequence of modifications is applied. If any content-specific auto-actions have been registered they are performed.

   Certain auto-actions, when executed, cause the creation of an entry in the Auto-action-log entry-class. If any of these auto-actions causes an auto-action error, the MS shall attach an auto-action-error attribute indicating the nature of the error to the Auto-action-log entry, and shall set the auto-action-error-indication, which is returned to the MS-user in the MS-message-submission-result. The MS then resumes processing of registered auto-actions.

   If the Message-submission is unsuccessful, or failed in step a) above, the error is reported to the MS-user and the procedure continues at step i).

h) The MS-message-submission-result is returned to the MS-user. This shall indicate the sequence number of any main-entry created in the Draft, Submission or Submission-log entry-class. If Message-submission

was requested (rather than storage of a draft-message), then the MS-message-submission-result shall include the MTS Message-submission result (for details see 8.2.1.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4).

i) The MS may choose to terminate the abstract-association with the MTS either when the MS-user terminates its abstract-association with the MS, or after a certain period of inactivity.

j) If a security-policy is in force, then to ensure that such a security-policy is not violated during MS-message-submission, the message-security-label is checked against the security-context by the MS. If the MS-message-submission is barred either by the security-policy or by temporary security restrictions, a security-error shall be indicated.

16.2.2 Performance of the MS-probe-submission abstract-operation

When the MS-user invokes the MS-probe-submission abstract-operation the MS performs the following actions:

a) If a 1994 Application Context is in use, the submission-options parameter of the MS-probe-submission-argument is extracted and the MTS Probe-submission argument is constructed from the remainder. If the submission-options parameter is absent and the present abstract-association identified a UA-registration in its MS-bind-argument, the submission-options are drawn from the UA-submission-defaults of that UA registration. If the present abstract-association did not identify a UA-registration, or the UA-registration did not contain UA-submission-defaults, the submission-options are drawn from the general (non UA-specific) submission-defaults registered by Register-MS. If the MS is unable to progress the submission because the submission-options are incorrectly specified, then the MS shall create an entry in the Submission-log entry-class and attach an MS-submission-error to it; in this case the procedure resumes at step e).

If a 1988 Application Context is in use, but the MS-user has registered general (non UA-specific) submission-defaults by means of Register-MS when using a 1994 Application Context, then the submission-options parameter of the MS-probe-submission-argument is drawn from that registration.

b) The MS attempts to establish an abstract-association with the MTS if one does not already exist. If an abstract-association cannot be established, a remote-bind-error is returned to the MS-user and the procedure terminates.

c) The MS creates an entry in the Submission-log entry-class and invokes the Probe-submission abstract-operation over its abstract-association with the MTS. The mandatory and optional attribute-types for submitted-probe entries of the Submission-log entry-class are indicated in Table 2. If Probe-submission is unsuccessful the MS attaches an MS-submission-error attribute to the Submission-log entry to record the error and the procedure continues at step e).

d) If the Probe-submission is successful and the submission-options parameter (or its registered default) requested the creation of an entry in the Submission entry-class then that entry is created. The mandatory and optional attribute-types for submitted-probe entries of the Submission entry-class are indicated in Table 2. If the Auto-correlate-reports auto-action is subscribed to by the MS-user then the MS generates the correlation attributes indicated in 13.3.

e) The entry created in the Submission-log entry-class, or the entries created in both the Submission-log and Submission entry-classes are modified as follows. If the submission-options parameter (or its registered default) specified one or more message-group-names to be added to the new entry or entries, then these attribute-values are added. If Auto-modify auto-actions have been registered by the MS-user, and enabled in the submission-options parameter (or its registered default), then each is examined in turn. If the entry belongs to the entry-class specified for that registered auto-action, and satisfies any selection criteria specified, then the sequence of modifications is applied.

Certain auto-actions, when executed, cause the creation of an entry in the Auto-action-log entry-class. If any of these auto-actions causes an auto-action error, the MS shall attach an auto-action-error attribute indicating the nature of the error to the Auto-action-log entry, and shall set the auto-action-error-indication, which is returned to the MS-user in the MS-probe-submission-result. The MS then resumes processing of registered auto-actions.

If the Probe-submission was unsuccessful, or failed in step a) above, the error is reported to the MS-user and the procedure continues at step g).

f) The MS-probe-submission-result is returned to the MS-user. This shall indicate the sequence number of any entries created in the Submission and Submission-log entry-classes. The MS-probe-submission-result shall include the MTS Probe-submission result (for details see 8.2.1.2.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4).

g) The MS may choose to terminate the abstract-association with the MTS either when the MS-user terminates its abstract-association with the MS, or after a certain period of inactivity.

h) If a security-policy is in force, then to ensure that such a security-policy is not violated during MS-probe-submission, the message-security-label is checked against the security-context by the MS. If the MS-probe-submission is barred either by the security-policy or by temporary security restrictions, a security-error shall be indicated.

16.2.3 Performance of the MS-cancel-deferred-delivery abstract-operation

When the MS-user invokes the MS-cancel-deferred-delivery abstract-operation, the MS performs the following actions:

a) The MS verifies that the supplied arguments are valid for an MS-cancel-deferred-delivery abstract-operation. For details see 8.2.1.3.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

b) The MS attempts to establish an abstract-association with the MTS if one does not already exist. If an abstract-association cannot be established, a remote-bind-error is returned to the MS-user and the procedure terminates.

c) The MS invokes a Cancel-deferred-delivery abstract-operation to the MTS. If the request for delivery cancellation is successful and the user has subscribed to the Auto-correlation of Reports element of service, the MS searches for an entry in the Submission and Submission-log entry-classes corresponding to the submitted-message for which deferred-delivery has been cancelled. If this entry is present, the MS attaches a deferred-delivery-cancellation-time attribute to it to record the date and time at which delivery cancellation occurred, and updates the AC-report-summary attribute to record the cancellation.

d) The MS reports the outcome of the abstract-operation to the MS-user.

e) The MS may choose to terminate the abstract-association with the MTS either when the MS-user terminates its abstract-association with the MS or after a certain period of inactivity.

16.2.4 Invocation of the Submission-control abstract-operation

If the MTS invokes the Submission-control abstract-operation, or if for some internal reason the MS requires a temporary restriction on the submission abstract-operations that the MS-user may invoke, or requires an alteration to the existing restrictions, the MS performs the following actions:

a) The MS invokes the MS-submission-control abstract-operation (see 8.3.4).

b) The MS waits for the MS-user to generate an MS-submission-control result, confirming the acceptance of the MS-submission-control abstract-operation.

c) If the Submission-control was originally invoked by the MTS, the MS returns the MS-submission-control result received from the MS-user to the MTS.

16.2.5 Generation rules for general-attributes

The entries of the Submission, Submission-log, and Draft entry-classes created as a result of MS-message-submission and MS-probe-submission shall contain the mandatory attributes indicated in Table 2, and such optional attributes as are supported by the MS implementation and subscribed to by the MS-user. The generated attributes form a new entry, or in some cases a parent-entry and one or more child-entries (see 6.3.4). Each attribute of an entry in the Submission-log entry-class has the same value as the equivalent attribute in the corresponding entry of the Submission entry-class (with the exception of the deletion-time and MS-submission-error attributes which are specific to Submission-log entries). The following kinds of general-attributes shall be generated:

a) General-attributes generated by the MS itself (e.g. sequence number).

b) General-attributes generated from components of the message-submission-envelope and probe submission envelope, and from the result of the MTS Message-submission and Probe-submission abstract-operations. For components which are not present, but for which default values are defined, a general-attribute containing the default value is generated.

See Table 4 and 11.4 for the rules on how the general-attributes are generated. The generation rules for content-specific attributes are described in the relevant Specification for the content-type concerned (e.g. the IPMS-specific attributes are described in clause 19 of ITU-T Rec. X.420 | ISO/IEC 10021-7).

16.3 Supply of the Administration Port abstract-services

This subclause describes the performance of the Register and Change-credentials abstract-operations. The MS abstract-service supply of the Administration Port abstract-services assumes that an abstract-association exists between the Administration Port supplier (the MS) and the Administration Port consumer (the MS-user). The performance of the abstract-operations is in sequential order, no parallel processing takes place. Not all error cases are described.

16.3.1 Performance of the Register abstract-operation

When the MS-user invokes the Register abstract-operation, the MS performs the following actions:

a) The MS verifies that the supplied arguments are valid for the Register abstract-operation. See 8.4.1.1.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4. Subject to local policy or subscription, the MS may impose additional restrictions on the registrations which may be performed by the MS-user; if these restrictions are violated, an abstract-error is returned to the MS-user and the procedure terminates.

b) The MS attempts to establish an abstract-association with the MTS if one does not already exist. If an abstract-association cannot be established, a remote-bind-error is returned to the MS-user and the procedure terminates.

c) The MS invokes the Register abstract-operation over its abstract-association with the MTS, containing the arguments of the original abstract-operation.

d) The result or error returned by the MTS is returned to the MS-user.

e) The MS may choose to terminate the abstract-association with the MTS either when the MS-user terminates its abstract-association with the MS or after a certain period of inactivity.

f) The scope of permitted changes by the MS-user to the user-security-labels shall be confined by the security-policy in force. Some security-policies may only permit user-security-labels to the changed in this way if a secure link is employed. Other local means of changing user-security-labels in a secure manner may be provided.

16.3.2 Invocation of the Change-credentials abstract-operation

When the MTS invokes the Change-credentials abstract-operation, the MS performs the following actions:

a) The MS verifies that the supplied arguments are valid for the Change-credentials abstract-operation. See 8.4.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4. If the old credentials are incorrect or the new credentials are not acceptable, an error is returned and the procedure terminates.

b) The MS stores the new credentials for use on subsequent occasions when it binds to the MTS and returns a result to the MTS.

16.3.3 Performance of the Change-credentials abstract-operation

When the MS-user invokes the Change-credentials abstract-operation, the MS performs the following actions:

a) The MS verifies that the supplied arguments are valid for the Change-credentials abstract-operation. See 8.4.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4. Subject to local policy, the MS may impose restrictions on the use of the Change-credentials abstract-operation by the MS-user; if these restrictions are violated, an abstract-error is returned to the MS-user and the procedure terminates.

b) The MS attempts to establish an abstract-association with the MTS if one does not already exist. If an abstract-association cannot be established, a remote-bind-error is returned to the MS-user and the procedure terminates.

c) The MS invokes the Change-credentials abstract-operation over its abstract-association with the MTS.

d) The result or error returned by the MTS is returned to the MS-user. If the abstract-operation was performed successfully, the MS stores the new credentials.

e) The MS may choose to terminate the abstract-association with the MTS either when the MS-user terminates its abstract-association with the MS or after a certain period of inactivity.

17 Ports realization

This clause describes how the Retrieval, the MS-submission and the Administration Ports of the MS abstract-service are provided. For a description of how the MTS abstract-service provides the Delivery, the Submission and the Administration Ports, refer to clause 8 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

17.1 Retrieval Port

The Retrieval Port abstract-services are realized on a one-to-one basis between abstract-operations and real operations in the Message Retrieval Service Element 1988 (MRSE-88) and Message Retrieval Service Element 1994 (MRSE-94) which are defined in ITU-T Rec. X.419 | ISO/IEC 10021-6.

NOTE – The MRSE-88 applies when using a 1988 Application Context; the MRSE-94 applies when using a 1994 Application Context.

17.2 MS-submission Port

The MS-submission Port abstract-services are realized on a one-to-one basis between abstract-operations and real operations in the Message Submission Service Element (MSSE) and MS Message Submission Service Element (MS-MSSE) which are defined in ITU-T Rec. X.419 | ISO/IEC 10021-6.

NOTE – The MSSE applies when using a 1988 Application Context; the MS-MSSE applies when using a 1994 Application Context.

17.3 Administration Port

The Administration Port abstract-services are realized on a one-to-one basis between abstract-operations and real operations in the Message Administration Service Element 1988 (MASE-88) and the Message Administration Service Element 1994 (MASE-94) which are defined in ITU-T Rec. X.419 | ISO/IEC 10021-6.

NOTE – The MASE-88 applies when using a 1988 Application Context; the MASE-94 applies when using a 1994 Application Context.

Annex A

Formal assignment of Object Identifiers

(This annex forms an integral part of this Recommendation | International Standard)

All Object Identifiers assigned in this Service Definition are formally assigned in the present annex using ASN.1. The specified values are cited in the ASN.1 modules of subsequent annexes.

This annex is definitive for all values except those for ASN.1 modules and for the whole subject matter of this Service Definition. The definitive assignments for the former occur in the modules themselves. The latter is fixed. Other references to the values assigned to modules appear in IMPORT clauses.

```
MSObjectIdentifiers {joint-iso-itu-t mhs(6) ms(4) modules(0) object-identifiers(0) version-1994(0)};
DEFINITIONS ::=

BEGIN

-- Prologue

-- Exports everything

IMPORTS
        ID, id-ms
            ----
            FROM MHSObjectIdentifiers {joint-iso-itu-t mhs(6) arch(5) modules(0) object-identifiers(0) version-1994(0)};

-- Categories

id-mod    -- modules --              ID ::= {id-ms 0}
id-ot     -- objects --              ID ::= {id-ms 1}
id-pt     -- port types --           ID ::= {id-ms 2}
id-att    -- attribute types --      ID ::= {id-ms 3}
id-act    -- auto-action types --    ID ::= {id-ms 4}
id-crt    -- contracts --            ID ::= {id-ms 5}
id-cp     -- connection-packages --  ID ::= {id-ms 6}
id-aae    -- auto-action-errors --   ID ::= {id-ms 7}
id-mr     -- matching-rules --       ID ::= {id-ms 8}

-- Modules

id-mod-object-identifiers          ID ::= {id-mod 0} -- not definitive
id-mod-abstract-service            ID ::= {id-mod 1} -- not definitive
id-mod-attribute-types             ID ::= {id-mod 2} -- not definitive
id-mod-action-types                ID ::= {id-mod 3} -- not definitive
id-mod-upper-bounds                ID ::= {id-mod 4} -- not definitive
id-mod-matching-rules              ID ::= {id-mod 5} -- not definitive

-- Objects

id-ot-ms                           ID ::= {id-ot 0}
id-ot-ms-user                      ID ::= {id-ot 1}

-- Port types

id-pt-retrieval-88                 ID ::= {id-pt 0}
id-pt-retrieval-94                 ID ::= {id-pt 1}
id-pt-ms-submission                ID ::= {id-pt 2}

-- Contracts

id-crt-ms-access-88                ID ::= {id-crt 0}
id-crt-ms-access-94                ID ::= {id-crt 1}

-- Connection-packages

id-cp-ms-connection                ID ::= {id-cp 0}

-- Attribute-types
id-att-ac-correlated-report-list                ID ::= {id-att 42}
id-att-ac-report-summary                        ID ::= {id-att 43}
id-att-ac-uncorrelated-report-list              ID ::= {id-att 44}
```

id-att-auto-action-error	ID ::= {id-att 46}
id-att-auto-action-registration-identifier	ID ::= {id-att 47}
id-att-auto-action-subject-entry	ID ::= {id-att 48}
id-att-auto-action-type	ID ::= {id-att 49}
id-att-child-sequence-numbers	ID ::= {id-att 0}
id-att-content	ID ::= {id-att 1}
id-att-content-confidentiality-algorithm-identifier	ID ::= {id-att 2}
id-att-content-correlator	ID ::= {id-att 3}
id-att-content-identifier	ID ::= {id-att 4}
id-att-content-integrity-check	ID ::= {id-att 5}
id-att-content-length	ID ::= {id-att 6}
id-att-content-returned	ID ::= {id-att 7}
id-att-content-type	ID ::= {id-att 8}
id-att-conversion-with-loss-prohibited	ID ::= {id-att 9}
id-att-converted-EITs	ID ::= {id-att 10}
id-att-creation-time	ID ::= {id-att 11}
id-att-deferred-delivery-cancellation-time	ID ::= {id-att 50}
id-att-deferred-delivery-time	ID ::= {id-att 51}
id-att-deletion-time	ID ::= {id-att 52}
id-att-delivered-EITs	ID ::= {id-att 12}
id-att-delivery-flags	ID ::= {id-att 13}
id-att-dl-expansion-history	ID ::= {id-att 14}
id-att-dl-expansion-prohibited	ID ::= {id-att 53}
id-att-entry-type	ID ::= {id-att 16}
id-att-internal-trace-information	ID ::= {id-att 54}
id-att-latest-delivery-time	ID ::= {id-att 55}
id-att-marked-for-deletion	ID ::= {id-att 56}
id-att-message-delivery-envelope	ID ::= {id-att 18}
id-att-message-delivery-time	ID ::= {id-att 20}
id-att-message-group-name	ID ::= {id-att 57}
id-att-message-identifier	ID ::= {id-att 19}
id-att-message-notes	ID ::= {id-att 58}
id-att-message-origin-authentication-check	ID ::= {id-att 21}
id-att-message-security-label	ID ::= {id-att 22}
id-att-message-submission-envelope	ID ::= {id-att 59}
id-att-message-submission-time	ID ::= {id-att 23}
id-att-message-token	ID ::= {id-att 24}
id-att-ms-originated	ID ::= {id-att 60}
id-att-ms-submission-error	ID ::= {id-att 61}
id-att-original-EITs	ID ::= {id-att 25}
id-att-originally-intended-recipient-name	ID ::= {id-att 17}
id-att-originating-MTA-certificate	ID ::= {id-att 62}
id-att-originator-certificate	ID ::= {id-att 26}
id-att-originator-name	ID ::= {id-att 27}
id-att-originator-report-request	ID ::= {id-att 63}
id-att-originator-return-address	ID ::= {id-att 64}
id-att-other-recipient-names	ID ::= {id-att 28}
id-att-parent-sequence-number	ID ::= {id-att 29}
id-att-per-message-indicators	ID ::= {id-att 65}
id-att-per-recipient-message-submission-fields	ID ::= {id-att 66}
id-att-per-recipient-probe-submission-fields	ID ::= {id-att 67}
id-att-per-recipient-report-delivery-fields	ID ::= {id-att 30}
id-att-priority	ID ::= {id-att 31}
id-att-probe-origin-authentication-check	ID ::= {id-att 68}
id-att-probe-submission-envelope	ID ::= {id-att 69}
id-att-proof-of-delivery-request	ID ::= {id-att 32}
id-att-proof-of-submission	ID ::= {id-att 70}
id-att-recipient-names	ID ::= {id-att 71}
id-att-recipient-reassignment-prohibited	ID ::= {id-att 72}
id-att-redirection-history	ID ::= {id-att 33}
id-att-report-delivery-envelope	ID ::= {id-att 34}
id-att-reporting-DL-name	ID ::= {id-att 35}
id-att-reporting-MTA-certificate	ID ::= {id-att 36}
id-att-report-origin-authentication-check	ID ::= {id-att 37}
id-att-retrieval-status	ID ::= {id-att 15}
id-att-security-classification	ID ::= {id-att 38}
id-att-sequence-number	ID ::= {id-att 39}
id-att-storage-period	ID ::= {id-att 73}

```
id-att-storage-time                                  ID ::= {id-att 74}
id-att-subject-submission-identifier                 ID ::= {id-att 40}
id-att-this-recipient-name                           ID ::= {id-att 41}
id-att-trace-information                             ID ::= {id-att 75}

-- Auto-action-types

id-act-ipm-auto-forward          ID ::= {id-act 0} -- Reserved for use in ITU-T Rec. X.420 | ISO/IEC 10021-7
id-act-auto-alert                ID ::= {id-act 1}
id-act-auto-correlate-reports    ID ::= {id-act 2}
id-act-auto-delete               ID ::= {id-act 3}
id-act-auto-modify               ID ::= {id-act 4}

-- Auto-action errors

id-aae-auto-alert-error          ID ::= {id-aae 0}

-- Matching-rules

id-mr-content-correlator-match                                ID ::= {id-mr 1}
id-mr-content-identifier-match                                ID ::= {id-mr 2}
id-mr-ms-single-substring-list-elements-match                 ID ::= {id-mr 3}
id-mr-ms-single-substring-list-match                          ID ::= {id-mr 4}
id-mr-ms-single-substring-match                               ID ::= {id-mr 5}
id-mr-ms-substrings-match                                     ID ::= {id-mr 6}
id-mr-msstring-case-sensitive-match                           ID ::= {id-mr 7}
id-mr-msstring-list-elements-match                            ID ::= {id-mr 8}
id-mr-msstring-list-match                                     ID ::= {id-mr 9}
id-mr-msstring-match                                          ID ::= {id-mr 10}
id-mr-msstring-ordering-match                                 ID ::= {id-mr 11}
id-mr-mts-identifier-match                                    ID ::= {id-mr 12}
id-mr-oraddress-elements-match                                ID ::= {id-mr 13}
id-mr-oraddress-match                                         ID ::= {id-mr 14}
id-mr-oraddress-substring-elements-match                      ID ::= {id-mr 15}
id-mr-orname-elements-match                                   ID ::= {id-mr 16}
id-mr-orname-match                                            ID ::= {id-mr 17}
id-mr-orname-single-element-match                             ID ::= {id-mr 18}
id-mr-orname-substring-elements-match                         ID ::= {id-mr 19}
id-mr-redirection-or-dl-expansion-elements-match              ID ::= {id-mr 20}
id-mr-redirection-or-dl-expansion-match                       ID ::= {id-mr 21}
id-mr-redirection-or-dl-expansion-substring-elements-match    ID ::= {id-mr 22}
id-mr-redirection-reason-match                                ID ::= {id-mr 23}
id-mr-value-count-match                                       ID ::= {id-mr 24}

END -- of MSObjectIdentifiers
```

Annex B

Formal definition of the Message Store abstract-service

(This annex forms an integral part of this Recommendation | International Standard)

This annex, a supplement to Section 2, formally defines the Message Store abstract-service. It employs ASN.1 and the MHS-OBJECT, PORT, ABSTRACT-OPERATION, and ABSTRACT-ERROR information object classes of ITU-T Rec. X.411 | ISO/IEC 10021-4, and the CONTRACT and CONNECTION-PACKAGE information object classes of ITU-T Rec. X.880 | ISO/IEC 13712-1.

NOTE – The use of the MHS-OBJECT, PORT, ABSTRACT-OPERATION, and ABSTRACT-ERROR information object classes, which are derived from the ROS-OBJECT-CLASS, OPERATION-PACKAGE, OPERATION and ERROR information object classes of ROS, does not imply that the abstract-operations and abstract-errors are invoked and reported across the boundary between open-systems in every instance. However, frequently this will be done. Just how this is accomplished is the subject of ITU-T Rec. X.419 | ISO/IEC 10021-6.

```
MSAbstractService {joint-iso-itu-t mhs(6) ms(4) modules(0) abstract-service(1) version-1994(0)}
DEFINITIONS ::=

BEGIN

-- Prologue

-- Exports everything

IMPORTS

        -- MTS information object classes

    ABSTRACT-ERROR, ABSTRACT-OPERATION, EXTENSION, MHS-OBJECT, PORT,

        -- MTS objects and ports

    administration, delivery, mts-user, submission,

        -- MTS abstract-operations and abstract-errors

    cancel-deferred-delivery, element-of-service-not-subscribed, inconsistent-request, new-credentials-unacceptable,
    old-credentials-incorrectly-specified, originator-invalid, recipient-improperly-specified, remote-bind-error,
    security-error, submission-control, submission-control-violated, unsupported-critical-function,

        -- MTS abstract-service data-types

    Credentials, InitiatorCredentials, MessageSubmissionArgument, MessageSubmissionResult,
    ORAddressAndOrDirectoryName, ProbeSubmissionArgument, ProbeSubmissionResult, ResponderCredentials,
    SecurityContext, SecurityLabel
        FROM MTSAbstractService {joint-iso-itu-t mhs(6) mts(3) modules(0) mts-abstract-service(1)
                                 version-1994(0)}

        -- MTS abstract-service 1988 ports

    administration-88
        ----
        FROM MTSAbstractService88 {joint-iso-itu-t mhs(6) mts(3) modules(0)
                                   mts-abstract-service(1) version-1988(1988)}

        -- MTS abstract-service upper bounds

    ub-content-types, ub-encoded-information-types, ub-labels-and-redirections
        ----
        FROM MTSUpperBounds {joint-iso-itu-t mhs(6) mts(3) modules(0) upper-bounds(3)}

        -- MS attribute table

    AttributeTable
        ----
        FROM MSGeneralAttributeTypes {joint-iso-itu-t mhs(6) ms(4) modules(0)
                                      general-attribute-types(2) version-1994(0)}
```

```
        -- MS matching rule table

    MatchingRuleTable
        ----
        FROM MSMatchingRules {joint-iso-itu-t mhs(6) ms(4) modules(0) general-matching-rules(5)}

        -- MS auto-action-table and auto-action-error table

    AutoActionTable, AutoActionErrorTable
        ----
        FROM MSGeneralAutoActionTypes {joint-iso-itu-t mhs(6) ms(4) modules(0)
                                       general-auto-action-types(3) version-1994(0)}

        -- MS object-identifiers

    id-cp-ms-connection, id-crt-ms-access-88, id-crt-ms-access-94, id-ot-ms, id-ot-ms-user, id-pt-retrieval-88,
    id-pt-retrieval-94, id-pt-ms-submission
        ----
        FROM MSObjectIdentifiers {joint-iso-itu-t mhs(6) ms(4) modules(0) object-identifiers(0)
                                  version-1994(0)}

        -- MS Access abstract-operation and error codes

    err-attribute-error, err-auto-action-request-error, err-ms-extension-error,
    err-delete-error, err-entry-class-error, err-fetch-restriction-error,
    err-invalid-parameters-error, err-message-group-error, err-modify-error,
    err-range-error, err-security-error, err-sequence-number-error, err-service-error,
    err-register-ms-error, op-alert, op-delete, op-fetch, op-list, op-modify,
    op-ms-message-submission, op-ms-probe-submission, op-register-ms, op-summarize
        ----
        FROM MSAccessProtocol {joint-iso-itu-t mhs(6) protocols(0) modules(0)
                               ms-access-protocol(2) version-1994(0)}

        -- MS abstract-service upper bounds

    ub-attributes-supported, ub-attribute-values, ub-auto-action-errors, ub-auto-actions, ub-auto-registrations,
    ub-default-registrations, ub-entry-classes, ub-error-reasons, ub-extensions, ub-group-depth,
    ub-group-descriptor-length, ub-group-part-length, ub-matching-rules, ub-message-groups, ub-messages,
    ub-modifications, ub-per-entry, ub-per-auto-action, ub-service-information-length, ub-summaries, ub-
    supplementary-info-length,
    ub-ua-registration-identifier-length, ub-ua-registrations, ub-restrictions
        ----
        FROM MSUpperBounds {joint-iso-itu-t mhs(6) ms(4) modules(0) upper-bounds(4)
                            version-1994(0)}

        -- MATCHING-RULE information object class

    MATCHING-RULE
        ----
        FROM InformationFramework {joint-iso-ccitt ds(5) modules(1) informationFramework(1) 2}

        -- Remote Operations

    CONTRACT, CONNECTION-PACKAGE
        ----
        FROM Remote-Operations-Information-Objects {joint-iso-ccitt remote-operations(4)
                                                    informationObjects(5) version1(0)}

    emptyUnbind
        FROM Remote-Operations-Useful-Definitions {joint-iso-ccitt remote-operations(4)
                                                   useful-definitions(7) version1(0)};

-- MS Abstract Objects

ms MHS-OBJECT ::= {
        IS          {mts-user}
        RESPONDS    {ms-access-contract-88 | ms-access-contract-94}
        ID          id-ot-ms }

ms-user MHS-OBJECT ::= {
        INITIATES   {ms-access-contract-88 | ms-access-contract-94}
        ID          id-ot-ms-user }
```

```
-- Contracts

ms-access-contract-94 CONTRACT ::= {
        CONNECTION                  ms-connect
        INITIATOR CONSUMER OF       {retrieval | ms-submission | administration}
        ID                          id-crt-ms-access-94 }

ms-access-contract-88 CONTRACT ::= {
        CONNECTION                  ms-connect -- with all 1994 extensions omitted --
        INITIATOR CONSUMER OF       {retrieval-88 | submission | administration-88}
        ID                          id-crt-ms-access-88 }

-- Connection-package

ms-connect CONNECTION-PACKAGE ::= {
        BIND        ms-bind
        UNBIND      ms-unbind
        ID          id-cp-ms-connection }

-- MS Ports

retrieval PORT ::= {
        CONSUMER INVOKES {summarize | list | fetch | delete | register-MS,
                          ... -- 1994 extension addition --,
                          modify}
        SUPPLIER INVOKES  {alert}
        ID                id-pt-retrieval-94 }

retrieval-88 PORT ::= {
        -- With all 1994 extensions to the abstract-operations absent --
        CONSUMER INVOKES {summarize | list | fetch | delete | register-MS}
        SUPPLIER INVOKES  {alert}
        ID                id-pt-retrieval-88 }

ms-submission PORT ::= {
        CONSUMER INVOKES {ms-message-submission | ms-probe-submission | ms-cancel-deferred-delivery}
        SUPPLIER INVOKES  {ms-submission-control}
        ID                id-pt-ms-submission }

-- ATTRIBUTE information object class

ATTRIBUTE ::= CLASS {
        &id                         AttributeType UNIQUE,
        &Type,
        &equalityMatch              MATCHING-RULE OPTIONAL,
        &substringsMatch            MATCHING-RULE OPTIONAL,
        &orderingMatch              MATCHING-RULE OPTIONAL,
        &numeration                 ENUMERATED {single-valued(0), multi-valued(1)},
                    -- 1994 extension --
        &OtherMatches               MATCHING-RULE OPTIONAL }

WITH SYNTAX {
        WITH ATTRIBUTE-SYNTAX       &Type,
        [EQUALITY MATCHING-RULE     &equalityMatch,]
        [SUBSTRINGS MATCHING-RULE   &substringsMatch,]
        [ORDERING MATCHING-RULE     &orderingMatch,]
        [OTHER MATCHING-RULES       &OtherMatches,]
        NUMERATION                  &numeration,
        ID                          &id }

Attribute ::= SEQUENCE {
        attribute-type              ATTRIBUTE.&id ({AttributeTable}),
        attribute-values            SEQUENCE SIZE (1.. ub-attribute-values) OF ATTRIBUTE.&Type
                                                    ({AttributeTable} {@attribute-type}) }
```

```
AttributeType ::= OBJECT IDENTIFIER

-- AUTO-ACTION information object class

AUTO-ACTION ::= CLASS {
        &id                             AutoActionType UNIQUE,
        &RegistrationParameter          OPTIONAL,
        &Errors                         AUTO-ACTION-ERROR OPTIONAL }

WITH SYNTAX {
        [REGISTRATION PARAMETER IS  &RegistrationParameter]
        [ERRORS                     &Errors]
        IDENTIFIED BY               &id }

AutoActionType ::= OBJECT IDENTIFIER

AutoActionRegistration ::= SEQUENCE {
        auto-action-type            AUTO-ACTION.&id ({AutoActionTable}),
        registration-identifier     [0] INTEGER (1..ub-per-auto-action) DEFAULT 1,
        registration-parameter      [1] AUTO-ACTION.&RegistrationParameter ({AutoActionTable}
                                        {@auto-action-type}) OPTIONAL }

-- AUTO-ACTION-ERROR information object class

AUTO-ACTION-ERROR ::= ABSTRACT-ERROR

AutoActionError ::= SET {
        error-code          [0] AUTO-ACTION-ERROR.&errorCode ({AutoActionErrorTable}),
        error-parameter     [1] AUTO-ACTION-ERROR.&ParameterType ({AutoActionErrorTable} {@error-code})
                                OPTIONAL }

-- MS-EXTENSION information object class

MS-EXTENSION ::= TYPE-IDENTIFIER

MSExtensionItem ::= INSTANCE OF MS-EXTENSION

MSExtensions ::= SEQUENCE SIZE (1..ub-extensions) OF MSExtensionItem

-- Common data-types related to the information model

EntryClass ::= INTEGER {
        delivery            (0),
        -- 1994 extensions --
        submission          (1),
        draft               (2),
        stored-message      (3),
        delivery-log        (4),
        submission-log      (5),
        message-log         (6),
        auto-action-log     (7) } (0..ub-entry-classes)

EntryType ::= INTEGER {
        delivered-message       (0),
        delivered-report        (1),
        returned-content        (2),
        -- 1994 extensions --
        submitted-message       (3),
        submitted-probe         (4),
        draft-message           (5),
        auto-action-event       (6) }

SequenceNumber ::= INTEGER (0..ub-messages)

RetrievalStatus ::= INTEGER {
        new             (0),
        listed          (1),
        processed       (2) }

MessageGroupName ::= SEQUENCE SIZE (1..ub-group-depth) OF GroupNamePart

GroupNamePart ::= GeneralString (SIZE (1..ub-group-part-length))
```

```
-- MS-bind abstract-operation

ms-bind ABSTRACT-OPERATION ::= {
        ARGUMENT        MSBindArgument
        RESULT          MSBindResult
        ERRORS          {ms-bind-error} }

MSBindArgument ::= SET {
        initiator-name                  ORAddressAndOrDirectoryName,
        initiator-credentials           [2] InitiatorCredentials,
        security-context                [3] IMPLICIT SecurityContext OPTIONAL,
        fetch-restrictions              [4] Restrictions OPTIONAL -- default is none--,
        ms-configuration-request        [5] BOOLEAN DEFAULT FALSE,
                                        -- 1994 extensions --
        ua-registration-identifier      [6] RegistrationIdentifier OPTIONAL,
        bind-extensions                 [7] MSExtensions OPTIONAL }

Restrictions ::= SET {
        allowed-content-types           [0] SET SIZE (1..ub-content-types) OF OBJECT IDENTIFIER OPTIONAL
                                                                -- default is no restriction --,
        allowed-EITs                    [1] MS-EITs OPTIONAL -- default is no restriction --,
        maximum-attribute-length        [2] INTEGER OPTIONAL -- default is no restriction -- }

MS-EITs ::= SET SIZE (1..ub-encoded-information-types) OF MS-EIT

MS-EIT ::= OBJECT IDENTIFIER

RegistrationIdentifier ::= PrintableString (SIZE (1..ub-ua-registration-identifier-length))

MSBindResult ::= SET {
        responder-credentials           [2] ResponderCredentials,
        available-auto-actions          [3] SET SIZE (1..ub-auto-actions) OF AUTO-ACTION.&id
                                                ({AutoActionTable}) OPTIONAL,
        available-attribute-types       [4] SET SIZE (1..ub-attributes-supported) OF ATTRIBUTE.&id
                                                        ({AttributeTable}) OPTIONAL,
        alert-indication                [5] BOOLEAN DEFAULT FALSE,
        content-types-supported         [6] SET SIZE (1..ub-content-types) OF OBJECT IDENTIFIER OPTIONAL,
                                        -- 1994 extensions --
        entry-classes-supported         [7] SET SIZE (1..ub-entry-classes) OF EntryClass OPTIONAL,
        matching-rules-supported        [8] SET SIZE (1..ub-matching-rules) OF OBJECT IDENTIFIER OPTIONAL,
        additional-capabilities         [9] MSExtensions OPTIONAL,
        message-group-depth             [10] INTEGER (1..ub-group-depth) OPTIONAL,
        auto-action-error-indication    [11] AutoActionErrorIndication OPTIONAL,
        unsupported-extensions          [12] SET SIZE (1..ub-extensions) OF OBJECT IDENTIFIER OPTIONAL,
        ua-registration-id-unknown      [13] BOOLEAN DEFAULT FALSE,
        service-information             [14] GeneralString (SIZE (1..ub-service-information-length)) OPTIONAL }

AutoActionErrorIndication ::= CHOICE {
        indication-only                 [0] NULL,
        auto-action-log-entry           [1] SequenceNumber }

ms-bind-error ABSTRACT-ERROR ::= {
        PARAMETER  CHOICE {
            unqualified-error           BindProblem,
                                        -- 1994 extension --
            qualified-error             SET {
                bind-problem                    [0] BindProblem,
                supplementary-information       [1] GeneralString (SIZE (1..ub-supplementary-info-length)) OPTIONAL,
                bind-extension-errors           [2] SET SIZE (1..ub-extensions) OF OBJECT IDENTIFIER OPTIONAL } } }

BindProblem ::= ENUMERATED {
        authentication-error            (0),
        unacceptable-security-context   (1),
        unable-to-establish-association (2),
        ... -- 1994 extension addition --,
        bind-extension-problem          (3) }

-- MS Unbind abstract-operation

ms-unbind ABSTRACT-OPERATION ::= emptyUnbind
```

```
-- Common data-types

Range ::= CHOICE {
        sequence-number-range       [0]  NumberRange,
        creation-time-range         [1]  TimeRange }

NumberRange ::= SEQUENCE {
        from        [0]  SequenceNumber OPTIONAL -- omitted means no lower bound --,
        to          [1]  SequenceNumber OPTIONAL -- omitted means no upper bound -- }

TimeRange ::= SEQUENCE {
        from        [0]  CreationTime OPTIONAL -- omitted means no lower bound --,
        to          [1]  CreationTime OPTIONAL -- omitted means no upper bound -- }

CreationTime ::= UTCTime

Filter ::= CHOICE {
        item        [0]  FilterItem,
        and         [1]  SET OF Filter,
        or          [2]  SET OF Filter,
        not         [3]  Filter }

FilterItem ::= CHOICE {
        equality                [0]  AttributeValueAssertion,
        substrings              [1]  SEQUENCE {
            type                         ATTRIBUTE.&id ({AttributeTable}),
            strings                      SEQUENCE OF CHOICE {
                initial                      [0]  ATTRIBUTE.&Type ({AttributeTable} {@substrings.type}),
                any                          [1]  ATTRIBUTE.&Type ({AttributeTable} {@substrings.type}),
                final                        [2]  ATTRIBUTE.&Type ({AttributeTable} {@substrings.type}) } },
        greater-or-equal        [2]  AttributeValueAssertion,
        less-or-equal           [3]  AttributeValueAssertion,
        present                 [4]  ATTRIBUTE.&id ({AttributeTable}),
        approximate-match       [5]  AttributeValueAssertion,
                                     -- 1994 extension --
        other-match             [6]  MatchingRuleAssertion }

MatchingRuleAssertion ::= SEQUENCE {
        matching-rule       [0]  MATCHING-RULE.&id ({MatchingRuleTable}),
        attribute-type      [1]  ATTRIBUTE.&id,
        match-value         [2]  MATCHING-RULE.&AssertionType ({MatchingRuleTable} {@matching-rule}) }

AttributeValueAssertion ::= SEQUENCE {
        attribute-type      ATTRIBUTE.&id ({AttributeTable}),
        attribute-value     ATTRIBUTE.&Type ({AttributeTable} {@attribute-type}) }

Selector ::= SET {
        child-entries   [0]  BOOLEAN DEFAULT FALSE,
        range           [1]  Range OPTIONAL -- default is unbounded --,
        filter          [2]  Filter OPTIONAL -- default is all entries within the specified range --,
        limit           [3]  INTEGER (1..ub-messages) OPTIONAL,
        override        [4]  OverrideRestrictions OPTIONAL -- by default, any fetch-restrictions in force apply -- }

OverrideRestrictions ::= BIT STRING {
        override-content-types-restriction          (0),
        override-EITs-restriction                   (1),
        override-attribute-length-restriction       (2) } (SIZE (1.. ub-restrictions))

EntryInformationSelection ::= SET SIZE (0..ub-per-entry) OF AttributeSelection

AttributeSelection ::= SET {
        type        ATTRIBUTE.&id ({AttributeTable}),
        from        [0]  INTEGER (1..ub-attribute-values) OPTIONAL -- used if type is multi valued --,
        count       [1]  INTEGER (1..ub-attribute-values) OPTIONAL -- used if type is multi valued -- }

EntryInformation ::= SEQUENCE {
        sequence-number         SequenceNumber,
        attributes              SET SIZE (1..ub-per-entry) OF Attribute OPTIONAL,
                                -- 1994 extension --
        value-count-exceeded    [0] SET SIZE (1..ub-per-entry) OF AttributeValueCount OPTIONAL }
```

```
AttributeValueCount ::= SEQUENCE {
        type      [0] ATTRIBUTE.&id ({AttributeTable}),
        total     [1] INTEGER }

MSSubmissionOptions ::= SET {
        object-entry-class          [0] EntryClass (submission|submission-log|draft) OPTIONAL,
        disable-auto-modify         [1] BOOLEAN DEFAULT FALSE,
        add-message-group-names     [2] SET SIZE (1..ub-message-groups) OF MessageGroupName OPTIONAL,
        ms-submission-extensions    [3] MSExtensions OPTIONAL }

CommonSubmissionResults ::= SET {
        created-entry                   [0] SequenceNumber OPTIONAL,
        auto-action-error-indication    [1] AutoActionErrorIndication OPTIONAL,
        ms-submission-result-extensions [2] MSExtensions OPTIONAL }

-- Retrieval Port abstract-operations

summarize ABSTRACT-OPERATION ::= {
        ARGUMENT SummarizeArgument
        RESULT      SummarizeResult
        ERRORS      {attribute-error | invalid-parameters-error | range-error |
                     security-error | sequence-number-error | service-error,
                     ... -- 1994 extension additions --,
                     entry-class-error | ms-extension-error}
        CODE        op-summarize }

SummarizeArgument ::= SET {
        entry-class             [0] EntryClass DEFAULT delivery,
        selector                [1] Selector,
        summary-requests        [2] SEQUENCE SIZE (1..ub-summaries) OF ATTRIBUTE.&id ({AttributeTable})
                                    OPTIONAL  -- absent if no summaries are requested --,
                                -- 1994 extension --
        summarize-extensions    [3] MSExtensions OPTIONAL }

SummarizeResult ::= SET {
        next                            [0] SequenceNumber OPTIONAL,
        count                           [1] INTEGER (0..ub-messages) -- of the entries selected --,
        span                            [2] Span OPTIONAL -- of the entries selected,
                                                          -- omitted if count is zero --,
        summaries                       [3] SEQUENCE SIZE (1..ub-summaries) OF Summary OPTIONAL,
                                        --1994 extension --
        summarize-result-extensions     [4] MSExtensions OPTIONAL }

Span ::= SEQUENCE {
        lowest      [0] SequenceNumber,
        highest     [1] SequenceNumber }

Summary ::= SET {
        absent      [0] INTEGER (1..ub-messages) OPTIONAL -- count of entries where attribute is absent --,
        present     [1] SET SIZE (1..ub-attribute-values) OF -- one for each attribute value present --
                            SEQUENCE {
                                type    ATTRIBUTE.&id ({AttributeTable}),
                                value   ATTRIBUTE.&Type ({AttributeTable} {@.type}),
                                count   INTEGER (1..ub-messages) } OPTIONAL }

--

list ABSTRACT-OPERATION ::= {
        ARGUMENT ListArgument
        RESULT      ListResult
        ERRORS      {attribute-error | invalid-parameters-error | range-error |
                     security-error | sequence-number-error | service-error,
                     ... -- 1994 extension additions --,
                     entry-class-error | ms-extension-error}
        CODE        op-list }
```

```
ListArgument ::= SET {
        entry-class             [0] EntryClass DEFAULT delivery,
        selector                [1] Selector,
        requested-attributes    [3] EntryInformationSelection OPTIONAL,
                                -- 1994 extension --
        list-extensions         [4] MSExtensions OPTIONAL }

ListResult ::= SET {
        next                    [0] SequenceNumber OPTIONAL,
        requested               [1] SEQUENCE SIZE (1..ub-messages) OF EntryInformation OPTIONAL
                                    -- omitted if none found --,
                                -- 1994 extension --
        list-result-extensions  [2] MSExtensions OPTIONAL }

--

fetch ABSTRACT-OPERATION ::= {
        ARGUMENT        FetchArgument
        RESULT          FetchResult
        ERRORS          {attribute-error | fetch-restriction-error | invalid-parameters-error | range-error |
                        security-error | sequence-number-error | service-error,
                        ... -- 1994 extension additions --,
                        entry-class-error | ms-extension-error}
        CODE            op-fetch }

FetchArgument ::= SET {
        entry-class             [0] EntryClass DEFAULT delivery,
        item                    CHOICE {
                search                  [1] Selector,
                precise                 [2] SequenceNumber },
        requested-attributes            [3] EntryInformationSelection OPTIONAL,
                                -- 1994 extension --
        fetch-extensions        [4] MSExtensions OPTIONAL }

FetchResult ::= SET {
        entry-information       [0] EntryInformation OPTIONAL -- if an entry was selected --,
        list                    [1] SEQUENCE SIZE (1..ub-messages) OF SequenceNumber OPTIONAL,
        next                    [2] SequenceNumber OPTIONAL,
                                -- 1994 extension --
        fetch-result-extensions [3] MSExtensions OPTIONAL }

--

delete ABSTRACT-OPERATION ::= {
        ARGUMENT        DeleteArgument
        RESULT          DeleteResult
        ERRORS          {delete-error | invalid-parameters-error | range-error | security-error |
                        sequence-number-error | service-error,
                        ... -- 1994 extension additions --,
                        entry-class-error | ms-extension-error}
        CODE            op-delete }

DeleteArgument ::= SET {
        entry-class             [0] EntryClass DEFAULT delivery,
        items                   CHOICE {
                selector                [1] Selector,
                sequence-numbers        [2] SET SIZE (1..ub-messages) OF SequenceNumber },
                                -- 1994 extension --
        delete-extensions       [3] MSExtensions OPTIONAL }

DeleteResult ::= CHOICE {
        delete-result-88        NULL,
                                -- 1994 extension --
        delete-result-94        SET {
            entries-deleted             [0] SEQUENCE SIZE (1..ub-messages) OF SequenceNumber OPTIONAL,
            delete-result-extensions    [1] MSExtensions OPTIONAL } }
```

```
--

register-MS ABSTRACT-OPERATION ::= {
        ARGUMENT        Register-MSArgument
        RESULT          Register-MSResult
        ERRORS          {attribute-error | auto-action-request-error | invalid-parameters-error |
                        security-error | service-error | old-credentials-incorrectly-specified |
                        new-credentials-unacceptable,
                        ... -- 1994 extension additions --,
                        message-group-error | ms-extension-error | register-ms-error}
        CODE            op-register-ms }

Register-MSArgument ::= SET {
        auto-action-registrations       [0] SET SIZE (1..ub-auto-registrations) OF AutoActionRegistration OPTIONAL,
        auto-action-deregistrations     [1] SET SIZE (1..ub-auto-registrations) OF AutoActionDeregistration OPTIONAL,
        list-attribute-defaults         [2] SET SIZE (0..ub-default-registrations) OF ATTRIBUTE.&id
                                                ({AttributeTable}) OPTIONAL,
        fetch-attribute-defaults        [3] SET SIZE (0..ub-default-registrations) OF ATTRIBUTE.&id
                                                ({AttributeTable})OPTIONAL,
        change-credentials              [4] SEQUENCE {
            old-credentials                     [0] Credentials,
            new-credentials                     [1] Credentials} OPTIONAL,
        user-security-labels            [5] SET SIZE (1..ub-labels-and-redirections) OF SecurityLabel OPTIONAL,
                                        -- 1994 extensions --
        ua-registrations                [6] SET SIZE (1..ub-ua-registrations) OF UARegistration OPTIONAL,
        submission-defaults             [7] MSSubmissionOptions OPTIONAL,
        message-group-registrations     [8] MessageGroupRegistrations OPTIONAL,
        registration-status-request     [9] RegistrationTypes OPTIONAL,
        register-ms-extensions         [10] MSExtensions OPTIONAL }

AutoActionDeregistration ::= SEQUENCE {
        auto-action-type                AUTO-ACTION.&id ({AutoActionTable}),
        registration-identifier         [0] INTEGER (1..ub-per-auto-action) DEFAULT 1 }

UARegistration ::= SET {
        ua-registration-identifier      [0] RegistrationIdentifier,
        ua-list-attribute-defaults      [1] SET SIZE (0..ub-default-registrations) OF ATTRIBUTE.&id
                                                ({AttributeTable}) OPTIONAL,
        ua-fetch-attribute-defaults     [2] SET SIZE (0..ub-default-registrations) OF ATTRIBUTE.&id
                                                ({AttributeTable}) OPTIONAL,
        ua-submission-defaults          [3] MSSubmissionOptions OPTIONAL,
        content-specific-defaults       [4] MSExtensions OPTIONAL }

MessageGroupRegistrations ::= SEQUENCE SIZE (1..ub-default-registrations) OF CHOICE {
        register-group                  [0] MessageGroupNameAndDescriptor,
        deregister-group                [1] MessageGroupName,
        change-descriptors              [2] MessageGroupNameAndDescriptor }

MessageGroupNameAndDescriptor ::= SET {
        message-group-name              [0] MessageGroupName,
        message-group-descriptor        [1] GeneralString (SIZE (1..ub-group-descriptor-length)) OPTIONAL }

RegistrationTypes ::= SET {
        registrations                   [0] BIT STRING {
            auto-action-registrations       (0),
            list-attribute-defaults         (1),
            fetch-attribute-defaults        (2),
            ua-registrations                (3),
            submission-defaults             (4),
            message-group-registrations     (5) } OPTIONAL,
        extended-registrations          [1] SET OF MS-EXTENSION.&id OPTIONAL,
        restrict-message-groups         [2] MessageGroupsRestriction OPTIONAL }

MessageGroupsRestriction ::= SET {
        parent-group                    [0] MessageGroupName OPTIONAL,
        immediate-descendants-only      [1] BOOLEAN DEFAULT TRUE,
        omit-descriptors                [2] BOOLEAN DEFAULT TRUE }
```

```
Register-MSResult ::= CHOICE {
        no-status-information           NULL,
                                        -- 1994 extension --
        registered-information  SET {
            auto-action-registrations       [0]  SET SIZE (1..ub-auto-registrations) OF AutoActionRegistration
                                                    OPTIONAL,
            list-attribute-defaults         [1]  SET SIZE (1..ub-default-registrations) OF ATTRIBUTE.&id
                                                    ({AttributeTable}) OPTIONAL,
            fetch-attribute-defaults        [2]  SET SIZE (1..ub-default-registrations) OF ATTRIBUTE.&id
                                                    ({AttributeTable}) OPTIONAL,
            ua-registrations                [3]  SET SIZE (1..ub-ua-registrations) OF UARegistration OPTIONAL,
            submission-defaults             [4]  MSSubmissionOptions OPTIONAL,
            message-group-registrations     [5]  SET SIZE (1..ub-message-groups) OF
                                                    MessageGroupNameAndDescriptor OPTIONAL,
            register-ms-result-extensions   [6]  MSExtensions OPTIONAL } }

--

alert ABSTRACT-OPERATION ::= {
        ARGUMENT AlertArgument
        RESULT   AlertResult
        ERRORS   {security-error}
        CODE     op-alert }

AlertArgument ::= SET {
        alert-registration-identifier  [0]  INTEGER (1..ub-auto-actions),
        new-entry                      [2]  EntryInformation OPTIONAL }

AlertResult ::= NULL

--

modify ABSTRACT-OPERATION ::= {
        ARGUMENT     ModifyArgument
        RESULT       ModifyResult
        ERRORS       {attribute-error | invalid-parameters-error | security-error | sequence-number-error |
                     service-error | modify-error | message-group-error | entry-class-error |
                     ms-extension-error,
                     ... -- For future extension additions --}
        CODE         op-modify }

ModifyArgument ::= SET {
        entry-class          [0]  EntryClass DEFAULT delivery,
        entries              CHOICE {
            selector         [1]  Selector,
            specific-entries [2]  SEQUENCE SIZE (1..ub-messages) OF SequenceNumber },
        modifications        [3]  SEQUENCE SIZE (1..ub-modifications) OF EntryModification,
        modify-extensions    [4]  MSExtensions OPTIONAL }

EntryModification ::= SET {
        strict               [0]  BOOLEAN DEFAULT FALSE,
        modification         CHOICE {
            add-attribute    [1]  Attribute,
            remove-attribute [2]  ATTRIBUTE.&id ({AttributeTable}),
            add-values       [3]  OrderedAttribute,
            remove-values    [4]  OrderedAttribute } }

OrderedAttribute ::= SEQUENCE {
        attribute-type   ATTRIBUTE.&id ({AttributeTable}),
        attribute-values SEQUENCE SIZE (1..ub-attribute-values) OF SEQUENCE {
            -- at least one must be specified --
            value        [0]  ATTRIBUTE.&Type ({AttributeTable} {@attribute-type}) OPTIONAL,
            position     [1]  INTEGER (1..ub-attribute-values) OPTIONAL } }

ModifyResult ::= SET {
        entries-modified          [0]  SEQUENCE SIZE (1..ub-messages) OF SequenceNumber OPTIONAL,
        modify-result-extensions  [1]  MSExtensions OPTIONAL }
```

```
-- MS-submission Port abstract-operations

ms-message-submission ABSTRACT-OPERATION ::= {
        ARGUMENT        MSMessageSubmissionArgument
        RESULT          MSMessageSubmissionResult
        ERRORS          {submission-control-violated | element-of-service-not-subscribed | originator-invalid |
                        recipient-improperly-specified | inconsistent-request | security-error |
                        unsupported-critical-function | remote-bind-error,
                        ... -- 1994 extension additions --,
                        ms-extension-error | message-group-error | entry-class-error | service-error}
        CODE            op-ms-message-submission }

MSMessageSubmissionArgument ::= SEQUENCE {
        COMPONENTS OF       MessageSubmissionArgument -- This imported type has IMPLICIT tags --,
                            -- 1994 extension --
        submission-options  [4]  MSSubmissionOptions OPTIONAL }

forwarding-request EXTENSION ::= {
        SequenceNumber,
        IDENTIFIED BY standard-extension:36 }

MSMessageSubmissionResult ::= CHOICE {
        mts-result SET {
            COMPONENTS OF   MessageSubmissionResult -- This imported type has IMPLICIT tags --,
                            -- 1994 extension --
            ms-message-result   [4] CommonSubmissionResults OPTIONAL },
                            -- 1994 extension --
        store-draft-result  [4] CommonSubmissionResults }

--

ms-probe-submission ABSTRACT-OPERATION ::= {
        ARGUMENT        MSProbeSubmissionArgument
        RESULT          MSProbeSubmissionResult
        ERRORS          {submission-control-violated | element-of-service-not-subscribed | originator-invalid |
                        recipient-improperly-specified | inconsistent-request | security-error |
                        unsupported-critical-function | remote-bind-error,
                        ... -- 1994 extension additions --,
                        ms-extension-error | message-group-error | entry-class-error | service-error}
        CODE            op-ms-probe-submission }

MSProbeSubmissionArgument ::= SET {
        COMPONENTS OF       ProbeSubmissionArgument -- This imported type has IMPLICIT tags --,
                            -- 1994 extension --
        submission-options  [4]  MSSubmissionOptions OPTIONAL }

MSProbeSubmissionResult ::= SET {
        COMPONENTS OF       ProbeSubmissionResult -- This imported type has IMPLICIT tags --,
                            -- 1994 extension --
        ms-probe-result     [4]  CommonSubmissionResults OPTIONAL }

ms-cancel-deferred-delivery ABSTRACT-OPERATION ::= cancel-deferred-delivery

ms-submission-control ABSTRACT-OPERATION ::= submission-control

-- Abstract-errors

attribute-error ABSTRACT-ERROR ::= {
        PARAMETER SET {
            problems    [0]  SET SIZE (1.. ub-per-entry) OF SET {
                problem     [0]  AttributeProblem,
                type        [1]  ATTRIBUTE.&id ({AttributeTable}),
                value       [2]  ATTRIBUTE.&Type ({AttributeTable} {@.type}) OPTIONAL } }
        CODE        err-attribute-error }
```