

INTERNATIONAL STANDARD

ISO/IEC
10021-5

First edition
1990-12-01

Information technology — Text Communication — Message-Oriented Text Interchange Systems (MOTIS) —

Part 5: Message Store: Abstract Service Definition

*Technologies de l'information — Communication de texte — Systèmes d'échange
de texte en mode message —*

Partie 5: Dépôt de message: Définition de service abstrait



Reference number
ISO/IEC 10021-5 : 1990 (E)

Contents

	Page
Foreword	iv
Introduction	v
Section one - General	
1 Scope	1
2 Normative references	2
3 Definitions	4
4 Abbreviations	9
5 Conventions	10
Section two - Message Store abstract-service definition	
6 Message Store model	13
7 Abstract-bind and Abstract-unbind-operations	23
8 Abstract-operations	27
9 Abstract-errors	42
Section three - General-attribute-types and general-auto-action-types	
10 Overview	47
11 General-attribute-types	47
12 General-auto-action-types	64
Section four - Procedures for Message Store and ports realization	
13 Overview	68
14 Consumption of the Message Transfer System abstract-service	68
15 Supply of the Message Store abstract-service	74
16 Ports realization	81

ANNEXES

A	Formal assignment of object identifiers	82
B	Formal definition of the Message Store abstract-service	84
C	Formal definition of general-attribute-types	93
D	Formal definition of general-auto-action-types	99
E	Formal definition of MS parameter upper bounds	101
F	Example of the Summarize abstract-operation	102
G	Differences between the CCITT Recommendation X.413 text and ISO/IEC 10021-5 text	104

IECNORM.COM : Click to view the full PDF of ISO/IEC 10021-5:1990

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*.

ISO/IEC 10021-5 consists of the following parts, under the general title: *Information technology — Text Communication — Message-Oriented Text Interchange Systems (MOTIS)* —

- *Part 1: System and Service Overview*
- *Part 2: Overall Architecture*
- *Part 3: Abstract Service Definition Conventions*
- *Part 4: Message Transfer System: Abstract Service Definition and Procedures*
- *Part 5: Message Store: Abstract Service Definition*
- *Part 6: Protocol Specifications*
- *Part 7: Interpersonal Messaging System*

Annexes A, B, C and D form an integral part of this part of ISO/IEC 10021. Annexes E, F and G are for information only.

Introduction

This part of ISO/IEC 10021 is one of a number of parts of ISO/IEC 10021 (the International Standard for Message Oriented Text Interchange System (MOTIS)).

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

This part of ISO/IEC 10021 defines the Message Store abstract-service (MS abstract-service) which supports message-retrieval from a Message Store (MS) and indirect-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.

This International Standard has been produced by joint CCITT-ISO/IEC agreement. The corresponding CCITT Recommendation is X.413 (1988). Annex G list the differences between the two documents.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10021-5:1990

IECNORM.COM : Click to view the full PDF of ISO/IEC 10021-5:1990

Information technology — Text Communication — Message-Oriented Text Interchange Systems (MOTIS) —

Part 5: Message Store: Abstract Service Definition

Section one - General

1 Scope

This part of ISO/IEC 10021 defines the Message Store abstract-service. This abstract-service is provided by the Message Store access-protocol (specified in ISO/IEC 10021-6) in conjunction with the MTS abstract-service (defined in ISO/IEC 10021-4), together with the Remote Operations Service Element (ROSE) services (defined in ISO/IEC 9072-1). The abstract-syntax-notation for the application-layer protocols used in this part of ISO/IEC 10021 is defined in ISO 8824.

Other parts of ISO/IEC 10021 define other aspects of the MHS. ISO/IEC 10021-1 defines the user-oriented services provided by the MHS. ISO/IEC 10021-2 provides an architectural overview of the MHS. ISO/IEC 10021-3 provides a description of the abstract-service definition conventions used in MHS. ISO/IEC 10021-7 defines the abstract-service for interpersonal messaging and defines the format of interpersonal-messages.

Section two of this part of ISO/IEC 10021 contains the Message Store abstract-service definition. Clause 6 describes the MS model. Clause 7 specifies the abstract-syntax-notation for the abstract-bind and the abstract-unbind-operations. Clause 8 specifies the abstract-syntax-notation for the operations of the abstract-service. Clause 9 specifies the abstract-syntax-notation for the errors of the abstract-service.

Section three of this part of ISO/IEC 10021 defines the general-attribute-types and general-auto-action-types related to the MS. Clause 10 contains an overview. Clause 11 specifies the abstract-syntax-notation for the general-attribute-types. Clause 12 specifies the abstract-syntax-notation for the general-auto-action-types.

Section four of this part of ISO/IEC 10021 describes the procedures for Message Store and the ports realization. Clause 13 contains an overview. Clause 14 describes how the Message Store abstract-service is supplied. Clause 15 describes how the Message Transfer System abstract-service is consumed. Clause 16 describes how the MS ports are realized.

No requirement is made for conformance to this part of ISO/IEC 10021.

2 Normative references

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

2.1 Reference Model references

This part of ISO/IEC 10021 cites the following reference model related documents

ISO 7498:1984, *Information processing systems - Open Systems Interconnection - Basic Reference Model.*

2.2 Presentation references

This part of ISO/IEC 10021 cites the following presentation related documents

ISO 8824:1987, *Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).*

ISO 8824:1987/Add.1: ¹⁾ *Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1) - Addendum 1: ASN.1 Extensions.*

2.3 Remote Operations references

This part of ISO/IEC 10021 cites the following remote operations related document

ISO/IEC 9072-1:1989, *Information processing systems - Text Communication - Remote Operations Part 1: Model, Notation and Service Definition.*

2.4 Directory References

This part of ISO/IEC 10021 cites the following directory related documents

ISO/IEC 9594-1: ¹⁾ *Information processing systems - The Directory - Overview of Concepts, Models and Services.*

ISO/IEC 9594-2: ¹⁾ *Information processing systems - The Directory - Models.*

ISO/IEC 9594-3: ¹⁾ *Information processing systems - The Directory - Abstract Service Definition.*

ISO/IEC 9594-4: ¹⁾ *Information processing systems - The Directory - Procedures for Distributed Operation.*

ISO/IEC 9594-5: ¹⁾ *Information processing systems - The Directory - Protocols Specifications.*

ISO/IEC 9594-6: ¹⁾ *Information processing systems - The Directory - Selected Attribute Types.*

¹⁾ To be published

- ISO/IEC 9594-7: 1) *Information processing systems - The Directory - Selected Object Classes.*
 ISO/IEC 9594-8: 1) *Information processing systems - The Directory - Authentication Framework.*

2.5 Message Handling references

This International Standard cites the following ISO/IEC message handling documents

- ISO/IEC 10021-1:1989 *Information processing systems - Text Communication - Message Oriented Text Interchange System (MOTIS) - Part 1: System and Service Overview.*
- ISO/IEC 10021-2:1989 *Information processing systems - Text Communication - Message Oriented Text Interchange System (MOTIS) - Part 2: Overall Architecture.*
- ISO/IEC 10021-3:1989 *Information processing systems - Text Communication - Message Oriented Text Interchange System (MOTIS) - Part 3: Abstract Service Definition Conventions.*
- ISO/IEC 10021-4:1989 *Information processing systems - Text Communication - Message Oriented Text Interchange System (MOTIS) - Part 4: Message Transfer System - Abstract Service Definition and Procedures.*
- ISO/IEC 10021-6:1989 *Information processing systems - Text Communication - Message Oriented Text Interchange System (MOTIS) - Part 6: Protocol Specifications.*
- ISO/IEC 10021-7:1989 *Information processing systems - Text Communication - Message Oriented Text Interchange System (MOTIS) - Part 7: Interpersonal Messaging System.*

1) To be published

3 Definitions

3.1 Common Definitions for MHS

For a list of the common definitions for MHS refer to ISO/IEC 10021-2.

3.2 Message Store Definitions

For the purpose of this part of ISO/IEC 10021 the following definitions apply

- 3.2.1 abstract-association:** An abstract binding between two communication partners, in this part of ISO/IEC 10021 the binding between a UA and an MS for the provision of the MS abstract-service, or between an MS and an MTA for the provision of the MTS abstract-service.
- 3.2.2 abstract-bind-parameters:** Parameters defined in this part of ISO/IEC 10021 which are contained in the abstract-bind operation.
- 3.2.3 abstract-unbind-parameters:** Parameters defined in this part of ISO/IEC 10021 which are contained in the abstract-unbind operation.
- 3.2.4 Administration Port:** The port offering the administration (for MTS) set of abstract-services within the MS abstract-service.
- 3.2.5 Alert abstract-operation:** An abstract-operation which allows the MS to signal, based on selection criteria, to the UA that messages or reports are waiting in the MS.
Can only be issued on an existing abstract-association.
- 3.2.6 attribute:** The information of a particular type appearing in an entry in an information-base.
- 3.2.7 attribute-type:** That component of an attribute which indicates the class of information given by that attribute.
- 3.2.8 attribute-value:** A particular instance of that class of information indicated by an attribute type.
- 3.2.9 attribute-value-assertion:** A proposition, which may be true, false, or undefined, concerning the values of attributes in an entry.
- 3.2.10 auto-action:** Actions, that can be performed automatically by the MS, based on previously registered information from the MS-owner via the UA.
- 3.2.11 auto-action-type:** An auto-action-type is used to indicate the type of auto-action, e.g. Alert.
- 3.2.12 auto-alert:** Auto-alert is the auto-action within the MS, which triggers an Alert abstract-operation or another action by the MS.
- 3.2.13 auto-forward:** Auto-forward is the auto-action within the MS, which triggers a message to be auto-forwarded to another recipient (or other recipients) by the MS.

- 3.2.14 child-entry:** An entry, other than the main-entry in an information-base. The parent-entry for a child-entry can be either the main-entry or another child-entry, depending on the number of entry levels in each case.
- 3.2.15 child-sequence-number:** A sequence-number in a parent-entry pointing to a child-entry. A parent-entry can have more than one child-sequence-number value, depending on the number of child-entries.
- 3.2.16 conditional (C) component:** An ASN.1 element which shall be present in an instance of its class as dictated by this International Standard.
See **grade**.
- 3.2.17 content-length:** An attribute which gives the length of the content of a delivered-message (or returned-content).
- 3.2.18 content-returned:** An attribute which signals that a delivered-report (or a delivered-message) contained a returned content.
- 3.2.19 converted EITs:** An attribute identifying the encoded-information-types of the message content after conversion.
- 3.2.20 creation-time:** An attribute which gives the creation-time (by the MS) of an entry.
- 3.2.21 Delete abstract-operation:** An abstract-operation used to delete one or more entries from an information-base.
- 3.2.22 delivered-EITs:** A multi-valued attribute, giving information about EITs in a delivered-message.
- 3.2.23 delivered-message entry:** An entry in the stored-messages information-base resulting from a delivered-message.
- 3.2.24 delivered-report entry:** An entry in the stored-messages information-base resulting from a delivered-report.
- 3.2.25 entry:** An information set in an information-base.
See main-entry, parent-entry and child-entry for further classification of entries.
- 3.2.26 entry-information:** A parameter, used in abstract-operations, which conveys selected information from an entry.
- 3.2.27 entry-information-selection:** A parameter, used in abstract-operations, which indicates what information from an entry is being requested.
- 3.2.28 entry-status:** An attribute giving information about the processing status of that entry. Possible values are new, listed or processed.
- 3.2.29 entry-type:** An attribute which signals if an entry is associated with a delivered-message or a delivered-report.
- 3.2.30 Fetch abstract-operation:** An abstract-operation which allows one entry to be fetched from the stored-messages information-base.

- 3.2.31 fetch-restrictions:** Restrictions, imposed by the UA, on what kind of messages it is prepared to receive as a result of fetch. The possible restrictions are on message-length, content-types and EITs.
- 3.2.32 filter:** A parameter, used in abstract-operations, to test a particular entry in an information-base and is either satisfied or not by that entry.
- 3.2.33 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 or false.
- 3.2.34 forwarding-request:** This is a parameter that may be present in a Message-submission abstract-operation, invoked by the UA, to request that a message is forwarded from the MS.
- 3.2.35 general-attribute:** A set of MS attributes which are valid for all types of messages and reports, independent of content-type.
Only these MS attributes are explicitly defined in this part of ISO/IEC 10021.
- 3.2.36 general-auto-action:** Auto-actions which are valid for all types of messages and reports, independent of content-type.
Only these auto-actions are explicitly defined in this part of ISO/IEC 10021.
- 3.2.37 grade:** Defined in ISO/IEC 10021-2.
- 3.2.38 Indirect-submission Port:** The port offering the Indirect-submission abstract-service within the MS abstract-service.
The Indirect-submission abstract-service offers the same services as the Message-submission abstract-service (from the MTS abstract-service) with the added functionality of forwarding messages residing in the MS.
- 3.2.39 information-base:** Objects within the MS which store information relevant to the MS abstract-service, e.g. the stored-messages information-base, which stores the messages and reports that have been delivered into the MS.
- 3.2.40 information-base-type:** The type of information-base, e.g. the stored-messages.
- 3.2.41 limit:** A component in the selector parameter which identifies the maximum number of selected entries to be returned in the result of an abstract-operation.
- 3.2.42 list abstract-operation:** An abstract-operation which allows a selection of entries from an information-base and requested attribute information to be returned for those entries.
- 3.2.43 listed:** An entry-status value.
- 3.2.44 macro:** See ISO 8824.
- 3.2.45 main-entry:** For each successful abstract-operation which creates information-base entries, there is always one main-entry. Further, or more detailed, information resulting from the same abstract-operation can be stored in child-entries.
- 3.2.46 mandatory (M) component:** An ASN.1 element which shall always be present in an instance of its class.
See **grade**.

- 3.2.47 matching:** The process of comparing the value supplied in an attribute-value-assertion with the value of the indicated attribute-type stored in the MS or deciding whether the indicated attribute-type is present.
- 3.2.48 Message Retrieval Service Element (MRSE):** The application-service-element by means of which a receiving UA effects retrieval of messages from an MS, or any of various related tasks.
- 3.2.49 MS:** Message Store, also used as a shorter form for "MS abstract-service-provider".
- 3.2.50 MS abstract-service:** The set of capabilities that the MS offers to its users by means of its ports.
- 3.2.51 MS abstract-service-user:** The user of the MS abstract-service.
This is the UA.
- 3.2.52 MS abstract-service-provider:** The MS which provides the MS abstract-service.
- 3.2.53 MS-user:** A shorter form for "MS abstract-service-user".
- 3.2.54 Message-submission abstract-operation:** An abstract-operation which allows the UA to submit a message to the MTS via the MS, and/or to forward a message from the MS to the MTS.
- 3.2.55 multi-valued attribute:** An attribute which can have several values associated with it.
- 3.2.56 new:** An entry-status value.
- 3.2.57 optional (O) component:** An ASN.1 element which shall be present in an instance of its class at the discretion of the object (e.g. user) supplying that instance.
See **grade**.
- 3.2.58 original-EITs:** An attribute identifying the original encoded-information-types of the message content.
- 3.2.59 override:** A component of the selector parameter indicating that the previously registered restrictions for this abstract-operation should not apply for this instance of this abstract-operation.
- 3.2.60 parent-entry:** A parent-entry has one or more child-entries, which were created as a result of the same abstract-operation.
If a parent-entry is not a child-entry of another parent-entry, it is a main-entry.
- 3.2.61 parent-sequence-number:** A sequence-number in a child-entry pointing to its parent-entry.
There can only be one parent-sequence-number in a child-entry.
- 3.2.62 partial-attribute-request:** A component of the entry-information-selection which enables the return of only selected values of a multi-valued attribute.
- 3.2.63 position:** Positions are parameters used to specify a bound of a range.
- 3.2.64 processed:** An entry-status value.
- 3.2.65 range:** A parameter, used in abstract-operations, to select a contiguous sequence of entries from an information-base.

- 3.2.66 Register-MS abstract-operation:** An abstract-operation which allows the UA to register certain information, relevant to the UA-MS interworking, in the MS .
- 3.2.67 registration:** Information which is registered in the MS and stored (until changed by the Register-MS abstract-operation) between abstract-associations.
See Register-MS abstract-operation.
- 3.2.68 registration-identifier:** An identifier for one particular set of registration-parameters for an auto-action-type.
- 3.2.69 Retrieval Port:** The port offering the retrieval set of abstract-services within the MS abstract-service.
- 3.2.70 returned-content entry:** An entry-type in the stored-messages information-base which contains the returned content from a previously submitted message.
- 3.2.71 selector:** A parameter, used in abstract-operations, to select entries from an information-base.
- 3.2.72 sequence-number:** An attribute which uniquely identifies an entry.
Sequence-numbers are allocated in ascending order.
- 3.2.73 single-valued attribute:** An attribute which can only have one value associated with it.
- 3.2.74 span:** A component in the Summarize abstract-operation result containing the lowest and highest sequence-numbers of the entries that matched the selection criteria.
- 3.2.75 stored-messages:** The most important information-base in this part of ISO/IEC 10021, used to store entries containing messages and reports delivered by the MTS to the MS.
- 3.2.76 subscription:** A long-term agreement between the MS supplier or administrator and the MS customers (MS-owners) on the availability and use of optional MS features such as optional services and attributes.
This part of ISO/IEC 10021, assumes that such a mechanism is provided, but does not prescribe or offer any standardized method for how to provide this.
- 3.2.77 substring:** A filter-item used to specify a string of characters which appear (in the same given order) in a value of an attribute.
- 3.2.78 Summarize abstract-operation:** An abstract-operation which allows a quick overview of the kind and number of entries which are currently stored in an information-base.
- 3.2.79 synopsis:** A content specific attribute that may be used to show how child-entries, containing parts of the content, are related to each other and the main-entry.
The attribute has to be specified in the International Standard, which describes the content-type, e.g. see IPM-synopsis defined in ISO/IEC 10021-7.

4 Abbreviations

ASN.1:	Abstract Syntax Notation One
C:	conditional
DL:	distribution-list
EIT:	encoded-information-type(s)
M:	mandatory
M:	multi-valued
MASE:	Message Administration Service Element
MDSE:	Message Delivery Service Element
MH:	Message Handling
MHS:	Message Handling System
MOTIS:	Message Oriented Text Interchange System
MRSE:	Message Retrieval Service Element
MS:	Message Store
MT:	Message Transfer
MTS:	Message Transfer System
N:	no
O:	optional
O/R:	originator/recipient
P:	present
ROSE:	Remote Operations Service Element
S:	single-valued
UA:	User Agent
UTC:	Universal Co-ordinated Time
Y:	yes

5 Conventions

This part of ISO/IEC 10021 uses the description conventions listed in the following four clauses.

5.1 Conventions for abstract-services

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

- 1) ASN.1 itself, to specify the abstract-syntax of information-bases and their components, and common data-types.
- 2) The ASN.1 PORT macro and associated abstract-service definition conventions of ISO/IEC 10021-3, to specify the Retrieval Port.
- 3) The ASN.1 ABSTRACT-BIND, ABSTRACT-UNBIND, ABSTRACT-OPERATION, and ABSTRACT-ERROR macros and associated abstract-service definition conventions of ISO/IEC 10021-3, to specify the MS abstract-service.

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

- 1) **Mandatory (M):** A mandatory component shall be present in every instance of the class.
- 2) **Optional (O):** An optional component shall be present in an instance of the class at the discretion of the object (e.g user) supplying that instance.
- 3) **Conditional (C):** A conditional component shall be present in an instance of the class as dictated by this part of ISO/IEC 10021.

5.2 Conventions for attribute-types used in table 1 of clause 11

This part of ISO/IEC 10021 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 can occur

S	single-valued
M	multi-valued.

For the column headed "Support level by the MS and access UA" the following values can occur

M	mandatory
O	optional.

For the columns headed "Presence in delivered message entry", "Presence in delivered report entry", and "Presence in returned message entry", the presence of each attribute-type is described by one of the following values

P	<u>always present</u> 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 would be present because
- it is supported by the MS and subscribed to by the user and;
 - it was present in an optional parameter in the relevant abstract-operation
- always absent, otherwise.

For the columns headed "Available for List, Alert" and "Available for Summarize", the following values can occur

N	no
Y	yes.

5.3 Conventions for attribute-types used in table 2 of clause 11

This part of ISO/IEC 10021 uses the conventions listed below in its definition of the attribute-types for the MS abstract-service. Clause 11 includes the table that lists the attribute-types.

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

S	single-valued
M	multi-valued.

For the column headed "Source generated by" the following values can occur

MD	MessageDelivery abstract-operation
MS	Message Store
RD	ReportDelivery abstract-operation.

5.4 Font conventions for text in general

Throughout this part of ISO/IEC 10021, terms are rendered in **bold** when defined, without emphasis upon all other occasions. 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 part of ISO/IEC 10021, ASN.1 definitions are written in a different (**bold**) font than the rest of the document in order to highlight the difference between normal text and ASN.1 definitions. The font used for ASN.1 definitions is also one size smaller than the ordinary text. When ASN.1 protocol elements and elements values are described in accompanying text, their names are rendered in **bold**.

5.6 Rules for ASN.1 definitions

ASN.1 definitions appears both in the body of the document 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.

Section two - Message Store abstract-service definition

6 Message Store model

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

The MS serves an intermediary role between the UA and the MTS. Its primary function is to accept delivery of messages on behalf of a single MHS end-user, and to retain them for subsequent retrieval by the end-user's UA. The MS also provides indirect message-submission and message-administration services to the UA, in effect, via "pass-through" to the MTS. This enables the MS to provide additional functionality compared to submission directly to the MTA; such as forwarding of messages residing in the MS and logging facilities.

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

The MS is described using an abstract model in order to define the services provided by the MS - the Message Store abstract-service. Figure 1 shows the MS abstract-service in relation to its 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.

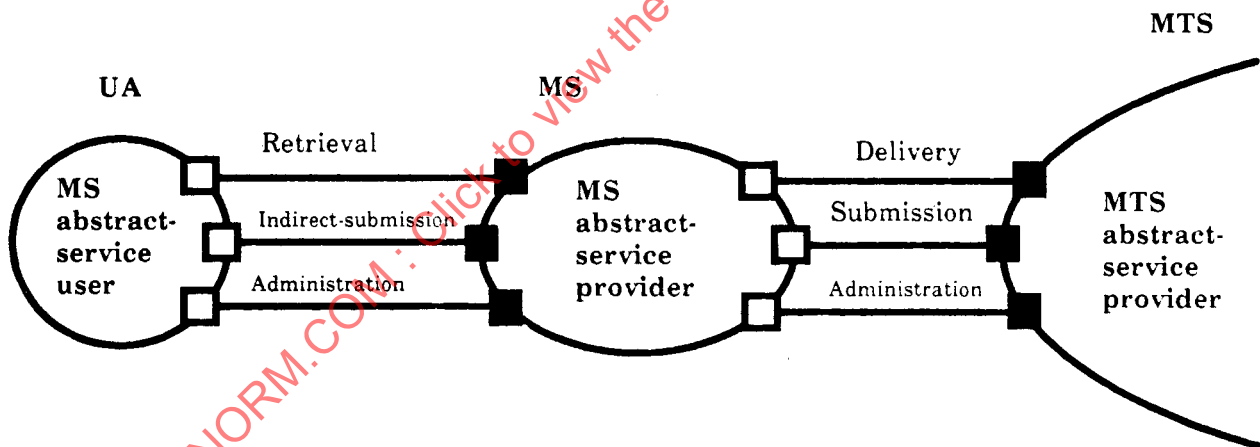


Figure 1 - Message Store abstract-service

For an introduction and description of the abstract-service concept and its definition conventions, see ISO/IEC 10021-3.

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) to the UA.

6.1 Message Store object

The MS is modeled as an atomic object. It supplies the MS Retrieval Port abstract-services to the MS abstract-service-user. Acting as a "surrogate" MTS abstract-service-provider, the MS also supplies the MTS submission and administration abstract-services to the MS abstract-service-user (MS-user), and acting as a UA "surrogate", it consumes the MTS Delivery Port, Submission Port, and Administration Port abstract-services in its role as MTS abstract-service user.

The formal definition for the Message Store object is as follows:

```

mS OBJECT
  PORTS { retrieval[S],
           indirectSubmission[S],
           administration[S],
           delivery[C],
           submission[C],
           administration[C]}
  ::= id-ot-ms
  
```

The MS-user is also modeled as an object. It consumes the MS Retrieval Port and Indirect-submission Port abstract-services and the Administration Port abstract-services provided transparently by the MS.

```

msUser OBJECT
  PORTS { retrieval[C],
           indirectSubmission[C],
           administration[C]}
  ::= id-ot-ms-user
  
```

6.2 Message Store Ports

An MS provides the Retrieval, Indirect-submission, and Administration Ports to the MS abstract-service user. The collection of capabilities provided by these ports provides the MS abstract-service. The retrieval capabilities are unique to the MS. These capabilities include obtaining information on, fetching (in whole or in part), and deleting messages residing in the MS. Additional capabilities are provided for registering certain MS provided automatic actions (i.e. auto-forwarding and alert).

Additional message management services, are planned as future extensions to this part of ISO/IEC 10021. They will be performed by the MS on the UA's behalf, for logging incoming and outgoing messages, and for auto-correlating incoming notifications with logging information about outgoing messages. These extensions would form an Addendum to ISO/IEC 10021.

In order to provide the services described in 6.1 to an MS-user, the MS interacts, on behalf of the MS-user, with the MTS abstract-service, and acts as a consumer of the MTS Delivery, Submission and Administration Ports. The abstract-services provided by the MTS ports are defined in clause 8 of ISO/IEC 10021-4.

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

With the exception of the Retrieval Port provided Alert service and the Indirect-submission Port provided Submission-control service, 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 is discretionary.

If security-contexts are established between the UA and the MS, and between the MS and the MTA, 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 Port

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

```
retrieval PORT
  CONSUMER INVOKES {
    Summarize,
    List,
    Fetch,
    Delete,
    Register-MS}
  SUPPLIER INVOKES {
    Alert}
  ::= id-pt-retrieval
```

The details of the **Retrieval Port** abstract-services are described in clauses 7 to 9.

6.2.2 Indirect-submission Port

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

```
indirectSubmission PORT ::= submission
```

The **Indirect-submission Port** makes use of the Submission Port abstract-services defined in 8.2 of ISO/IEC 10021-4.

6.2.3 Administration Port

The **Administration Port** is defined in 8.4 of ISO/IEC 10021-4.

The MS shall have no interaction with the change-credentials abstract-service. If the MS-user needs to have its credentials updated, then the Register-MS abstract-operation is used. See 8.6.

6.3 Information model

This subclause describes the information model used by the MS. It models **information-bases**, which consist of **entries**, which consist of **attributes**.

6.3.1 Information-bases

The MS stores and maintains **information-bases**. An **information-base** in the MS is a "data-base" containing all the **entries** representing constituent objects of a particular category or categories.

This part of ISO/IEC 10021 defines and describes the **stored-messages information-base**. This holds information derived from message-deliveries and report-deliveries to the MS across the MTS Delivery Port, and is described in 6.4. Other information-bases are the inlog and the outlog, which are planned as a future addendum to this part of ISO/IEC 10021. These extensions are outside the scope of the corresponding CCITT Recommendation.

```
InformationBase ::= INTEGER {
    stored-messages (0),
    inlog           (1),
    outlog          (2)} (0..ub-information-bases)
```

6.3.2 Entries

Each **information-base** is organized as a sequence of **entries**. An **entry** represents a single object (such as a delivered message) within the **information-base**.

Each entry is identified by means of its **sequence-number**, unique within an **information-base**, and generated by the MS as new entries are created. Within an **information-base**, the MS generates the **sequence-numbers** in ascending order without cycling, and they are never re-used.

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

NOTE - For example, the MS may choose to allocate **sequence-numbers** by using the time to a sufficient granularity to ensure uniqueness.

6.3.3 Attributes

6.3.3.1 Introduction

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

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**.

An **attribute** consists of an **attribute-type**, which identifies the class of information given by an **attribute**, and the corresponding **attribute-value(s)**, which are particular instances of that class appearing in the **entry**.

```
Attribute ::= SEQUENCE {
    type      AttributeType,
    values    SEQUENCE SIZE (1..ub-attribute-values) OF ANY --DEFINED BY type--}
```

NOTE - Thus, for example, in a delivered-message-entry (described in 6.4) the **attribute-type** could be the message's **priority**, and a corresponding **attribute-value** could be **urgent**.

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

For some **attribute-types**, an **attribute** shall only contain 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 6.3.3.2).

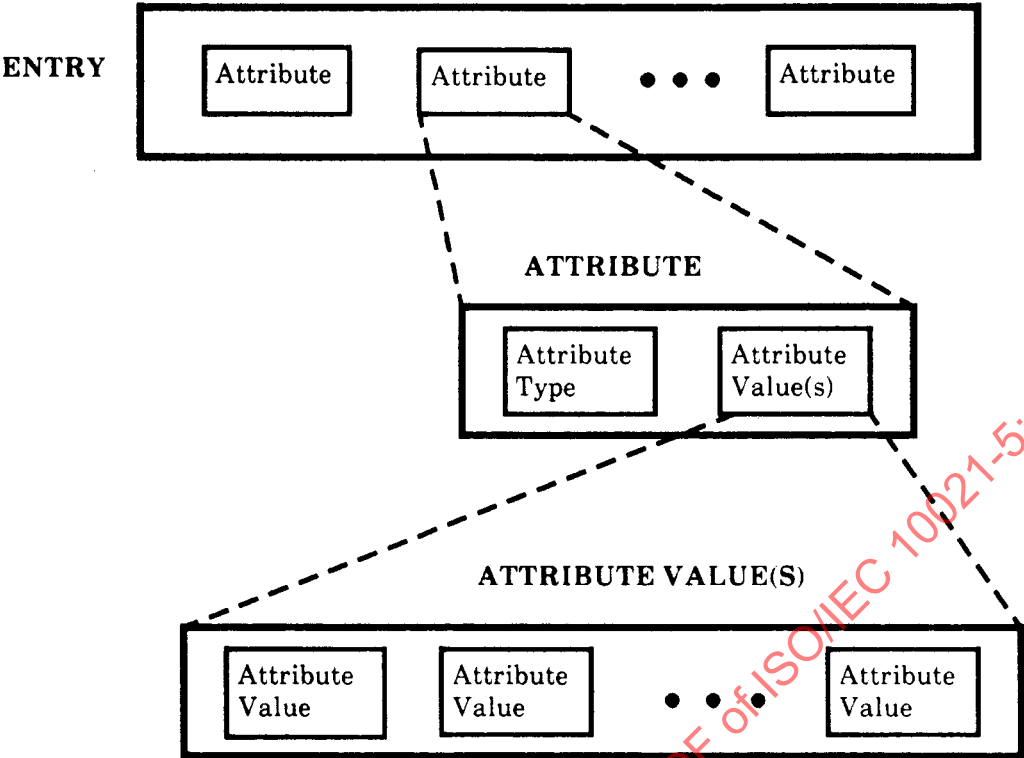


Figure 2 - The Components of an entry

NOTE - Thus, for example, the **attribute-type** for the **originator-name** attribute (described in 11.2.28) is **single-valued**, whereas that for **other-recipient-names** (described in 11.2.29) is **multi-valued**.

6.3.3.2 Attribute-type

Some **attribute-types** will be internationally standardized. Other **attribute-types** will 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** for the stored-messages information-base are defined in clause 11. Such **attribute-types** are known as **general-attribute-types** and attributes of these types as **general-attributes**.

6.3.3.3 Attribute-values

Defining an **attribute-type** also involves specifying the ASN.1 data-type to which every value in such attributes shall conform. The data-type of an **attribute-value** for the **attribute-type** is defined through the object identifier for the **attribute-type**.

6.3.3.4 Attribute-type definition and the ATTRIBUTE Macro

The definition of an **attribute-type** involves

- a) assigning an **object identifier** to the **attribute-type**;
- b) indicating the ASN.1 data-type of an **attribute-value**;
- c) indicating whether an attribute of this **attribute-type** may have more than one value;
- d) indicating whether an attribute of this **attribute-type** may be used for filtering using equality, substrings, and/or ordering relations, see 8.1.2.

NOTE - A filter may always test for the presence or absence in an entry of an attribute of a particular **attribute-type**.

The following ASN.1 macro is used to define an **attribute-type**. The formal definition of this macro is given in ISO/IEC 9594-2 and is documented here as an aid to the reader.

```

ATTRIBUTE MACRO      ::=
BEGIN
  TYPE NOTATION      ::=  AttributeSyntax Multivalued | empty
  VALUE NOTATION     ::=  value (VALUE OBJECT IDENTIFIER)

  AttributeSyntax    ::=  "WITH ATTRIBUTE-SYNTAX" SyntaxChoice
  SyntaxChoice       ::=  value (ATTRIBUTE-SYNTAX) Constraint | type MatchTypes

  Constraint         ::=  "(" ConstraintAlternative ")" | empty
  ConstraintAlternative ::=  StringConstraint | IntegerConstraint
  StringConstraint   ::=  "SIZE" "(" SizeConstraint ")" | empty
  SizeConstraint     ::=  SingleValue | Range
  SingleValue        ::=  value (INTEGER)
  Range              ::=  value (INTEGER) ".." value (INTEGER)
  IntegerConstraint  ::=  Range

  MatchTypes         ::=  "MATCHES FOR" Matches | empty
  Matches            ::=  Match Matches | Match
  Match              ::=  "EQUALITY" | "SUBSTRINGS" | "ORDERING"

  Multivalued        ::=  "SINGLE VALUE" | "MULTI VALUE" | empty
END

```

The correspondence between the parts of the definition, as listed above, and the various pieces of the notation introduced by the **ATTRIBUTE** macro, is as follows:

- a) **MACRO value**: The **object identifier** which is used to identify an attribute.
- b) **Attribute-syntax**: Notes which syntax-choice has been made.
- c) **Syntax-choice**: Notes whether the attribute is defined externally or internally. The syntax of all the attributes defined in this part of ISO/IEC 10021 is defined internally, which means using the choice type **MatchTypes**.
- d) **Multivalued**: Denotes whether the attribute is single or multi-valued.
- e) **Match-types**: Gives the data-type of the contents of the attribute, and describes whether the **attribute** can be matched ("MATCHES FOR") for equality ("EQUALITY"), for substrings ("SUBSTRINGS"), and for an ordering relation ("ORDERING").

Matching for this part of ISO/IEC 10021 is restricted as follows:

- i) **EQUALITY** is applicable to any attribute-syntax. The presented value shall conform to the data-type of the attribute-syntax;
- ii) **SUBSTRING** is applicable to any attribute-syntax with a **string** data type. The presented value shall be a sequence ("SEQUENCE OF"), each of whose elements conforms to the data-type, and
- iii) **ORDERING** is applicable to any attribute-syntax for which a rule can be defined that will allow a presented value to be described as less than equal to, or greater than a target value. The presented value shall conform to the data-type of the attribute-syntax.
MS uses this for the INTEGER and UTCTime data types. For UTCTime, the ordering is chronological, not alphabetical.

The remaining choices and parameters of the **ATTRIBUTE** macro are not used in this part of ISO/IEC 10021.

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

Although entries in a single information-base are generally independent of each other, the MS information model allows such entries to be related to one another. One entry, 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 11.2.30.
- 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 11.2.1.

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 a Delete abstract-operation (see 8.5) may only select **main-entries**, in which case the **main-entry** and all its children and children's children, etc, will also be deleted.

NOTE - This concept allows, for example, that body-parts of an Interpersonal Message which contain a forwarded message (for details see 19.1 of ISO/IEC 10021-7) to be represented by individual **child-entries**. The **content general-attribute** of the **main-entry** will comprise the complete **content**, so the data representing that message **body-part** is logically present in more than one entry.

6.4 Stored-messages

The **stored-messages information-base** acts as a repository for information obtained from the MessageDelivery and ReportDelivery abstract-operations of the Delivery Port. It contains entries for delivered messages (**delivered-message-entries**), of an open-ended number of content-types, and for reports (**delivered-report-entries**). An entry in the **stored-messages information-base** is created by the MS when a message or report is delivered to the MS. For more details of these entries and how they are generated, see clauses 11 and 15.

To draw information from the content of a message, the MS must know the content's syntax and semantics, as signaled via the content-type. In general, a particular instance of the MS has knowledge of

zero or more content-types. When an MS encounters a message of whose content-type it has insufficient knowledge, it is unable to generate any content-type-specific attributes in the message's entry.

A delivered-message or an arriving notification may result in a main-entry and one or more levels of child-entries. The one case defined by this part of ISO/IEC 10021 is when a non-delivery notification contains a returned-content (the **delivered-report entry** is the main-entry and the returned-content is its child-entry, known as a **returned-content entry**).

The rules for how a message-content may be split across several entries is specific to each content-type. A content-specific **synopsis-attribute** may be used to show how the main-entry and the corresponding child-entries are related. When such an attribute is defined, it appears in the International Standard which defines the content-type itself. The **synopsis-attribute** is constructed by the MS.

NOTE - For Interpersonal Messaging (ISO/IEC 10021-7), nested IP-messages within an IP-message are each represented by a child-entry. The ipm-synopsis attribute-type is an example of a content-specific **synopsis-attribute-type**.

An important property of an entry in the stored-messages is its **entry-status**. It is created and maintained by the MS. It can take the following values:

- a) **New** The message has neither been listed by a UA nor has it been automatically processed by the MS.
- b) **Listed** Information about the message has been returned to the UA in either a List abstract-operation or a Fetch abstract-operation, but the message has not yet been completely **processed**.
- c) **Processed** Either a UA has "completely fetched" the message, or the MS has performed some auto-action on it. (Note that some auto-actions result in the message being deleted). The exact definition of "completely fetched" is content-specific and is defined in the corresponding content-specific International Standard.

The **entry-status** of a (non-)delivery-notification becomes **processed** when the delivered-report-envelope is retrieved.

The definition for **entry-status** is as follows;

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

6.5 Auto-actions

6.5.1 Introduction

This subclause defines a framework for the automatic actions (**auto-actions**) which may be registered with the MS.

An **auto-action** is an action that will occur automatically whenever the associated registration criteria have been satisfied. The result of an action being invoked is visible externally to the MS. **Auto-actions** are registered in the MS using the Register-MS abstract-operation (see 8.6).

Each class of **auto-action** is identified by means of an **auto-action-type**. Associated with the registration of an **auto-action**, there is a corresponding **auto-action-registration-parameter**, which are the parameters needed by the MS to perform the registered **auto-action** automatically. The

registration of an **auto-action** requires the use of an **auto-action-registration-identifier** to identify the particular registration.

```
AutoActionRegistration ::= SEQUENCE(  
    type                AutoActionType,  
    registration-identifier [0] INTEGER (1..ub-per-auto-action) DEFAULT 1,  
    registration-parameter [1] ANY DEFINED BY type)
```

6.5.2 Auto-action-type

Some **auto-action-types** will be internationally standardized. Other **auto-action-types** will 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 **auto-action-types**. This is accomplished by identifying each **auto-action-type** with an object identifier when the **auto-action-type** is defined .

```
AutoActionType ::= OBJECT IDENTIFIER
```

Certain general-purpose **auto-action-types** are defined in clause 12. Such **auto-action-types** are known as **general-auto-action-types** and **auto-actions** of these types as **general-auto-actions**.

6.5.3 Auto-action-registration-parameter

Defining an **auto-action-type** also involves specifying the ASN.1 data-type to which the **auto-action-registration-parameter** shall conform. The data-type of an **registration-parameter** is defined through the object identifier for the **auto-action-type**.

6.5.4 Auto-action-type definition and the AUTO-ACTION Macro

The definition of an **auto-action-type** involves

- a) assigning an object identifier to the **auto-action-type**;
- b) indicating the ASN.1 data-type of the **auto-action-registration-parameter**;

The following ASN.1 macro may (but need not be) used to define an **auto-action-type**

```
AUTO-ACTION MACRO ::=   
BEGIN  
    TYPE NOTATION      ::= Registration  
    VALUE NOTATION     ::= value (VALUE OBJECT IDENTIFIER)  
  
    Registration       ::= "REGISTRATION PARAMETER IS" type  
END
```

The correspondence between the parts of the definition, as listed above, and the various pieces of the notation introduced by the **AUTO-ACTION** macro, is as follows:

- a) **Registration**: gives the data-type of the registration parameters associated with an auto-action.
- b) **Value**: the **object identifier** which is used to identify the auto-action.

NOTE - No support is provided in the macro for defining the interaction (if any) between different registrations of the same (or different) auto-actions.

6.6 Forwarding of messages

The MS-user makes use of the Message-submission abstract-operation and its parameters as defined in 8.2 of ISO/IEC 10021-4 to request that a message stored in the MS be explicitly forwarded to other users.

The **forwarding-request parameter** is defined using the **EXTENSION** macro defined in clause 9 of ISO/IEC 10021-4 as follows:

```
forwarding-request EXTENSION
  SequenceNumber
  CRITICAL FOR SUBMISSION
  ::= 36
```

If the **SequenceNumber** supplied does not match that of an entry in the **stored-messages** information-base, or matches an entry which is unsuitable for forwarding this is reported using the **Inconsistent-request** abstract-error of 8.2.2.7 of ISO/IEC 10021-4.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10021-5:1990

7 Abstract-bind and abstract-unbind operations

7.1 Abstract-bind-operation

The **MS-Bind abstract-bind-operation** binds the Indirect-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. MS-Bind is defined as follows:

```
MSBind ::= ABSTRACT-BIND
  TO {indirectSubmission[S], retrieval[S], administration[S]}
  BIND
    ARGUMENT MSBindArgument
    RESULT MSBindResult
    BIND-ERROR MSBindError
```

7.1.1 Abstract-bind-argument

The **abstract-bind-argument** parameters are used to identify, authenticate and set the security-context for an MS abstract-service-user. They also contain a set of restrictions for entries to be returned as result of a Fetch abstract-operation, and finally, a request to be informed of the auto-action-types, attribute-types and content-types supported by the MS.

The definition of these parameters is as follows:

```
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}
```

- 1) **Initiator-name** (C): This argument contains the name of the initiator of the association and is supplied by the initiator. This argument is defined further in 8.1.1.1.1.1 of ISO/IEC 10021-4.
- 2) **Initiator-credentials** (M): This parameter contains the **credentials** of the initiator of the association. It shall be generated by the initiator of the abstract-association.

The **initiator-credentials** may be used by the responder to authenticate the identity of the initiator (see ISO/IEC 9594-8).

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

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 further in 8.1.1.1.1.2 of 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 8.1.1.1.1.2 of ISO/IEC 10021-4.

- 3) **Security-context** (O): This parameter identifies the **security-context** that the initiator of the abstract-association proposes to operate at. It is generated by the initiator of the abstract-association. The **security-context** is defined further in 8.1.1.1.1.3 of 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 abstract-service-user and the MS abstract-service 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 abstract-service-user and by the **security-labels** associated with the MS.

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

- 4) **Fetch-restrictions** (O): This contains the restrictions on entries to be returned as result of a Fetch abstract-operation. The **fetch-restrictions** remain set until an abstract-unbind-operation is issued.

In the absence of this argument, the default is that no fetch-restrictions need to be performed.

This argument consists of the following components:

```
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-content-length    [2] ContentLength OPTIONAL --default is no restriction--}
```

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

In the absence of this component, the default is that no fetch-restrictions on content-types need to be performed.

- b) **Allowed-EITs** (C): The encoded-information-types that the MS abstract-service-user is prepared to accept as result of a Fetch abstract-operation. If a message contains encoded-information-types other than the ones 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, an error will be reported. No filtering will take place if the Fetch abstract-operation has explicitly overridden the restriction.

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

```
MS-EIT :: = OBJECT IDENTIFIER
```

In the absence of this component, the default is that no fetch-restrictions on encoded-information-types need to be performed.

- c) **Maximum-content-length** (C): The maximum content length that the MS abstract-service-user is prepared to accept as result of a Fetch abstract-operation. Any message with a **content-length** exceeding the one specified will not be returned, but result in an error, unless the Fetch abstract-operation has explicitly overridden the restriction.

In the absence of this component, the default is that no fetch-restrictions on content-length need to be performed.

- 5) **MS-configuration-request (C)**: The **MS-configuration-request** is requested to obtain information relating to which auto-actions and optional attributes the MS provides support for.

In the absence of this component, the default is false which indicates that no such request is being made.

7.1.2 Abstract-bind-result

The **abstract-bind-result** parameters are as follows:

MSBindResult ::= SET {	
responder-credentials	[2] ResponderCredentials,
available-auto-actions	[3] SET SIZE(1 .. ub-auto-actions) OF AutoActionType OPTIONAL,
available-attribute-types	[4] SET SIZE(1 .. ub-attributes-supported) OF AttributeType OPTIONAL,
alert-indication	[5] BOOLEAN DEFAULT FALSE,
content-types-supported	[6] SET SIZE (1..ub-content-types) OF OBJECT IDENTIFIER OPTIONAL]

- 1) **Responder-credentials (M)**: This parameter contains the credentials of the responder of the abstract-association. It shall be generated by the responder of the abstract-association.

The **responder-credentials** may be used by the initiator to authenticate the identity of the responder (see ISO/IEC 9594-8).

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

If **strong-authentication** is used, the **responder-credentials** comprise a **responder-bind-token**, and, optionally, a **responder-certificate**, both of which are generated by the responder of the abstract-association. The **responder-bind-token** and **responder-certificate** are defined further in 8.1.1.1.2.2 of ISO/IEC 10021-4.

- 2) **Available-auto-actions (C)**: Specifies the set of all possible auto-actions that are supported by the MS (not just those requested by the MS abstract-service-user). Only present if an **MS-configuration-request** is made.
- 3) **Available-attribute-types (C)**: Specifies the set of all optional attributes supported by the MS. Only present if an **MS-configuration-request** is made.
- 4) **Alert-indication (C)**: If true, then an alert condition has occurred since the last successful **Alert-indication**.
- 5) **Content-types-supported (C)**: Specifies a set of object-identifiers that define the content-types that the MS has knowledge of. Only present if an **MS-configuration-request** is made.

7.1.3 Abstract-bind-errors

There are two possible errors defined by the **Retrieval Port**, namely **authentication-error** and **unacceptable-security-context**.

The definition of the errors is

```

MSBindError :: = ENUMERATED {
    authentication-error      (0),
    unacceptable-security-context (1),
    unable-to-establish-association (2)}

```

- 1) **Authentication-error** (C): This error reports that an abstract-association cannot be established because the initiator's **credentials** are not acceptable or are improperly specified.

The **authentication-error** has no parameters.

- 2) **Unacceptable-security-context** (C): This error reports that the **security-context** proposed by the initiator of the abstract-association is unacceptable to the responder.

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

- 3) **Unable-to-establish-association** (C): This error reports that the responder has rejected the initiator's attempt to establish an abstract-association.

The **unable-to-establish-association** error has no parameters.

7.2 Abstract-unbind-operation

The **MS-Unbind** abstract-unbind-operation closes the abstract-association. The issuing of an abstract-unbind-operation results in the relaxation of any fetch-restrictions that were specified in the abstract-bind operation argument. There is not argument, result, or error associated with the abstract-unbind-operation.

```

MSUnbind :: = ABSTRACT-UNBIND
    FROM {indirectSubmission[S], retrieval[S], administration[S]}

```

8 Abstract-operations

This clause defines the following **abstract-operations** available at the Retrieval Port

- a) Summarize;
- b) List;
- c) Fetch;
- d) Delete;
- e) Register-MS;
- f) Alert.

The MS is the MS abstract-service-provider of each of these **abstract-operations**. For the formal definition of the Retrieval Port see 6.2.

The abstract-operations may be performed asynchronously subject to the following conditions. The Delete and Register-MS abstract-operations shall not be performed until all outstanding abstract-operations have been completed. Additionally these abstract-operations are performed in the order in which they are invoked and are required to complete prior to any other abstract-operations being performed. As a consequence of this and the fact that the List and Fetch abstract-operations change the status of a message entry, the results of the Summarize, List and Fetch abstract-operations may be non-deterministic.

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 the remainder of clause 8. Many of the **abstract-operations** also make use of entries and attributes as defined in 6.3.

The common data-types defined in this part of ISO/IEC 10021 are

- a) Range;
- b) Filter;
- c) Selector;
- d) Entry Information Selection;
- e) Entry Information.

8.1.1 Range

A **range** parameter is used to select a contiguous sequence of entries from an information-base.

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 meanings:

1) **Sequence-number-range** (C), and

2) **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 meanings:

a) **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 message (lowest **sequence-number**) in the information-base.

b) **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 message (highest **sequence-number**) in the information-base.

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 of certain attributes of the entry, and is satisfied if and only if it evaluates to **true**.

Filter ::= CHOICE {
 item [0] **FilterItem**,
 and [1] SET SIZE (1 .. ub-nested-filters) OF **Filter**,
 or [2] SET SIZE (1 .. ub-nested-filters) OF **Filter**,
 not [3] **Filter**}

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

Where the **filter** is

- a) an **item**, it is **true** if and only if the corresponding **filter-item** is **true**;
- b) an **and**, it is **true** unless any of the **filters** in the **SET** are **false**;

NOTE- Thus, if there are no **filters** in the **SET**, the **and** evaluates to **true**.

c) an **or**, it is **false** unless any of the **filters** in the **SET** are **true**;

NOTE - Thus, if there are no **filters** in the **SET**, the **or** evaluates to **false**.

d) a **not**, it is **true** if and only if the **filter** is **false**.

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 either **true** or **false**.

FilterItem ::= CHOICE {	
equality	[0] AttributeValueAssertion,
substrings	[1] SEQUENCE {
type	AttributeType,
strings	SEQUENCE SIZE (1..ub-attribute-values) OF CHOICE {
initial	[0] ANY --DEFINED BY type--,
any	[1] ANY --DEFINED BY type--,
final	[2] ANY --DEFINED BY type--}},
greater-or-equal	[2] AttributeValueAssertion ,
less-or-equal	[3] AttributeValueAssertion,
present	[4] AttributeType,
approximate-match	[5] AttributeValueAssertion}

Every filter-item includes an attribute-type which identifies the particular attribute concerned.

Any assertion about the value of such an attribute is only evaluated if the attribute-type is defined, and the purported attribute-value(s) are of the data-type defined for attribute-values of that attribute.

Assertions about the value of an attribute are evaluated by matching the attribute for EQUALITY, SUBSTRINGS, and ORDERING, as defined in 6.3.3.4.

Where the **filter-item** asserts

- a) **equality**, it is **true** if and only if there is a value of the attribute which is equal to that asserted;
- b) **substrings**, it is **true** if and only if there is a value of the attribute in which the specified **substrings** appear in the given 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.

The first character in **initial**, if present, shall match the first character in the attribute-value; the last character in **final**, if present, shall match the last character in the attribute-value; **any**, if present, may match any substring in the attribute-value;

- c) **greater-or-equal**, it is **true** if and only if the relative ordering places the supplied value after any value of the attribute;
- d) **less-or-equal**, it is **true** if and only if the relative ordering places the supplied value before any value of the attribute;
- 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 which matches that which is asserted by some locally-defined approximate matching algorithm (e.g. spelling variations, phonetic match etc.) There are no specific guidelines for approximate matching in this version of ISO/IEC 10021-5. If approximate matching is not supported, this **FilterItem** should be treated as match for **equality**.

NOTE - If no matching rules are given in the attribute definition, this means that only the presence of the attribute can be tested in a filter-item.

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 involves an attribute-type and an attribute-value

```
AttributeValueAssertion ::= SEQUENCE {
    type      AttributeType,
    value     ANY DEFINED BY type}
```

and is

- a) **undefined**, if any of the following holds:
 - 1) the attribute-type is not present in the entry,
 - 2) the definition of the attribute-type cannot be matched for equality or ordering,
 - 3) the attribute-value does not conform to the data type of the attribute-values;
- b) **true**, if the entry contains an attribute of that attribute-type, one of whose attribute-values matches that attribute-value;
- c) **false**, otherwise.

8.1.3 Selector

A **selector** parameter is used to select entries from an information-base. The selection operates in three stages. Firstly, the total set of entries in the information-base may be restricted to particular contiguous set by specifying its range. Secondly, entries from within this set may be selected by specifying a filter which the selected entry shall satisfy. Thirdly, a limit may be placed on the number of entries thus selected; in this case, it is those entries with the lowest sequence-numbers which are selected.

```
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 -- default is that any fetch-restrictions in force do apply--}
```

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

- 1) **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, the default is *only main-entries are considered*.

- 2) **Range (O)**: The abstract-syntax-notation of **range** is given in 8.1.1.

In the absence of this component, the default is *unbounded*.

- 3) **Filter (O)**: The abstract-syntax-notation of **filter** is given in 8.1.2.

In the absence of this component, the default is *all entries within the specified range*.

- 4) **Limit (O)**: This allows the specification of an upper limit on how many entries shall be selected.

In the absence of this component, all of the selected entries will be returned.

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 can also be used to give back an exact number of information-sets to fit a particular output device.

- 5) **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 {
    overrideContentTypesRestriction    (0),
    overrideEITsRestriction            (1),
    overrideContentLengthRestriction   (2)} (SIZE (1..ub-information-bases))
```

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

- a) **Override-content-types-restriction (M)**: This bit shall be set to 1 if the **content-types-restriction** shall be overridden.

If this bit is set to 0, the **content-types-restrictions** as specified in the abstract-bind-operation will be applied.

- b) **Override-EITs-restriction (M)**: This bit shall be set to 1 if the **EITs-restriction** shall be overridden.

If this bit is set to 0, the **EITs-restrictions** as specified in the abstract-bind-operation will be applied.

- c) **Override-content-length-restriction (M)**: This bit shall be set to 1 if the **content-length-restriction** shall be overridden.

If this bit is set to 0, the **content-length-restrictions** as specified in the abstract-bind-operation will be applied.

In the absence of **override-restrictions**, the default is that all the **fetch-restrictions** as specified in the abstract-bind-operation will be applied.

8.1.4 Entry-information-selection

An **entry-information-selection** parameter indicates what information from an entry is being requested.

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

An empty set indicates that information about the entry itself, rather than the attributes of the entry, is being requested.

```

AttributeSelection ::= SET {
    type      AttributeType,
    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:

- 1) **Type (M)**: This indicates the attribute-type of the attribute.
- 2) **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 may only be present if the attribute-type is multi-valued. If it is omitted, values starting at the first value are returned.
- 3) **Count (O)**: When an attribute is multi-valued, this integer gives the number of values to be returned. If there are less than **count** values present in the attribute, all values are returned. This component may only be present if the attribute-type is multi-valued. If it is omitted, there is no limit as to how many values are returned.

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}

```

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

- 1) **Sequence-number (M)**: The sequence-number identifying the entry. See 6.3.2.2.
- 2) **Attributes (O)**: The set of selected attributes from the entry. Where explicitly requested by a partial-attribute-request, a selected attribute that is defined to be multi-valued may contain a subset of all the attribute-values in that attribute as stored in the entry. This parameter is absent if information from the selected messages is not requested, for example, when the MS abstract-service-user wants only the sequence-numbers of the selected messages.

8.2 Summarize abstract-operation

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

The **Summarize abstract-operation** will only be successful when the information-base permits access according to the security-context and the enforced security-policy.

The attributes that may be used for summaries are restricted. For the general-attributes in the stored-messages information-base, the restrictions are given in table 1.

```

Summarize :: = ABSTRACT-OPERATION
ARGUMENT   SummarizeArgument
RESULT     SummarizeResult
ERRORS {
    AttributeError,
    InvalidParametersError,
    RangeError,
    SecurityError,
    SequenceNumberError,
    ServiceError}

```

NOTE - An example of the Summarize abstract-operation is given in annex F.

8.2.1 Summarize-argument

```

SummarizeArgument :: = SET {
    information-base-type [0] InformationBase DEFAULT stored-messages,
    selector              [1] Selector,
    summary-requests      [2] SEQUENCE SIZE (1 .. ub-summaries) OF AttributeType OPTIONAL
    -- absent if no summaries are requested--}

```

The components of **summarize-argument** have the following meanings:

- 1) **Information-base-type** (O): This specifies which **information-base** is addressed by the abstract-operation. See 6.3.1.

In the absence of the **information-base-type** component, the default is **stored-messages**.

- 2) **Selector** (M): This is a set of selection criteria to determine which entries shall be summarized. See 8.1.3.
- 3) **Summary-requests** (O): This is the sequence of attribute-types for which summaries are requested. This parameter is only present if a summary is requested.

8.2.2 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}

```

The components of **summarize-result** have the following meanings:

- 1) **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 component contains the sequence-number for the next entry that would have been selected.
- 2) **Count** (M): This is an integer giving the count of entries that matched the selection criteria.
- 3) **Span** (C): This contains the lowest and highest sequence-numbers of the entries that matched 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 meanings:

- a) **Lowest (M)**: This is the starting-point for the **span**, given as a sequence-number. See 6.3.2.2.
- b) **Highest (M)**: This is the end-point for the **span**, given as a sequence-number. See 6.3.2.2.
- 4) **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 the attribute is absent--,
    present [1] SET SIZE (1 .. ub-attribute-values) OF --one for each attribute value present--
        SEQUENCE {
            type      AttributeType,
            value     ANY DEFINED BY type,
            count     INTEGER (1 .. ub-messages)} OPTIONAL}
```

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

- a) **Absent (C)**: A count of the entries that do not contain an attribute of the attribute-type specified in the request. It is omitted if there are no such entries.
- b) **Present (C)**: A summary of the entries that contain an attribute of the attribute-type specified, broken down by the attribute-values actually present. It is omitted if there are no such entries.

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

- 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.

8.2.3 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.

8.3 List abstract-operation

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

The **List abstract-operation** will only be successful when the information-base permits access according to the security-context and the enforced security-policy.

The information that may be selected for entries in an information-base may be restricted. For the general-attributes in the stored-messages information-base, the restrictions are given in table 1.

```
List :: = ABSTRACT-OPERATION
  ARGUMENT ListArgument
  RESULT ListResult
  ERRORS {
    AttributeError,
    InvalidParametersError,
    RangeError,
    SecurityError,
    SequenceNumberError,
    ServiceError}
```

8.3.1 List-argument

```
ListArgument :: = SET {
  information-base-type [0] InformationBase DEFAULT stored-messages,
  selector [1] Selector,
  requested-attributes [3] EntryInformationSelection OPTIONAL}
```

The components of **list-argument** have the following meanings:

- 1) **Information-base-type** (O): This specifies which information-base is addressed by the abstract-operation. See 6.3.1.

In the absence of the **information-base-type** component, the default is stored-messages.

- 2) **Selector** (M): This is a set of selection criteria to determine which entries shall be returned. See 8.1.3.
- 3) **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 **list-attribute-defaults** is used. See 8.6.1 for more information on these defaults.

8.3.2 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--}
```

The components of **list-result** have the following meanings:

- 1) **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 component contains the sequence-number for the next entry that would have been selected.
- 2) **Requested** (C): This conveys the requested entry-information, see 8.1.5, from each selected entry (one or more), in ascending order of sequence-number. It is not present in the case that a search was performed and no entry was selected.

8.3.3 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.

8.4 Fetch abstract-operation

The **Fetch abstract-operation** is used to return selected information from a specific entry in an information-base. 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 only be successful when information-bases permitted by the security-context and the security-policy in force are requested.

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

```
Fetch ::= ABSTRACT-OPERATION
  ARGUMENT  FetchArgument
  RESULT    FetchResult
  ERRORS {
    AttributeError,
    FetchRestrictionError,
    InvalidParametersError,
    RangeError,
    SecurityError,
    SequenceNumberError,
    ServiceError}
```

8.4.1 Fetch-argument

```
FetchArgument ::= SET [
  information-base-type [0] InformationBase DEFAULT stored-messages,
  item
    search [1] Selector,
    precise [2] SequenceNumber},
  requested-attributes [3] EntryInformationSelection OPTIONAL]
```

The components of **fetch-argument** have the following meanings:

- 1) **Information-base-type** (O): This specifies which information-base is addressed by the abstract-operation. See 6.3.1.

In the absence of the information-base-type component, the default is stored-messages.

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

- a) **Search** (C): This is a selector specifying a set of entries of which the one with the lowest sequence-number is the entry to be fetched, see 8.1.3.

- b) **Precise** (C): This is the sequence-number of the entry to be fetched. See 6.3.2.2.

- 3) **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 **fetch-attribute-defaults** is used. See 8.6.1 for more information on these defaults.

8.4.2 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}
```

The components of **fetch-result** have the following meanings:

- 1) **Entry-information** (C): This is a set of attributes from one entry as requested in the argument. See 8.1.5. It is not present in the case that a search was performed and no entry was selected.
- 2) **List** (C): This is returned in the case that a search was performed and more than one entry was found that matched the search selector. The list gives the sequence-numbers, in ascending order, of these further entries.
- 3) **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 component contains the sequence-number for the next entry that would have been selected.

8.4.3 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.

8.5 Delete abstract-operation

The **Delete abstract-operation** is used to delete selected entries from an information-base. A main-entry and all its dependent child-entries may only be deleted together. This is achieved by specifying just the main-entry as an argument. The Delete abstract-operation will only be successful when operating on those information-bases permitted by the security-context and the security-policy in force.

For specific information-bases, there may be restrictions on which entries may be deleted. In addition, content specific actions may be taken as defined in the corresponding International Standard which defines the content-type. For the stored-messages, no entry may be deleted if its entry-status, see 6.4, is new.

```
Delete :: = ABSTRACT-OPERATION
    ARGUMENT DeleteArgument
    RESULT DeleteResult
    ERRORS {
        DeleteError,
        InvalidParametersError,
        RangeError,
        SecurityError,
        SequenceNumberError,
        ServiceError}
```

8.5.1 Delete-argument

```

DeleteArgument ::= SET {
    information-base-type [0] InformationBase DEFAULT stored-messages,
    items CHOICE {
        selector [1] Selector
        sequence-numbers [2] SET SIZE (1..ub-messages) OF SequenceNumber}}

```

The components of **delete-argument** have the following meanings:

- 1) **Information-base-type** (O): This specifies which information-base is addressed by the abstract-operation. See 6.3.1.

In the absence of the **information-base-type** component, the default is stored-messages.

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

- a) **Selector** (C): See 8.1.3.

- b) **Sequence-numbers** (C): An unordered list of **sequence-numbers**. See 6.3.2.2.

8.5.2 Delete-result

Should the request succeed, the **delete-result** will be returned. There are no parameters.

```
DeleteResult ::= NULL
```

8.5.3 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.

8.6 Register-MS abstract-operation

The Register-MS abstract-operation is used to register or deregister various information with the MS:

- a) auto-actions;
- b) default list of attribute-types;
- c) new credentials;
- d) new set of user-security-labels.

```

Register-MS ::= ABSTRACT-OPERATION
    ARGUMENT Register-MSArgument
    RESULT Register-MSResult
    ERRORS {
        AttributeError,
        AutoActionRequestError,
        InvalidParametersError,
        SecurityError,
        ServiceError}

```

8.6.1 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 (1..ub-default-registrations) OF AttributeType OPTIONAL,
  fetch-attribute-defaults       [3] SET SIZE (1..ub-default-registrations) OF AttributeType OPTIONAL,
  change-credentials             [4] SEQUENCE {
    old-credentials              [0] IMPLICIT Credentials,
    new-credentials              [1] IMPLICIT Credentials} OPTIONAL
    -- same CHOICE as for old-credentials --,
  user-security-labels           [5] SET SIZE (1..ub-labels-and-redirections) OF SecurityLabel OPTIONAL
}

```

The components of register-MS-argument have the following meanings:

- 1) **Auto-action-registrations** (O): This is a set of **auto-action-registration**, see 6.5.1, one for each auto-action to be registered. The new **auto-action-registration-parameter** supersedes any previously registered auto-action (if any) with that **registration-identifier** and **auto-action-type**.

In the absence of **auto-action-registrations**, the default is that no new auto-actions are registered.

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

```

AutoActionDeregistration ::= AutoActionRegistration
  (WITH COMPONENTS {... , registration-parameter ABSENT})

```

In the absence of **auto-action-deregistrations**, the default is that no registered auto-actions are deregistered.

- 3) **List-attribute-defaults** (O): This specifies a default set of attribute-types to indicate which attributes should be returned for any subsequent List or Alert abstract-operation if the entry-information-selection argument is absent.

In the absence of **list-attribute-defaults**, the default is that 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.

- 4) **Fetch-attribute-defaults** (O): This specifies a default set of attribute-types to indicate which attributes should be returned for any subsequent Fetch abstract-operation if the entry-information-selection argument is absent.

In the absence of **fetch-attribute-defaults**, the default is that 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.

- 5) **Change-credentials** (O): The old and new credentials if a **change-credentials** is requested.

The **old-credentials** are the end user's current credentials, and the **new-credentials** are the credentials the end user would like to change to.

In the absence of this argument, the default is that previously registered credentials remain unchanged.

The credentials of the MS-user may differ from the **initiator-credentials** detailed in 8.1.1.1.2 of ISO/IEC 10021-4.

- 6) **User-security-labels** (O): This contains the **security-label(s)** of the MS abstract-service-user, if they are to be changed. It may be generated by the MS abstract-service-user.

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

Note that some **security-policies** may only permit the **user-security-labels** to be changed in this way if a secure link is employed. Other local means of changing the **user-security-labels** in a secure manner may be provided. **User-security-labels** is defined in 8.4.1.1.7 of ISO/IEC 10021-4.

Security-label is defined in clause 9 of ISO/IEC 10021-4.

8.6.2 Register-MS-result

Should the request succeed, the register-MS-result will be returned. There are no parameters.

Register-MSResult ::= NULL

8.6.3 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.

8.7 Alert abstract-operation

The **Alert abstract-operation** enables the MS abstract-service-provider to immediately inform the MS abstract-service-user of a new entry having been entered into the MS, whose attributes match the selection criteria of one of the **auto-alert-registrations**, see 12.2, previously supplied using a Register-MS abstract-operation, see 8.6.

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

Entries matching the selection criteria which have been created between abstract-associations will be indicated in the result of the next abstract-bind-operation for the abstract-association. No **Alert abstract-operation** will be invoked for these entries. See clause 7.

The **Alert abstract-operation** will only be successful when the information-base permits access according to the security-context and the enforced security-policy.

```
Alert ::= ABSTRACT-OPERATION
  ARGUMENT  AlertArgument
  RESULT    AlertResult
  ERRORS {
    SecurityError}
```

8.7.1 Alert-argument

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

The components of the **alert-argument** have the following meanings:

- 1) **Alert-registration-identifier** (M): Identifies which of the **auto-alert-registrations** resulted in the alert, see 6.4 and 12.2.
- 2) **New-entry** (O): This conveys the information from the new entry which was requested in the **auto-alert-registration-parameter**, see 12.2. It is absent when the MS abstract-service-user did not specify an **auto-alert-registration-parameter**.

8.7.2 Alert-result

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

AlertResult :: = NULL

8.7.3 Alert 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.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10021-5:1990

9 Abstract-errors

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

- a) AttributeError;
- b) AutoActionRequestError;
- c) DeleteError;
- d) FetchRestrictionError;
- e) InvalidParametersError;
- f) RangeError;
- g) SecurityError;
- h) SequenceNumberError;
- i) ServiceError.

9.1 Error precedence

The performer of an abstract-operation is not required to continue processing the message beyond the point at which an error has been detected. This allows an implementation to choose whether to continue the processing of errors.

NOTE - An implication of this rule is that the first error encountered may differ for repeated instances of the same abstract-operation, as there is not necessarily a specific logical order in which to process it.

9.2 Attribute-error

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

```
AttributeError ::= ABSTRACT-ERROR
  PARAMETER SET {
    problems[0] SET SIZE (1.. ub-per-entry) OF SET {
      problem [0] AttributeProblem,
      type     [1] AttributeType,
      value    [2] ANY DEFINED BY type OPTIONAL}}
```

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

The parameter has the following meaning:

- 1) **Problems (M)**: The particular problems encountered. Any numbers of individual problems may be indicated, each problem being accompanied by an indication of the attribute-type, and, if necessary to avoid ambiguity, the value which caused the problem:
 - a) **Invalid-attribute-value (C)**: 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 (C)**: A purported attribute-type used as an argument of the abstract-operation is not one of those which is supported by the MS abstract-service-provider. If the MS abstract-service-provider is able to carry out the operation anyway, it is not prohibited from doing so.
 - c) **Inappropriate-matching (C)**: The filter contains a filter-item in which an attribute is matched using an operation (equality, ordering, or substrings) that is not defined for that attribute.
 - d) **Attribute-type-not-subscribed (C)**: An attribute-type used as an argument of the abstract-operation is not one of those to which the MS abstract-service-user has subscribed.

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

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

9.3 Auto-action-request-error

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

```

AutoActionRequestError ::= ABSTRACT-ERROR
  PARAMETER SET {
    problems[0] SET SIZE (1..ub-auto-registrations) OF SET {
      problem [0] AutoActionRequestProblem,
      type     [1] AutoActionType}}

AutoActionRequestProblem ::= INTEGER {
  unavailable-auto-action-type      (0),
  auto-action-type-not-subscribed   (1)} (0..ub-error-reasons)
  
```

The parameter has the following meaning:

- 1) **Problems (M)**: The particular problems encountered. Any numbers of individual problems may be indicated, each problem being 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) **Action-type-not-subscribed**: An action-type used as an argument of the abstract-operation is not one of those to which the MS abstract-service-user has subscribed.

9.4 Delete-error

A **Delete-error** reports a problem in an attempt to delete one or more entries from an information-base.

```

DeleteError ::= ABSTRACT-ERROR
  PARAMETER SET {
    problems[0] SET SIZE (1..ub-messages) OF SET {
      problem          [0] DeleteProblem,
      sequence-number [1] SequenceNumber}}

DeleteProblem ::= INTEGER {
  child-entry-specified      (0),
  delete-restriction-problem (1)} (0..ub-error-reasons)

```

The parameter has the following meaning:

- 1) **Problems (M)**: The particular problems encountered. Any number of individual problems may be indicated, each problem being accompanied by an indication of the sequence-number of the entry which caused the problem:
 - a) **Child-entry-specified**: An attempt has been made to delete a child-entry.
 - b) **Delete-restriction-problem**: An attempt has been made to violate a restriction specified for the Delete abstract-operation (see 8.5).

9.5 Fetch-restriction-error

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

```

FetchRestrictionError ::= ABSTRACT-ERROR
  PARAMETER SET {
    problems[0] SET SIZE (1..ub-default-registrations) OF SET {
      problem          [3] FetchRestrictionProblem,
      restriction      CHOICE {
        content-type [0] ContentType,
        eit           [1] MS-EITs,
        content-length [2] ContentLength}}

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

```

The parameter has the following meaning:

- 1) **Problems (M)**: The particular problems encountered. Any number of individual problems may be indicated, each problem being accompanied by an indication of the offending content-type, encoded-information-type or content-length which caused the problem:
 - a) **Content-type-problem (C)**: The content-type of the message being fetched is disallowed by the fetch-restrictions currently in force.
 - b) **Eit-problem (C)**: The encoded-information-types requested in the Fetch abstract-operation are disallowed by the fetch-restrictions currently in force.

- c) **Content-length-problem (C)**: The content-length of the message being fetched is longer than that allowed by the fetch-restrictions currently in force.

9.6 Invalid-parameters-error

An **Invalid-parameters-error** reports a semantic problem in the set of parameters received. This error would 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.

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

This error has no parameters.

9.7 Range-error

A **Range-error** reports a problem related to the limit specified in a selector as an argument to an abstract-operation.

```
RangeError ::= ABSTRACT-ERROR
    PARAMETER SET {
        problem [0] RangeProblem}

RangeProblem ::= INTEGER {
    reversed (0)} (0 .. ub-error-reasons)
```

The parameter has the following meaning:

- 1) **problems (M)**: The particular problems encountered
 - a) **Reversed (C)**: The upper bound indicated a sequence-number or creation-time before that indicated by the lower bound.

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 ISO/IEC 10021-4.

9.9 Sequence-number-error

A **SequenceNumberError** reports a problem related to the sequence-number specified in an argument to an abstract-operation.

```
SequenceNumberError ::= ABSTRACT-ERROR
    PARAMETER SET {
        problems[1] SET SIZE (1..ub-messages) OF SET {
            problem [0] SequenceNumberProblem,
            sequence-number [1] SequenceNumber}}

SequenceNumberProblem ::= INTEGER {
    no-such-entry (0)} (0 .. ub-error-reasons)
```

The parameter has the following meaning:

1) **Problems (M)**: The particular problems encountered. Any numbers of individual problems may be indicated, each problem being accompanied by an indication of the sequence-numbers which caused the problem:

- a) **No-such-entry**: The sequence-number supplied does not match that of any entry in the information-base.

9.10 Service-error

A **Service-error** reports an error related to the provision of the service.

```
ServiceError ::= ABSTRACT-ERROR
    PARAMETER SET {
        problem      [0] ServiceProblem}

ServiceProblem ::= INTEGER {
    busy              (0),
    unavailable       (1),
    unwilling-to-perform (2)} (0 .. ub-error-reasons)
```

The parameter has the following meaning:

1) **Problem (M)**: The particular problem encountered:

- a) **Busy (C)**: The MS, or some part of it, is presently too busy to perform the requested abstract-operation, but may be able to do so after a short while.
- b) **Unavailable (C)**: The MS, or some part of it, is presently unavailable.
- c) **Unwilling-to-perform (C)**: The MS is not prepared to execute this request, because it would lead to excessive consumption of resources.

Section three - General-attribute-types and general-auto-action-types

10 Overview

The MS information-model and the **attribute** and **auto-action** concepts were introduced in 6.3.3 and 6.5. Clause 11 defines the **general-attribute-types** which are specified for MS. Clause 12 defines the **general-auto-action-types** which are specified for 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 International Standards, e.g. the IPMS-specific attribute-types for MS are defined in Annex C of ISO/IEC 10021-7.

11.1 General-attribute-types overview

The **general-attributes** that may occur in a stored-messages information-base entry are listed in Table 1. They are constructed mainly from the parameter information from the MessageDelivery and ReportDelivery abstract-operations of the MTS abstract-service as defined in clause 8 of 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 1 defines the various **general-attributes** and defines the following for each attribute-type :

- whether the attribute-type is single-valued or multi-valued;
- whether or not support by the MS and the accessing UA is mandatory or optional;
- 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;
- whether or not the attribute-type can be returned in a List or an Alert abstract-operation;
- whether or not the attribute-type may be used in a Summarize abstract-operation.

NOTE - Only for simple ASN.1 data-types.

For a more detailed description of the classifications in table 1 refer to the conventions in 5.2.

An optional attribute-type is only supported by an MS if the support of that attribute-type has successfully been subscribed to (which implies that the MS and the accessing UA supports that attribute). Subscription to optional attribute-types can be per attribute-type per UA.

Table 1 - General-attribute-types for the Stored-messages (Part 1 of 3)

Attribute-type name	Single/ multi valued	Support level by MS and access UA	Presence in delivered message entry	Presence in delivered report entry	Presence in returned- content entry	Available for List, Alert	Available for Summa- rize
Child-sequence-numbers	M	M	C	C	C	Y	N
Content	S	M	P	-	P	N	N
Content-confidentiality-algorithm-identifier	S	O	C	-	-	Y	N
Content-correlator	S	O	-	C	-	Y	N
Content-identifier	S	O	C	C	-	Y	N
Content-integrity-check	S	O	C	-	-	Y	N
Content-length	S	O	P	-	P	Y	N
Content-returned	S	O	-	P	-	Y	Y
Content-type	S	M	P	C	C	Y	Y
Conversion-with-loss-prohibited	S	O	C	-	-	Y	N
Converted-EITs	M	O	C	-	-	Y	N
Creation-time	S	M	P	P	P	Y	N
Delivered-EITs	M	O	P	-	-	Y	N
Delivery-flags	S	O	P	-	-	Y	N
DL-expansion-history	M	O	C	C	-	Y	N
Entry-status	S	M	P	P	P	Y	Y
Entry-type	S	M	P	P	P	Y	Y
Intended-recipient-name	S	O	C	-	-	Y	N
Message-delivery-envelope	S	M	P	-	-	N	N
Message-delivery-identifier	S	O	P	-	-	Y	N
Message-delivery-time	S	O	P	-	-	Y	N

Table 1 - General-attribute-types for the Stored-messages (Part 2 of 3)

Attribute-type name	Single/ multi valued	Support level by MS and access UA	Presence in delivered message entry	Presence in delivered report entry	Presence in returned- content entry	Available for List, Alert	Available for Summa- rize
Message-origin-authentication-check	S	O	C	-	-	Y	N
Message-security-label	S	O	C	C	-	Y	N
Message-submission-time	S	O	P	-	-	Y	N
Message-token	S	O	C	-	-	Y	N
Original-EITs	M	O	C	C	-	Y	N
Originator-certificate	S	O	C	-	-	Y	N
Originator-name	S	O	P	-	-	Y	N
Other-recipient-names	M	O	C	-	-	Y	N
Parent-sequence-number	S	M	C	-	P	Y	N
Per-recipient-report-delivery-fields	M	M	-	P	-	Y	N
Priority	S	O	P	-	-	Y	Y
Proof-of-delivery-request	S	O	C	-	-	Y	N
Redirection-history	M	O	C	-	-	Y	N
Report-delivery-envelope	S	M	-	P	-	N	N
Reporting-DL-name	S	O	-	C	-	Y	N
Reporting-MTA-certificate	S	O	-	C	-	Y	N
Report-origin-authentication-check	S	O	C	C	-	Y	Y
Security-classification	S	O	C	C	-	Y	Y
Sequence-number	S	M	P	P	P	Y	N

Table 1 - General-attribute-types for the Stored-messages (Part 3 of 3)

Attribute-type name	Single/ multi valued	Support level by MS and access UA	Presence in delivered message entry	Presence in delivered report entry	Presence in returned- content entry	Available for List, Alert	Available for Summa- rize
Subject-submission- identifier	S	M	-	P	-	Y	N
This-recipient-name	S	O	P	-	-	Y	N

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 **ATTRIBUTE** macro described in 6.3.

It should be noted that some **general-attributes** are used primarily for filtering and listing purposes while others can contain more complex (further structured ASN.1 data-types) and potentially voluminous information. Only a few **general-attributes** are suitable for summaries.

11.2.1 Child-sequence-numbers

This general-attribute, which is multi-valued, contains one or more "pointers" to the next level of child-entries, if such exist. It is generated by the MS. It is present in a parent-entry that has one or more child-entries associated with it. It is absent in an entry without child-entries.

ms-child-sequence-numbers **ATTRIBUTE**
WITH **ATTRIBUTE-SYNTAX** SequenceNumber
MULTI VALUE
:: = id-att-child-sequence-numbers

11.2.2 Content

This general-attribute contains the complete content of a message as delivered by the MessageDelivery abstract-operation or as a returned-content by the ReportDelivery abstract-operation. For more details see 8.2.1.1.1.37 and 8.3.1.2.1.14 of ISO/IEC 10021-4.

ms-content **ATTRIBUTE**
WITH **ATTRIBUTE-SYNTAX** Content
SINGLE VALUE
:: = id-att-content

11.2.3 Content-confidentiality-algorithm-identifier

This general attribute contains the **algorithm-identifier** used by the originator of the message to encrypt the message content. It may be generated by the originator of the message. For further details see 8.5.10 of ISO/IEC 10021-4.

mt-content-confidentiality-algorithm-identifier ATTRIBUTE
WITH ATTRIBUTE-SYNTAX AlgorithmIdentifier
SINGLE VALUE
:: = id-att-content-confidentiality-algorithm-identifier

11.2.4 Content-correlator

This general-attribute contains information to enable correlation of the content of the message. It may be generated by the originating UA. For more details see 8.2.1.1.1.36 of ISO/IEC 10021-4.

mt-content-correlator ATTRIBUTE
WITH ATTRIBUTE-SYNTAX ContentCorrelator
MATCHES FOR EQUALITY
SINGLE VALUE
:: = id-att-content-correlator

11.2.5 Content-identifier

This general-attribute contains an identifier for the content of the message. It may be generated by the originating UA. For more details see 8.2.1.1.1.35 of ISO/IEC 10021-4.

mt-content-identifier ATTRIBUTE
WITH ATTRIBUTE-SYNTAX ContentIdentifier
MATCHES FOR EQUALITY
SINGLE VALUE
:: = id-att-content-identifier

11.2.6 Content-integrity-check

This general attribute provides the recipient(s) of the message with a means of validating that the message content has not been modified. It may be generated by the originator of the message and may specify a different value for each recipient of the message. For further details see 8.2.1.1.28 of ISO/IEC 10021-4.

mt-content-integrity-check ATTRIBUTE
WITH ATTRIBUTE-SYNTAX ContentIntegrityCheck
SINGLE VALUE
:: = id-att-content-integrity-check

11.2.7 Content-length

This general-attribute gives the length of the content, in octets, of a message as delivered by the MessageDelivery abstract-operation or of a returned-content (if any) notified by the ReportDelivery abstract-operation. Where there is no such returned-content, this attribute is absent. It is generated by the MS.

ms-content-length ATTRIBUTE
WITH ATTRIBUTE-SYNTAX ContentLength
MATCHES FOR ORDERING
SINGLE VALUE
:: = id-att-content-length

11.2.8 Content-returned

This general-attribute indicates whether a content has been returned in the ReportDelivery abstract-operation. It is generated by the MS.

ms-content-returned ATTRIBUTE
 WITH ATTRIBUTE-SYNTAX BOOLEAN
 MATCHES FOR EQUALITY
 SINGLE VALUE
 ::= id-att-content-returned

11.2.9 Content-type

This general-attribute is generated from the content-type in the MessageDelivery or ReportDelivery abstract-operation. See also 8.2.1.1.1.34 of ISO/IEC 10021-4.

mt-content-type ATTRIBUTE
 WITH ATTRIBUTE-SYNTAX OBJECT IDENTIFIER
 MATCHES FOR EQUALITY
 SINGLE VALUE
 ::= id-att-content-type

11.2.10 Conversion-with-loss-prohibited

This general-attribute contains information about whether conversion with loss of information was allowed or prohibited. For further details see 8.2.1.1.1.10 of ISO/IEC 10021-4.

mt-conversion-with-loss-prohibited ATTRIBUTE
 WITH ATTRIBUTE-SYNTAX ConversionWithLossProhibited
 MATCHES FOR EQUALITY
 SINGLE VALUE
 ::= id-att-conversion-with-loss-prohibited

11.2.11 Converted-EITs

This general-attribute, which is multi-valued, identifies the encoded-information-types of the content after conversion, as indicated by MessageDelivery or ReportDelivery abstract-operation. It is generated by the MS. It is absent if no conversion took place. For more details see 8.3.1.1.1.8 and 8.3.1.2.1.5 of ISO/IEC 10021-4.

ms-converted-EITs ATTRIBUTE
 WITH ATTRIBUTE-SYNTAX MS-EIT
 MATCHES FOR EQUALITY
 MULTI VALUE
 ::= id-att-converted-EITs

11.2.12 Creation-time

This general-attribute gives the time when the entry was created in the MS. It is generated by the MS. For more details see 6.3.2.

NOTE - Two or more consecutive entries may have the same creation-time.

ms-creation-time ATTRIBUTE
WITH ATTRIBUTE-SYNTAX CreationTime
MATCHES FOR EQUALITY ORDERING
SINGLE VALUE
:: = id-att-creation-time

11.2.13 Delivered-EITs

This general-attribute, which is multi-valued, identifies the encoded-information-types in the content of the message as delivered. It is generated by the MS based on information about the original-EITs and the converted-EITs in the MessageDelivery abstract-operation.

ms-delivered-EITs ATTRIBUTE
WITH ATTRIBUTE-SYNTAX MS-EIT
MATCHES FOR EQUALITY
MULTI VALUE
:: = id-att-delivered-EITs

11.2.14 Delivery-flags

This general-attribute contains information of the delivery. Presently, it is only used for indicating implicit-conversion of the content. For more details see 8.2.1.1.1.9 of ISO/IEC 10021-4.

mt-delivery-flags ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DeliveryFlags
MATCHES FOR EQUALITY
SINGLE VALUE
:: = id-att-delivery-flags

11.2.15 DL-expansion-history

This general-attribute, which is multi-valued, is used to show the history of distribution-list expansion. It contains one or more distribution-list names used during the expansion process. It is absent if the delivery to this recipient did not involve any expansion of a distribution-list. For more details see 8.3.1.1.1.7 of ISO/IEC 10021-4.

mt-dl-expansion-history ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DLExpansionHistory
MULTI VALUE
:: = id-att-dl-expansion-history

11.2.16 Entry-status

This general-attribute contains the current status of an entry in the stored-messages information-base. It is created and maintained by the MS. For more details see 6.4.

ms-entry-status ATTRIBUTE
WITH ATTRIBUTE-SYNTAX EntryStatus
MATCHES FOR EQUALITY
SINGLE VALUE
:: = id-att-entry-status

11.2.17 Entry-type

This general-attribute contains information about whether an entry concerns a delivered message or a delivered report. It is generated by the MS.

```
ms-entry-type ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX EntryType
  MATCHES FOR EQUALITY
  SINGLE VALUE
  ::= id-att-entry-type

EntryType ::= INTEGER {
  delivered-message      (0),
  delivered-report       (1),
  returned-content       (2) (0..ub-entry-types)}
```

11.2.18 Intended-recipient-name

This general-attribute contains the O/R-name of the originally intended recipient if the message has been redirected, with each value representing one redirection. For more details see 8.3.1.1.1.4 of ISO/IEC 10021-4.

```
mt-intended-recipient-name ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX ORName
  MATCHES FOR EQUALITY
  SINGLE VALUE
  ::= id-att-intended-recipient-name
```

11.2.19 Message-delivery-envelope

This general-attribute contains the complete **message-delivery-envelope** of a message as delivered by the MessageDelivery abstract-operation. For more details see clause 9 of ISO/IEC 10021-4.

```
mt-message-delivery-envelope ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX MessageDeliveryEnvelope
  SINGLE VALUE
  ::= id-att-message-delivery-envelope
```

11.2.20 Message-delivery-identifier

This general-attribute contains the **message-delivery-identifier** from the MessageDelivery abstract-operation. For more details see 8.3.1.1.1.1 of ISO/IEC 10021-4.

```
mt-message-delivery-identifier ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX MessageDeliveryIdentifier
  SINGLE VALUE
  ::= id-att-message-delivery-identifier
```

11.2.21 Message-delivery-time

This general-attribute contains the **message-delivery-time** from the MessageDelivery abstract-operation. For more details see 8.3.1.1.1.2 of ISO/IEC 10021-4.

NOTE - There is no general-attribute corresponding to the delivery-time parameter of the ReportDelivery 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. This information is included in the report-information general-attribute.

mt-message-delivery-time ATTRIBUTE
WITH ATTRIBUTE-SYNTAX MessageDeliveryTime
MATCHES FOR EQUALITY ORDERING
SINGLE VALUE
:: = id-att-message-delivery-time

11.2.22 Message-origin-authentication-check

This general attribute is computed using the algorithm identified by the message-origin-authentication-identifier. It provides the recipient(s) of the message with a means of authenticating the origin of the message and may be generated by the originator of the message. For further details see 8.2.1.1.1.29 of ISO/IEC 10021-4.

mt-message-origin-authentication-check ATTRIBUTE
WITH ATTRIBUTE-SYNTAX MessageOriginAuthenticationCheck
SINGLE VALUE
:: = id-att-message-origin-authentication-check

11.2.23 Message-security-label

This general attribute 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. For further details see 8.2.1.1.1.30 of ISO/IEC 10021-4.

mt-message-security-label ATTRIBUTE
WITH ATTRIBUTE-SYNTAX MessageSecurityLabel
SINGLE VALUE
:: = id-att-message-security-label

11.2.24 Message-submission-time

This general-attribute contains the message-submission-time from a MessageDelivery abstract-operation. For more details see 8.2.1.1.2.2 of ISO/IEC 10021-4.

mt-message-submission-time ATTRIBUTE
WITH ATTRIBUTE-SYNTAX MessageSubmissionTime
MATCHES FOR EQUALITY ORDERING
SINGLE VALUE
:: = id-att-message-submission-time

11.2.25 Message-token

This general attribute contains the token associated with the message. It is generated by the originator of the message and may contain a different value for each recipient of the message. For further details see 8.2.1.1.1.26 of ISO/IEC 10021-4.

mt-message-token ATTRIBUTE
WITH ATTRIBUTE-SYNTAX MessageToken
SINGLE VALUE
:: = id-att-message-token

11.2.26 Original-EITs

This general-attribute, which is multi-valued, contains the **original encoded-information-types** from the MessageDelivery abstract-operation. It is generated by the MS. For more details see 8.2.1.1.1.33 of ISO/IEC 10021-4.

```
ms-original-EITs ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX MS-EIT
  MATCHES FOR EQUALITY
  MULTI VALUE
  ::= id-att-original-EITs
```

11.2.27 Originator-certificate

This general attribute, contains the certificate of the originator of the message. It is generated by a trusted source (e.g. a certification-authority), and may be supplied by the originator of the message. For further details see 8.2.1.1.1.25 of ISO/IEC 10021-4.

```
mt-originator-certificate ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX OriginatorCertificate
  SINGLE VALUE
  ::= id-att-originator-certificate
```

11.2.28 Originator-name

This general-attribute contains the O/R-name of the originator from the MessageDelivery abstract-operation. For more details see 8.2.1.1.1.1 of ISO/IEC 10021-4.

```
mt-originator-name ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX ORName
  MATCHES FOR EQUALITY
  SINGLE VALUE
  ::= id-att-originator-name
```

11.2.29 Other-recipient-names

This general-attribute, which is multi-valued, contains the O/R-names of all other specified recipients, if any, of the message from the MessageDelivery abstract-operation. For more details see 8.3.1.1.1.6 of ISO/IEC 10021-4.

```
mt-other-recipient-names ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX ORName
  MATCHES FOR EQUALITY
  MULTI VALUE
  ::= id-att-other-recipient-names
```

11.2.30 Parent-sequence-number

This general-attribute, points to a parent-entry. It is generated by the MS. It is always present in a child-entry and is absent in a main-entry.

ms-parent-sequence-number ATTRIBUTE
WITH ATTRIBUTE-SYNTAX SequenceNumber
MATCHES FOR EQUALITY ORDERING
SINGLE VALUE
:: = id-att-parent-sequence-number

11.2.31 Per-recipient-report-delivery-fields

This general-attribute, which is multi-valued, contains report information on a per-recipient basis from the ReportDelivery abstract-operation. For more details see 8.3.1.2 of ISO/IEC 10021-4.

mt-per-recipient-report-delivery-fields ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PerRecipientReportDeliveryFields
MULTI VALUE
:: = id-att-per-recipient-report-delivery-fields

11.2.32 Priority

This general-attribute contains the relative **priority** of the message from the MessageDelivery abstract-operation. If no value is supplied in the MessageDelivery abstract-operation parameter, the MS uses its default value when generating this attribute. For more details see 8.2.1.1.1.8 of ISO/IEC 10021-4.

mt-priority ATTRIBUTE
WITH ATTRIBUTE-SYNTAX Priority
MATCHES FOR EQUALITY
SINGLE VALUE
:: = id-att-priority

11.2.33 Proof-of-delivery-request

This general attribute indicates whether or not the originator of the message requires **proof-of-delivery** of the message to the recipient. It may be generated by the originator of the message and may specify a different value for each recipient of the message. For more details see 8.2.1.1.1.32 of ISO/IEC 10021-4.

mt-proof-of-delivery-request ATTRIBUTE
WITH ATTRIBUTE-SYNTAX ProofOfDeliveryRequest
SINGLE VALUE
:: = id-att-proof-of-delivery-request

11.2.34 Redirection-history

This general-attribute, which is multi-valued, contains the history of recipient redirection(s) with reason(s) from the MessageDelivery or ReportDelivery abstract-operation. For more details see 8.3.1.1.1.5 of ISO/IEC 10021-4.

mt-redirection-history ATTRIBUTE
WITH ATTRIBUTE-SYNTAX RedirectionHistory
MULTI VALUE
:: = id-att-redirection-history

11.2.35 Report-delivery-envelope

This general-attribute contains all the parameters from the ReportDelivery abstract-operation, except for the returned-content (if present). For more details see 8.3.1.2 of ISO/IEC 10021-4.

mt-report-delivery-envelope ATTRIBUTE
WITH ATTRIBUTE-SYNTAX ReportDeliveryEnvelope
SINGLE VALUE
:: = id-att-report-delivery-envelope

11.2.36 Reporting-DL-name

This general-attribute contains the O/R-name of the distribution-list that forwarded the report to the owner of this distribution-list. For more details see 8.3.1.2.1.4 of ISO/IEC 10021-4.

mt-reporting-DL-name ATTRIBUTE
WITH ATTRIBUTE-SYNTAX ReportingDLName
SINGLE VALUE
:: = id-att-reporting-DL-name

11.2.37 Reporting-MTA-certificate

This general-attribute contains the certificate of the MTA that generated the report. For more details see 8.3.1.2.1.12 of ISO/IEC 10021-4.

mt-reporting-MTA-certificate ATTRIBUTE
WITH ATTRIBUTE-SYNTAX ReportingMTACertificate
SINGLE VALUE
:: = id-att-reporting-MTA-certificate

11.2.38 Report-origin-authentication-check

This general-attribute provides a means of authenticating the origin of the report. For more details see 8.3.1.2.1.13 of ISO/IEC 10021-4.

mt-report-origin-authentication-check ATTRIBUTE
WITH ATTRIBUTE-SYNTAX ReportOriginAuthenticationCheck
SINGLE VALUE
:: = id-att-report-origin-authentication-check

11.2.39 Security-classification

This general-attribute comprises the security-classification parameter from the message-security-label. It is defined as a separate attribute to allow its use in the Summarize abstract-operation. For more details see 8.5.9 of ISO/IEC 10021-4.

mt-security-classification ATTRIBUTE
WITH ATTRIBUTE-SYNTAX SecurityClassification
MATCHES FOR EQUALITY
SINGLE VALUE
:: = id-att-security-classification

11.2.40 Sequence-number

This general-attribute is used to identify the entry itself. It is allocated by the MS when the entry is created. For more details see 6.3.2.

ms-sequence-number ATTRIBUTE
WITH ATTRIBUTE-SYNTAX SequenceNumber
MATCHES FOR EQUALITY ORDERING
SINGLE VALUE
:: = id-att-sequence-number

11.2.41 Subject-submission-identifier

This general-attribute contains the message-submission-identifier or the probe-submission-identifier of the subject of the report. For more details see 8.3.1.2.1.1 of ISO/IEC 10021-4.

mt-subject-submission-identifier ATTRIBUTE
WITH ATTRIBUTE-SYNTAX SubjectSubmissionIdentifier
SINGLE VALUE
:: = id-att-subject-submission-identifier

11.2.42 This-recipient-name

This general-attribute contains the O/R-name of this (MS) recipient from the MessageDelivery abstract-operation. For more details see 8.3.1.1.1.3 of ISO/IEC 10021-4.

mt-this-recipient-name ATTRIBUTE
WITH ATTRIBUTE-SYNTAX ORName
MATCHES FOR EQUALITY
SINGLE VALUE
:: = id-att-this-recipient-name

11.3 Generation of the general-attributes

This subclause describes how the general-attributes are generated. The information is presented in Table 2. For a description of the classifications used see 5.3.

11.4 Attribute-types subscription

Attribute-type subscription is a local matter. If the attribute-type subscription is changed, then the UA may receive all of the attributes in the original subscription for messages present in the MS at the time the subscription was changed. The handling of these unsubscribed attributes is a local matter. Similarly, when a new attribute is subscribed to, the UA may not receive this attribute for messages in the MS when the subscription occurred.

Table 2 - Generation of the general-attribute-types (Part 1 of 4)

Attribute-type name	Single/ multi valued	Source parameter	Source generated by	Generation rules
Child-sequence-numbers	M	-	MS	A value is generated for each corresponding child-entry that a parent-entry has
Content	S	content returned-content	MD RD	The value of the parameter is the attribute-value
Content-confidentiality-algorithm-identifier	S	content- confidentiality- algorithm- identifier	MD	The value of the parameter is the attribute-value
Content-correlator	S	content-correlator	RD	The value of the parameter is the attribute-value
Content-identifier	S	content-identifier	MD RD	The value of the parameter is the attribute-value
Content-integrity-check	S	content-integrity- check	MD	The value of the parameter is the attribute-value
Content-length	S	-	MS	The (approximate) size of the stored content in octets based on the delivered or returned content
Content-returned	S	-	MS	The value is set to true if returned-content is present in a ReportDelivery and to false if not present
Content-type	S	content-type	MD RD	If represented by OBJECT IDENTIFIER, the value of the parameter. If represented by INTEGER, converted to the corresponding OBJECT IDENTIFIER
Conversion-with-loss-prohibited	S	conversion-with- loss-prohibited	MD	The value of the parameter is the attribute-value

NOTE - When a message-delivery entry is created, there are no separate general-attributes generated for Physical Delivery and Delivery Method Arguments, because the information in these arguments are not relevant to the MS. However, the UA can retrieve all the information contained in these arguments by retrieving the Message-delivery-envelope general-attribute.

Table 2 - Generation of the general-attribute-types (Part 2 of 4)

Attribute-type name	Single/ multi valued	Source parameter	Source generated by	Generation rules
Converted-EITs	M	converted- encoded- information-types	MD	A corresponding value is generated from each bit that it set to 1 in the built-in-encoded-information-types parameter and from each ExternalEncodedInformationType present in the external-encoded-information-type parameter.
Creation-time	S	-	MS	The time of creation of the entry
Delivered-EITs	M	Converted-EITs and Original-EITs	MS	A union of the other two general-attribute-types.
Delivery-flags	S	delivery-flags	MD	The value of the parameter is the attribute-value. If there are no delivery-flags in the MD, generate a default value with no flags set.
DL-expansion-history	M	dl-expansion- history	MD RD	A corresponding value is generated from each component of the SEQUENCE
Entry-status	S	-	MS	Generated when the entry is created with the value "new"
Entry-type	S	MessageDelivery ARGUMENT ReportDelivery ARGUMENT	MS MS	The value is set to "delivered-message" The value is set to "delivered-report". If a returned-content is present, a child-entry, which contains the returned-content is created. For the child-entry, the value of this attribute is set to "returned-content"
Intended-recipient-name	S	intended- recipient-name	MD	The value of the parameter is the attribute-value

Table 2 - Generation of the general-attribute-types (Part 3 of 4)

Attribute-type name	Single/ multi valued	Source parameter	Source generated by	Generation rules
Message-delivery-envelope	S	envelope	MD	The value of the parameter is the attribute-value
Message-delivery-time	S	message-delivery-time	MD	The value of the parameter is the attribute-value
Message-delivery-identifier	S	message-delivery-identifier	MD	The value of the parameter is the attribute-value
Message-origin-authentication-check	S	message-origin-authentication-check	MD	The value of the parameter is the attribute-value
Message-security-label	S	message-security-label	MD RD	The value of the parameter is the attribute-value
Message-submission-time	S	message-submission-time	MD	The value of the parameter is the attribute-value
Message-token	S	message-token	MD	The value of the parameter is the attribute-value
Original-EITs	M	original-encoded-information-types	MD RD	A corresponding value is generated from each bit that it set to 1 in the built-in-encoded-information-types parameter and from each ExternalEncoded InformationType present in the external-encoded-information-type parameter
Originator-certificate	S	originator-certificate	MD	The value of the parameter is the attribute-value
Originator-name	S	originator-name	MD	The value of the parameter is the attribute-value
Other-recipient-names	M	other-recipient-names	MD	A corresponding value is generated from each component of the SEQUENCE

Table 2 - Generation of the general-attribute-types (Part 4 of 4)

Attribute-type name	Single/ multi valued	Source parameter	Source generated by	Generation rules
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-recipient-report-delivery-fields	M	per-recipient-fields	RD	A corresponding value is generated from each component of the SEQUENCE
Priority	S	priority	MD	The value of the parameter is the attribute-value
Proof-of-delivery-request	S	proof-of-delivery-request	MD	The value of the parameter is the attribute-value
Redirection-history	M	redirection-history	MD	A corresponding value is generated from each component of the SEQUENCE
Report-delivery-envelope	S	envelope	RD	The value of the parameter is the attribute-value
Reporting-DL-name	S	reporting-dl-name	RD	The value of the parameter is the attribute-value
Reporting-MTA-certificate	S	reporting-MTA-certificate	RD	The value of the parameter is the attribute-value
Report-origin-authentication-check	S	report-origin-authentication-check	RD	The value of the parameter is the attribute-value
Security-classification	S	security-classification	MD RD	The value of the parameter is the attribute-value
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 value of the parameter is the attribute-value
This-recipient-name	S	this-recipient-name	MD	The value of the parameter is the attribute-value

12 General-auto-action-types

The **general-auto-action-types** are valid for all message content-types. However, their detailed effect may be content-specific, and so the procedure descriptions given in this part of ISO/IEC 10021 may need to be supplemented in their respective International Standards, e.g. the IPMS-specific procedure for the auto-forward **general-auto-action-type** is described in 19.4 of ISO/IEC 10021-7. Other **auto-action-types**, which are content-specific, may be defined in their respective International Standards.

Auto-actions are introduced in 6.5 and are registered and deregistered using the Register-MS abstract-operation described in 8.6.

The following **general-auto-action-types** are defined:

- a) Auto-forward;
- b) Auto-alert.

The operation of **auto-actions** may be affected by the implementation of a security-policy.

The following subclauses contain a short description of each **general-auto-action-type** together with its abstract-syntax using the **AUTO-ACTION** macro defined in 6.5.

12.1 Auto-forward

The **auto-forward auto-action** enables the MS abstract-service-provider to automatically **forward** any message that has been delivered into the stored-messages information-base. The exact definition of forwarding is content-specific, but it always involves the submission of a new message incorporating the delivered content to the MTS abstract-service.

The **auto-forward auto-action-type** allows one or more sets of **auto-forward** parameters to be registered with the MS, each identified by its **auto-forward-registration-identifier**. Each **auto-forward-registration-parameter** specifies criteria to determine whether it applies to a particular delivered message, and if so a copy of the message is **auto-forwarded** using the Message-submission abstract-operation. That is to say, if a message matches more than one set of criteria, the message is **auto-forwarded** that many times.

The **auto-forward-registration-parameter** specifies whether the main-entry (and any associated child-entries) corresponding to the message is to be deleted after **auto-forwarding**. If any of the parameters acted upon indicates no-deletion (or if any of the submissions fail), then the entry is not deleted.

auto-forward AUTO-ACTION

REGISTRATION PARAMETER IS AutoForwardRegistrationParameter
:: = id-act-auto-forward

AutoForwardRegistrationParameter :: = SET {
 filter [0] **Filter OPTIONAL,**
 auto-forward-arguments [1] **AutoForwardArguments,**
 delete-after-auto-forwarding [2] **BOOLEAN DEFAULT FALSE,**
 other-parameters [3] **OCTET STRING OPTIONAL}**
}

AutoForwardArguments ::= SET {
COMPONENTS OF PerMessageAutoForwardFields,
per-recipient-fields [1] IMPLICIT SEQUENCE SIZE (1..ub-recipients) OF PerRecipientAutoForwardFields}

PerMessageAutoForwardFields ::= SET {
originator-name OriginatorName,
content-identifier ContentIdentifier OPTIONAL,
priority Priority DEFAULT normal,
per-message-indicators PerMessageIndicators DEFAULT { },
deferred-delivery-time [0] IMPLICIT DeferredDeliveryTime OPTIONAL,
extensions [2] IMPLICIT PerMessageSubmissionExtensions DEFAULT { }}

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

The parameters of the **auto-forward-registration-parameter** have the following meanings:

- 1) **Filter (O)**: This is a set of criteria which a new entry representing a delivered message shall satisfy for the MS abstract-service-provider to **auto-forward** it using this set of parameters.

The absence of this parameter indicates that all new entries are **auto-forwarded**.

- 2) **Auto-forward-arguments (M)**: This is a set of arguments registered to be used for each Message-submission abstract-operation (see 8.2.1.1.1 of ISO/IEC 10021-4). Any argument which is not registered, not mandatory, and not specifically mentioned below, will be absent from each Message-submission.

If the following arguments are either not registered, or registered with their default values, the values used for each Message-submission abstract-operation are those of the corresponding Message-delivery arguments: **priority**, **implicit-conversion-prohibited**, and **conversion-with-loss-prohibited**.

If the following arguments are either not registered, or registered with their default values, their presence as Message-submission arguments depends upon the presence of the corresponding Message-delivery arguments, their values being transformed where appropriate: **message-token**, **content-confidentiality-algorithm-identifier**, **content-integrity-check**, **message-origin-authentication-check**, and **message-security-label**.

Certain Message-submission arguments may not be registered. These are: **proof-of-submission-request**, **original-encoded-information-types**, **content-type**, and **content**.

- 3) **Delete-after-auto-forwarding (O)**: This indicates whether an entry should be deleted or not, once the submission has succeeded.

The absence of this parameter indicates that the message should not be deleted.

- 4) **Other-parameters (O)**: This content-specific parameter need not be present. When it is present, the information it contains will be used during the **auto-forwarding** procedure.

NOTE - Thus, for example, with Interpersonal Messaging, this parameter may contain the **auto-forward-comment** that is returned in the non-receipt notification, a user specified prefix and a cover-note accompanying the IP-message being auto-forwarded. For a description of **auto-forward-comment** usage, see 19.4 of ISO/IEC 10021-7.

12.2 Auto-alert

The **auto-alert auto-action** enables the MS abstract-service-provider to automatically *alert* the user behind the MS abstract-service-user of the delivery of any message that has been delivered into the stored-messages information-base. **Auto-alert** will only be performed for delivered-message entries.

The **auto-alert auto-action-type** allows one or more sets of **auto-alert** parameters to be registered with the MS, each identified by its **auto-alert-registration-identifier**. Each **auto-alert-registration-parameter** specifies criteria to determine whether it applies to a particular delivered message. If a message matches the filter of more than one auto-alert-registration the matching registration with the lowest auto-alert-registration-identifier is processed, and if at least one address (or the UA) has been alerted successfully, no other registrations are processed. If none of these addresses can be successfully alerted, the auto-alert registration with the next higher identifier is processed. This continues until either at least one or more addresses of a registration has been successfully alerted or the list of registrations is exhausted.

The **Alert abstract-operation** will only be invoked if the alert-addresses in the auto-alert-registration is considered to have the UA as a member (see 2 below). If this Alert abstract-operation succeeds, any other address contained in the auto-alert-registration will not be alerted.

auto-alert AUTO-ACTION

REGISTRATION PARAMETER IS AutoAlertRegistrationParameter
::= id-act-auto-alert

AutoAlertRegistrationParameter ::= SET {

filter	[0] Filter OPTIONAL,
alert-addresses	[1] SEQUENCE SIZE (1 .. ub-alert-addresses) OF AlertAddress OPTIONAL,
requested-attributes	[2] EntryInformationSelection OPTIONAL}

The parameters of the **auto-alert-registration-parameter** have the following meanings:

- 1) **Filter (O)**: This is a set of criteria which a new entry representing a delivered-message shall satisfy for the MS abstract-service-provider to **auto-alert** it using this set of parameters.

The absence of this parameter indicates that **auto-alert** will be performed for all new delivered-message entries.

- 2) **Alert-addresses (O)**: This argument identifies the types of **alert** service to be invoked, together with any information required to access the particular instances of those **alert** services, and any further information that needs to be conveyed during those **alerts**.

Absence of this argument will default the Alert abstract-operation to informing the MS abstract-service-user of the existence of an alert-condition either by using the Alert abstract-operation (see 8.7), (which is only possible if an abstract-association already exists between the MS abstract-service-user and the MS abstract-service-provider) or by flagging in the Abstract-bind-operation next time the MS abstract-service-user establishes an abstract-association, see clause 7. If the parameter requested-attributes is present, the MS abstract-service-user (UA) will be considered as being among the addresses to be alerted.

Some types of **alert** will be internationally standardized. Others will 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 type with an object-identifier when the type is defined, and defining the ASN.1 data-type of the auxiliary addressing information.

The **alert-qualifier** contains any further information that needs to be conveyed during the **auto-alert**. Absence of this parameter means that no additional information will be conveyed to the MS abstract-service-user.

```
AlertAddress :: = SEQUENCE {  
    address          EXTERNAL,  
    alert-qualifier  OCTET STRING OPTIONAL}
```

- 3) **Requested-attributes** (O): This indicates what information from the selected entries is to be included with the auto-alert. See 8.1.4.

The absence of this parameter implies that only the **alert-registration-identifier** will be present in the **alert-argument**.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10021-5:1990

Section four - Procedures for Message Store and port realization

13 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 14. The provision of the MS abstract-service is described in clause 15. The port realization in the form of Service Elements is described in clause 16.

The performance of the abstract-operations described in clauses 14 and 15 shall be subject to the requirements of the security-policy, (if one is in force), which applies to the MTS abstract-services and to the MS abstract-services.

14 Consumption of the Message Transfer abstract-service

This clause specifies how an MS shall consume the MTS abstract-service which is defined in clause 8 of ISO/IEC 10021-4. Covered are its consumption of the MTS Delivery, Submission, and Administration Ports.

14.1 Consumption of the Delivery Port abstract-services

This clause covers the performance of the MessageDelivery and ReportDelivery abstract-operations, and the invocation of the DeliveryControl abstract-operation. The MS consumption of the Delivery Port abstract-services assumes that an abstract-association exists between the Delivery Port supplier (the MTA) 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.

14.1.1 Performance of the MessageDelivery abstract-operation

When the MS receives a MessageDelivery abstract-operation from the MTA, it performs the following steps:

- 1) Returns a MessageDelivery result to the MTA to indicate that the delivery was successful. The MessageDelivery 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; for more details see 8.5.7 and 8.3.1.1.2.2 of ISO/IEC 10021-4.
- 2) The next step is to examine if any auto-actions are activated. The auto-actions are partly content-specific and are therefore also described in the content-specific International Standards. The content-specific description shall contain rules about the order in which the auto-actions are to be performed. The performance of auto-actions may result in alerts, submissions, new entries being created and in the possible deletion of the delivered-message or other messages from the MS. See 12.1
 - a) If auto-forwarding criteria are registered, by the Register-MS abstract-operation, the new entry is matched against the criteria specified. The matching is made sequentially for each specified set of selection criteria. For every "hit" a new message is generated and submitted from the MS to the MTA using the MessageSubmission abstract-operation, see 15.2.1.

The rules for how to construct the new forwarded message are again content-specific and hence described in the respective content-specific International Standards. Other content-specific events may also be performed at this stage (e.g. suppression of looping of auto-forwarded messages and the issuing of a non-receipt-notification as described for IPMS in 19.4 of ISO/IEC 10021-7. Depending on the argument-values of the Register-MS abstract-operation for auto-forwarding, a copy of the delivered-message may be retained in the MS. If the auto-forwarding attempt is unsuccessful, a copy is always retained, to prevent messages from getting lost.

NOTE - The handling of a result or error from such a submission is a local matter.

- b) If auto-alert-registrations have been made via the Register-MS abstract-operation, the new entry is matched against the filter of each registration specified. The matching is made sequentially for each registration. If a "hit" is found, an attempt is made to invoke an Alert abstract-operation from the MS to the UA. This can only be done if there is an existing abstract-association between the MS and the UA. If no abstract-association exists, the MS may have other local or non-standardized means to invoke an alert. When attempts have been made to alert all of the addresses registered for the first matching registration parameter, and at least one of the alerts succeeded, the Alert auto-action has successfully completed, and no further alert registrations are processed. If there was no path found to give the alert, the MS sets the alert-flag, which is reported to the UA when an abstract-association is next time initiated by the UA to the MS.

NOTE - If the delivered-message was deleted as a result of an auto-forwarding action in a), the auto-alert is obviously not performed.

- 3) Only after the above steps have been performed, is the new entry made visible outside the MS over the Retrieval Port. If the delivered-message was deleted as result of an auto-action, any sequence-number which was allocated in step 2) is not re-used (in order not to conflict with ISO/IEC logging extensions). The entry-status of the entry is set to new.

14.1.1.1 Generation rules for general-attributes

Optional attributes are only generated if implemented by the MS and subscribed to by the user. The generated attributes form a new entry (in some cases a parent-entry and child-entries, see clause 6) in the MS.

Refer to Table 1 and 11.3 for the rules on how the general-attributes are generated. Note that for general-attributes which are absent in the corresponding Deliver-envelope, an attribute with the default value is generated in the entry.

14.1.2 Performance of the ReportDelivery abstract-operation

When the MS receives a ReportDelivery abstract-operation from the MTA, it performs the following steps:

- 1) Returns a ReportDelivery result to the MTA to indicate that the delivery was successful. The ReportDelivery result has no parameters. For details see 8.3.1.2.2 of ISO/IEC 10021-4.
- 2) Next, if any auto-actions or other internal procedures are activated, they are performed. These are content-specific and described in the respective content-specific International Standards.

14.1.2.1 Generation rules for general-attributes

Attributes may be generated either when a message is received or when an abstract-operation is performed in the MS, triggered by an invocation from the UA.

All mandatory attributes, see Table 1 are generated. Optional attributes are only generated if implemented by the MS and subscribed to by the user. The generated attributes form a new entry (in some cases a parent-entry and child-entries, see clause 6) in the MS. The following kinds of general-attributes may be produced as part of the process:

- a) general-attributes generated by the MS itself (e.g. sequence-number);
- b) general-attributes generated from the report-delivery-envelope components. For components which are not present, but for which default values are defined, a general-attribute containing the default value is generated.

The generation rules for a) and b) are described in 14.1.1.1. The generation rules for content-specific attributes are described in the respective content-specific International Standards e.g. the IPMS-specific attributes are described in annex C of ISO/IEC 10021-7.

See table 1 and 11.3 for the rules on how the general-attributes are generated. Note that for general-attributes which are absent in the corresponding Report-envelope, an attribute with the default value is generated in the entry.

14.1.3 Invocation of the DeliveryControl abstract-operation

If the MS wants to temporarily stop the MTA from passing messages and reports, or to alter the maximum-content-length or lowest-priority of messages and reports from the MTA, it performs the following steps:

- 1) It invokes a DeliveryControl abstract-operation, containing the parameters to be changed. For details see 8.3.1.3 of ISO/IEC 10021-4.
- 2) It gets a result back when the MTS abstract-service has accepted the changes. The result contains information about whether messages and/or reports are waiting in the MTA, due to the current restrictions. For details see 8.3.1.3.2 of ISO/IEC 10021-4.
- 3) When the MS is able to accept any waiting messages and/or reports again, it should invoke a new DeliveryControl abstract-operation to relax the restrictions. The effects of a DeliveryControl abstract-operation are cancelled when either a new DeliveryControl abstract-operation alters the restrictions or when the abstract-association is released.

14.2 Consumption of the Submission Port abstract-services

This clause covers the invocation of the MessageSubmission, ProbeSubmission, and CancelDeferredDelivery abstract-operations, and the consumption of the SubmissionControl abstract-operation. The MS abstract-service consumption of the Submission Port abstract-services assumes that an abstract-association exists between the Submission Port supplier (the MTA) 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.

14.2.1 Invocation of the MessageSubmission abstract-operation

The initiation of a MessageSubmission abstract-association can be either from an auto-action within the MS or because the UA invoked a MessageSubmission abstract-operation to the MS. In order to submit the message to the MTA the MS performs the following steps:

- 1) If the MessageSubmission argument does not contain the forwarding-request extension, see 6.6, it invokes a MessageSubmission abstract-operation, containing the message to be submitted and its associated parameters. For details see 8.2.1.1 of ISO/IEC 10021-4. Otherwise, checks to see that the entry is a delivered-message and incorporates information from one delivered-message entry in the stored-messages information-base, and then invokes the MessageSubmission abstract-operation with the new content. Forwarding of entries that are not delivered-messages may be the subject of future standardization.

Note that although this forwarding-request is generic, it is not necessarily meaningful for all content-types. Where it is meaningful, the content-type of the referenced delivered-message entry shall be appropriate for incorporation into the content argument.

- 2) It gets a MessageSubmission result back when the MTA has accepted the submission. The MessageSubmission result contains among others information about identification of and submission-time for the submitted message. For details see 8.2.1.1.2 of ISO/IEC 10021-4.
- 3) If the MessageSubmission abstract-operation was triggered by a corresponding MessageSubmission abstract-operation to the MS from the UA, the result of the abstract-operation is passed back to the UA in the form of a MessageSubmission result issued by the MS. This behaviour guarantees that the message has actually been accepted by the MTA before the result is given back to the UA.
- 4) If the MTA has not accepted the message submission due to problems such as an invalid sequence number or inappropriate content-type, the MS will generate an error of InconsistentRequest. Note, all errors generated by the MTA are relayed through to the UA.
- 5) If a security-policy is in force, then to ensure that such a security-policy is not violated during message submission, the message-security-label is checked against the security-context by the MS. If the message submission is barred either by the security-policy or by temporary security restrictions, a security-error shall be indicated.

14.2.2 Invocation of the ProbeSubmission abstract-operation

A ProbeSubmission abstract-operation is initiated because the UA invoked a ProbeSubmission abstract-operation to the MS. In order to submit the probe to the MTA, the MS performs the following steps:

- 1) It invokes a ProbeSubmission abstract-operation, containing the probe to be submitted and its associated parameters. For details see 8.2.1.2.1 of ISO/IEC 10021-4.
- 2) It gets a ProbeSubmission result back when the MTA has accepted the submission. The result contains among others information about identification of and submission-time for the submitted probe. For details see 8.2.1.2.2 of ISO/IEC 10021-4.
- 3) The result of the abstract-operation is passed back to the UA in the form of a ProbeSubmission result issued by the MS. This behaviour guarantees that the probe has actually got accepted by the MTA before the result is given back to the UA.
- 4) If a security-policy is in force, then to ensure that such a security-policy is not violated during Probe submission, the message-security-label of the Probe is checked against the security-context by the

MS. If the Probe submission is barred either by the security-policy or by temporary security restrictions, a Probe submission error is generated.

14.2.3 Invocation of the CancelDeferredDelivery abstract-operation

A CancelDeferredDelivery abstract-operation is initiated because the UA invoked a CancelDeferredDelivery abstract-operation to the MS. In order to send the cancel to the MTA the MS performs the following steps:

- 1) It invokes a CancelDeferredDelivery abstract-operation, containing the cancel to be submitted and its associated parameters. For details see 8.2.1.3.1 of ISO/IEC 10021-4.
- 2) It gets a result back when the MTA has accepted the cancel. The result returned is empty as an indication of success.
- 3) The result of the abstract-operation is passed back to the UA in the form of a CancelDeferredDelivery result issued by the MS. This behaviour guarantees that the probe has actually got accepted (or not) by the MTA before the result is given back to the UA.

14.2.4 Performance of the SubmissionControl abstract-operation

If the MTA wants to temporarily stop the MS from submitting messages or probes, or to alter the maximum-content-length or lowest-priority of messages from the MS it invokes a SubmissionControl abstract-operation, for details see 8.2.1.4.1 of ISO/IEC 10021-4, to the MS. The MS reacts with the following steps:

- 1) It invokes a corresponding SubmissionControl abstract-operation from the MS to the UA.
- 2) It waits for the UA to send back a SubmissionControl result which contains information about whether messages or probes are waiting in the UA, due to the current restrictions. For details see 8.2.1.4.2 of ISO/IEC 10021-4.
- 3) The MS sends back a SubmissionControl result to the MTA, containing information from the UA.
- 4) When the MTS is able to accept any messages or probes again it should invoke a new SubmissionControl abstract-operation to relax the restrictions. The effects of a SubmissionControl abstract-operation are cancelled when either a new SubmissionControl abstract-operation alters the restrictions or when the abstract-association is released. The MS then invokes a corresponding SubmissionControl abstract-operation to the UA and waits for the SubmissionControl result.

14.3 Consumption of the Administration Port abstract-services

This clause covers the performance of the Register and ChangeCredentials abstract-operations. The consumption of the Administration Port abstract-services assumes that an abstract-association exists between the Administration Port supplier (the MTA) 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.

14.3.1 Invocation of the Register abstract-operation

A Register abstract-operation is initiated because the UA invoked a Register abstract-operation to the MS. In order to send the registration to the MTA the MS performs the following steps:

- 1) It invokes a Register abstract-operation, containing the new data to be registered. For details see 8.4.1.1.1 of ISO/IEC 10021-4.
- 2) It gets a result back when the MTA has accepted the registration. The result returned is empty as an indication of success.
- 3) The scope of the permitted changes by the UA via the MS to the user-security-label arguments shall be confined to the security-policy in force.

14.3.2 Invocation of the ChangeCredentials abstract-operation

A ChangeCredentials abstract-operation is initiated because the UA invoked a ChangeCredentials abstract-operation to the MS. In order to relay the new credentials to the MTA from the UA, the MS performs the following steps:

- 1) It invokes a ChangeCredentials abstract-operation on the MTA, containing the new credentials to be registered. For details see 8.4.1.2.1 of ISO/IEC 10021-4.
- 2) It gets a ChangeCredentials result back when the MTA has accepted the change and stores the new credentials. The ChangeCredentials result or resultant error from the MTA is relayed to the UA and is empty as an indication of success.

14.3.3 Performance of the ChangeCredentials abstract-operation

When the MS receives a ChangeCredentials abstract-operation and its associated arguments from the MTA, it performs the following steps:

- 1) It establishes that the argument information is valid for a ChangeCredentials abstract-operation. For details see 8.4.1.2 of ISO/IEC 10021-4.
- 2) It checks if there is already an existing abstract-association between the MS and the UA. If an abstract-association between the MS and the UA does not exist, the MTA is informed by an error that change of credentials can not take place at present and no further steps are taken.
- 3) If the abstract-association between the MS and the UA exists, the MS invokes a ChangeCredentials abstract-operation to the UA.
- 4) If the UA sends back an empty ChangeCredentials result, indicating success, the MS sends back a corresponding ChangeCredentials result indicating success to the MTA and stores the credentials. If the UA returns an error, this is relayed to the MTA to indicate that error. Note, that the MS never sends back an indication of success to the MTA until it has received the corresponding result back from the UA first.

15 Supply of the Message Store abstract-service

This clause specifies how a MS supplies the MS abstract-service. Covered are its supply of the Retrieval, Indirect-submission, and Administration Ports.

15.1 Supply of the Retrieval Port abstract-services

This clause covers the supply of the Summarize, List, Fetch, Delete, Register-MS, 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 UA). The performance of the abstract-operations is in sequential order, no parallel processing takes place. Not all error cases are described.

15.1.1 Performance of the Summarize abstract-operation

When the MS receives a Summarize abstract-operation from the UA, it performs the following steps:

- 1) Establishes which information-base the Summarize abstract-operation addresses.
- 2) Checks if there are any entries in the information-base. If it is empty, a Summarize result with zero length is returned and no further steps are performed.
- 3) Checks that the supplied argument general-attributes and any content-specific attributes recognised by the MS are valid for a Summarize abstract-operation. For details see 8.2.1.
- 4) Accumulates counts in accordance with the supplied argument general-attributes and any content-specific attributes recognised by the MS.
- 5) Returns the Summarize result to the UA. For details see 8.2.2.
- 6) If a security-policy is in force, then to ensure that such a security-policy is not violated during the Summarize abstract-operation the security classification of the security label is checked against the security-context by the MS. If a summarize is barred by the security-policy the Summarize abstract-operation shall be abandoned and a security error shall be indicated.

15.1.2 Performance of the List abstract-operation

When the MS receives a List abstract-operation from the UA, it performs the following steps:

- 1) Establishes which information-base the List abstract-operation addresses.
- 2) Checks that the supplied argument general-attributes and any content-specific attributes recognised by the MS are valid for a List abstract-operation. For details see 8.3.1.
- 3) Identifies zero or more entries as requested in the argument of the abstract-operation, up to any supplied limit. Child-entries to a parent-entry are excluded, unless explicitly selected in the argument.
- 4) If a set of requested general-attributes has been specified as arguments in the abstract-operation, these general-attributes are returned, if present, to the UA for each selected entry. If no request has been done, the default List abstract-operation values, as specified with a previous Register-MS abstract-operation, are returned, if present. For more details see 8.3.2. The entry-status of each selected message is set to listed.

- 5) 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 list 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.

15.1.3 Performance of the Fetch abstract-operation

When the MS receives a Fetch abstract-operation from the UA, it performs the following steps:

- 1) Establishes which information-base the Fetch abstract-operation addresses.
- 2) Checks that the supplied argument general-attributes and any content-specific attributes recognised by the MS are valid for a Fetch abstract-operation. For details see 8.4.1.
- 3) Identifies zero or more entries as requested in the argument of the abstract-operation, up to any supplied limit. Child-entries to a parent-entry are excluded, unless explicitly selected in the argument.
- 4) If a set of requested general-attributes have been specified as arguments in the abstract-operation, these general-attributes are returned, if present, to the UA for the first selected entry. If no request has been done, the default Fetch abstract-operation values, as specified with a previous Register-MS abstract-operation, are returned, if present. If several entries that match the search criteria are found, the sequence-numbers for the second and following entries are returned in increasing order. If there were more matching entries than in the specified limit, the next sequence-number beyond the limit is also returned. For more details see 8.4.2.
- 5) 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 Fetch abstract-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.

15.1.4 Performance of the Delete abstract-operation

When the MS receives a Delete abstract-operation from the UA, it performs the following steps:

- 1) Establishes which information-base the Delete abstract-operation addresses.
- 2) Checks that the supplied arguments are valid for a Delete abstract-operation. For details see 8.5.1.
- 3) Identifies the entry or list of entries requested in the argument of the abstract-operation.
- 4) If any of the entries has delete restrictions, see 8.5, none of the deletions takes place. Otherwise all deletions are performed and an empty Delete result returned to the UA as indication of success.

15.1.5 Performance of the Register-MS abstract-operation

When the MS receives a Register-MS abstract-operation from the UA, it performs the following steps:

- 1) Checks that the supplied arguments are valid for a Register-MS abstract-operation. For details see 8.6.1.
- 2) Replaces any old parameters with the corresponding new ones. Auto-actions have effect on transactions, such as message-deliveries and report-deliveries, that occur after the initiation or

deletion of auto-action requests, there is no processing of entries that already reside in the MS at that point in time.

- 3) Sends back an empty Register-MS result to the UA to indicate that the abstract-operation has been performed successfully.
- 4) If a security-policy is in force then the Register-MS abstract-operations 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.

15.1.6 Invocation of the Alert abstract-operation

The invocation of the Alert abstract-operation is as a result of the consumption of the Delivery Port abstract-service (see 14.1.1).

If the Auto-alert auto-action is initiated by the UA, by an earlier Register-MS abstract-operation, the MS abstract-service performs the following steps:

- 1) Checks if an abstract-association exists. If not, the MS will never establish an abstract-association, and no Alert abstract-operation can be invoked.
- 2) If an abstract-association exists, the MS invokes an abstract-operation containing the relevant argument information, for details see 8.7.1, and waits for a empty Alert result to be returned by the UA as an indication of success.
- 3) If an abstract-association does not exist, there is a possibility to use a non-standardized protocol to inform the user. The alert signal in this case may be given on the user's terminal, but can alternatively be given on a telephone, a beeper or any other suitable terminal equipment associated with the user. The latter method can also be used in cases where the Alert abstract-operation has not been implemented.
- 4) 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.

15.2 Supply of the Indirect-submission Port abstract-services

This clause covers the performance of the MessageSubmission, ProbeSubmission, and CancelDeferredDelivery abstract-operations, and the invocation of the SubmissionControl abstract-operation. The MS abstract-service supply of the Indirect-submission Port abstract-services assumes that an abstract-association exists between the Indirect-submission Port supplier (the MS) and the Indirect-submission Port consumer (the UA). The performance of the abstract-operations is in sequential order, no parallel processing takes place. Not all error cases are described.

15.2.1 Performance of the MessageSubmission abstract-operation

When the MS receives a MessageSubmission abstract-operation and its associated arguments from the UA, it performs the following steps:

- 1) It establishes that the argument information is valid for a MessageSubmission abstract-operation. for details see 8.2.1.1.1 of ISO/IEC 10021-4.

- 2) It checks the arguments to establish if the message content was supplied by the UA or if it has to be inserted by the MS (i.e if the forwarding-request extension is present). In the latter case, if the entry is a delivered-message entry, the corresponding message is inserted and the MS-related arguments deleted. Forwarding of entries that are not delivered-messages may be the subject of future standardization.
- 3) It checks if there is already an existing abstract-association between the MS and the MTA. If not, the MS initiates such an abstract-association. If an abstract-association cannot be established, the UA is informed by an error that submission can not take place at present and no further steps are processed.
- 4) If the abstract-association between the MS and the MTA exists, the MS invokes a MessageSubmission abstract-operation to the MTA, after any modifications mentioned in step 2).
- 5) If the MTA sends back a MessageSubmission result, for details see 8.2.1.1.2 of ISO/IEC 10021-4, indicating success, the MS sends back a corresponding MessageSubmission result indicating success to the UA. Note, that the MS never sends back an indication of success to the UA until it has received the corresponding result back from the MTA first. This is to insure a consistent service from a user point of view, viz that a submission always means that the responsibility for the message has been taken over by the MTA when the result comes back.
- 6) The MS may either choose to terminate the abstract-association with the MTA after a certain period of inactivity, or when the UA terminates its corresponding abstract-association with the MS.

15.2.2 Performance of the ProbeSubmission abstract-operation

When the MS receives a ProbeSubmission abstract-operation and its associated arguments from the UA, it performs the following steps:

- 1) It establishes that the argument information is valid for a ProbeSubmission abstract-operation. For details see 8.2.1.2.1 of ISO/IEC 10021-4.
- 2) It checks if there is already an existing abstract-association between the MS and the MTA. If not, the MS initiates such an abstract-association. If an abstract-association cannot be established, the UA is informed by an error that submission can not take place at present and no further steps are processed.
- 3) If the abstract-association between the MS and the MTA exists, the MS invokes a ProbeSubmission abstract-operation to the MTA.
- 4) If the MTA sends back a ProbeSubmission result, for details see 8.2.1.2.2 of ISO/IEC 10021-4, indicating success, the MS sends back a corresponding ProbeSubmission result indicating success to the UA. Note, that the MS never sends back an indication of success to the UA until it has received the corresponding result back from the MTA first. This is to insure a consistent service from a user point of view, viz that a submission always means that the responsibility for the probe has been taken over by the MTS when the result comes back.
- 5) The MS may either choose to terminate the abstract-association with the MTA after a certain period of inactivity, or when the UA terminates its corresponding abstract-association with the MS.

15.2.3 Performance of the CancelDeferredDelivery abstract-operation

When the MS receives a CancelDeferredDelivery abstract-operation and its associated arguments from the UA, it performs the following steps:

- 1) It establishes that the argument information is valid for a CancelDeferredDelivery abstract-operation. For details see 8.2.1.3.1 of ISO/IEC 10021-4.
- 2) It checks if there is already an existing abstract-association between the MS and the MTA. If not, the MS initiates such an abstract-association. If an abstract-association cannot be established, the UA is informed by an error that CancelDeferredDelivery can not take place at present and no further steps are processed.
- 3) If the abstract-association between the MS and the MTA exists, the MS invokes a CancelDeferredDelivery abstract-operation to the MTA.
- 4) If the MTA sends back a CancelDeferredDelivery result, for details see 8.2.1.3.2 of ISO/IEC 10021-4, indicating success, the MS sends back a corresponding CancelDeferredDelivery result indicating success to the UA. Note, that the MS never sends back an indication of success to the UA until it has received the corresponding result back from the MTA first. This is to insure a consistent service from a user point of view, viz that the responsibility for the cancel deferred delivery has been taken over by the MTS, when the result comes back.
- 5) The MS may either choose to terminate the abstract-association with the MTA after a certain period of inactivity, or when the UA terminates its corresponding abstract-association with the MS.

15.2.4 Invocation of the SubmissionControl abstract-operation

If the MS receives a SubmissionControl abstract-operation from the MTA, or if the MS for some internal reasons wants to temporarily stop the UA from submitting messages or probes, or to alter the maximum-length or lowest-priority of messages from the UA, the MS performs the following steps:

- 1) It invokes a SubmissionControl abstract-operation to the UA. For details see 8.2.1.4.1 of ISO/IEC 10021-4.
- 2) It waits for a SubmissionControl result, for details see 8.2.1.4.2 of ISO/IEC 10021-4, from the UA confirming the acceptance of the SubmissionControl abstract-operation.
- 3) If the SubmissionControl abstract-operation had been triggered by a corresponding abstract-operation from the MTA to the MS the SubmissionControl result from the UA is passed on from the MS to the MTA and the MS waits for the SubmissionControl result to come back from the UA.

15.3 Supply of the Administration Port abstract-services

This clause covers the performance of the Register and ChangeCredentials abstract-operations. The MS abstract-service supply of the Administration Port abstract-services assumes that an abstract-association exists between the Indirect-submission Port supplier (the MS) and the Indirect-submission Port consumer (the UA). The performance of the abstract-operations is in sequential order, no parallel processing takes place. Not all error cases are described.

15.3.1 Performance of the Register abstract-operation

When the MS receives a Register abstract-operation and its associated arguments from the UA, it performs the following steps:

- 1) It establishes that the argument information is valid for a Register abstract-operation. For details see 8.4.1.1.1 of ISO/IEC 10021-4.

- 2) It checks if there is already an existing abstract-association between the MS and the MTA. If not, the MS initiates such an abstract-association. If an abstract-association cannot be established, the UA is informed by an error that register can not take place at present and no further steps are processed.
- 3) If the abstract-association between the MS and the MTA exists, the MS invokes a Register abstract-operation to the MTA.
- 4) If the MTA sends back a Register result, for details see 8.4.1.1.2 of ISO/IEC 10021-4, indicating success, the MS sends back a corresponding Register result indicating success to the UA. Note, that the MS never sends back an indication of success to the UA until it has received the corresponding result back from the MTA first. This is to insure a consistent service from a user point of view, viz that the responsibility for the register has been taken over by the MTS, when the result comes back.
- 5) The MS may either choose to terminate the abstract-association with the MTA after a certain period of inactivity, or when the UA terminates its corresponding abstract-association with the MS.
- 6) The scope of permitted changes by the UA via the MS to the user-security-labels shall be confined by the security-policy in force. Some security-policies may only permit user-security-labels to be 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.

15.3.2 Invocation of the ChangeCredentials abstract-operation

A ChangeCredentials abstract-operation is initiated because the MTA invoked a ChangeCredentials abstract-operation to the MS. In order to relay the new-credentials to the UA, from the MTA, the MS performs the following steps:

- 1) It establishes that the argument information is valid for a ChangeCredentials abstract-operation. For details see 8.4.1.2 of ISO/IEC 10021-4. If the old credentials are incorrect and the new credentials are not acceptable an error is returned and no further processing takes place.
- 2) It invokes a ChangeCredentials abstract-operation on the UA, containing the new credentials to be registered. For details see 8.4.1.2 of ISO/IEC 10021-4.
- 3) It gets a ChangeCredentials result back when the UA has accepted the change and stores the new credentials. The ChangeCredentials result or resultant error from the UA is relayed to the MTA.

15.3.3 Performance of the ChangeCredentials abstract-operation

When the MS receives a ChangeCredentials abstract-operation and its associated arguments from the UA, it performs the following steps:

- 1) It establishes that the argument information is valid for a ChangeCredentials abstract-operation. For details see 8.4.1.2 of ISO/IEC 10021-4.
- 2) It checks if there is already an existing abstract-association between the MS and the MTA. If not, the MS initiates such an abstract-association. If an abstract-association cannot be established, the UA is informed by an error that change of credentials can not take place at present and no further steps are processed.
- 3) If the abstract-association between the MS and the MTA exists, the MS invokes a ChangeCredentials abstract-operation to the MTA.

- 4) If the MTA sends back an empty ChangeCredentials result, indicating success, the MS sends back a corresponding ChangeCredentials result indicating success to the UA and stores the credentials. If the MTA returns an error, this is relayed to the UA to indicate that error. Note, that the MS never sends back an indication of success to the UA until it has received the corresponding result back from the MTA first.
- 5) The MS may either choose to terminate the abstract-association with the MTA after a certain period of inactivity, or when the UA terminates its corresponding abstract-association with the MS.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10021-5:1990

16 Ports realization

This clause describes how the Retrieval, the 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 ISO/IEC 10021-4.

16.1 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 (MRSE) which is documented in ISO/IEC 10021-6.

16.2 Indirect-submission Port

The Indirect-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) which is documented in ISO/IEC 10021-6.

16.3 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 (MASE) which is documented in ISO/IEC 10021-6.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10021-5:1990

Annex A

(normative)

Formal assignment of object identifiers

All OBJECT IDENTIFIERS this part of ISO/IEC 10021 assigns 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 part of ISO/IEC 10021. 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.

MObjectIdentifiers {joint-iso-ccitt mhs-motis(6) ms(4) modules(0) object-identifiers(0)}

DEFINITIONS :: =

BEGIN

-- Prologue

-- Exports everything

IMPORTS

ID, id-ms

FROM MObjectIdentifiers {joint-iso-ccitt mhs-motis(6) arch(5) modules(0) object-identifiers(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}

-- 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

-- Objects

id-ot-ms	ID :: = {id-ot 0}
id-ot-ms-user	ID :: = {id-ot 1}

-- Port-types

id-pt-retrieval	ID :: = {id-pt 0}
-----------------	-------------------

-- *Attribute-types*

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-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-entry-status	ID :: = {id-att 15}
id-att-entry-type	ID :: = {id-att 16}
id-att-intended-recipient-name	ID :: = {id-att 17}
id-att-message-delivery-envelope	ID :: = {id-att 18}
id-att-message-delivery-identifier	ID :: = {id-att 19}
id-att-message-delivery-time	ID :: = {id-att 20}
id-att-message-origin-authentication-check	ID :: = {id-att 21}
id-att-message-security-label	ID :: = {id-att 22}
id-att-message-submission-time	ID :: = {id-att 23}
id-att-message-token	ID :: = {id-att 24}
id-att-original-EITs	ID :: = {id-att 25}
id-att-originator-certificate	ID :: = {id-att 26}
id-att-originator-name	ID :: = {id-att 27}
id-att-other-recipient-names	ID :: = {id-att 28}
id-att-parent-sequence-number	ID :: = {id-att 29}
id-att-per-recipient-report-delivery-fields	ID :: = {id-att 30}
id-att-priority	ID :: = {id-att 31}
id-att-proof-of-delivery-request	ID :: = {id-att 32}
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-security-classification	ID :: = {id-att 38}
id-att-sequence-number	ID :: = {id-att 39}
id-att-subject-submission-identifier	ID :: = {id-att 40}
id-att-this-recipient-name	ID :: = {id-att 41}

-- *Auto-action-types*

id-act-auto-forward	ID :: = {id-act 0}
id-act-auto-alert	ID :: = {id-act 1}

END -- of MSObjectIdentifiers