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

*Technologies de l'information — Systèmes de messagerie (MHS): Mémoire
de messages: Définition du service abstrait*

Reference number
ISO/IEC 10021-5:1999(E)



PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 1999

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Published by ISO in 2000

Printed in Switzerland

CONTENTS

	<i>Page</i>
SECTION 1 – GENERAL	1
1 Scope	1
2 Normative references	1
2.1 Reference Model references	2
2.2 Presentation references	2
2.3 Remote Operations references	2
2.4 Directory references	2
2.5 Message Handling references	2
3 Definitions	2
3.1 Common Definitions for MHS	2
3.2 Message Store Definitions	3
4 Abbreviations	6
5 Conventions	6
5.1 Conventions for abstract-services	6
5.2 Conventions for attribute-types used in Tables 2 and 3 of clause 11	7
5.3 Conventions for attribute-types used in Table 4 of clause 11	7
5.4 General font conventions	7
5.5 Font conventions for ASN.1 definitions	7
5.6 Rules for ASN.1 definitions	8
5.7 Conventions for previous editions of this Service Definition	8
5.8 ASN.1 Packed Encoding Rules	8
5.9 Interpretation of UTC Time values	8
SECTION 2 – MESSAGE STORE ABSTRACT-SERVICE DEFINITION	8
6 Message Store model	8
6.1 Message Store objects and contracts	9
6.2 Message Store ports	10
6.2.1 Retrieval Ports	10
6.2.2 MS-submission Ports	11
6.2.3 Administration Ports	11
6.3 Information model	11
6.3.1 Entry-classes	11
6.3.2 Entries	12
6.3.3 Attributes	12
6.3.3.1 Attribute-type	13
6.3.3.2 Attribute-values	13
6.3.3.3 The ATTRIBUTE information object class	13
6.3.4 Main-entries, parent-entries, and child-entries	14
6.3.5 Content-specific Attributes	14
6.3.6 Entry-types	15
6.3.7 Organization of entry-classes	15
6.3.7.1 The Stored-message entry-class	16
6.3.7.2 The Message-log entry-class	16
6.3.7.3 The Auto-action-log entry-class	17
6.3.7.4 Entry-class support	17
6.3.8 Retrieval-status	17

6.3.9	Matching-rules	18
6.3.9.1	Generic Matching Rules.....	18
6.3.9.2	Matching Rule definition	19
6.3.9.3	The MATCHING-RULE information object class.....	19
6.4	Message grouping.....	19
6.5	Auto-actions.....	20
6.5.1	The AUTO-ACTION information object class	20
6.5.2	Auto-action registration	21
6.5.3	Auto-action errors.....	21
6.5.4	Auto-action execution.....	21
6.6	MS extensions.....	22
7	MS-bind and MS-unbind operations.....	22
7.1	MS-bind abstract-operation.....	22
7.1.1	MS-bind-argument.....	22
7.1.2	MS-bind-result	24
7.1.3	MS-bind-error	26
7.2	MS-unbind abstract-operation.....	27
8	Abstract-operations.....	27
8.1	Common data-types used in abstract-operations	27
8.1.1	Range.....	27
8.1.2	Filters.....	28
8.1.2.1	Filter	28
8.1.2.2	Filter-item	28
8.1.2.3	Attribute-value-assertion.....	29
8.1.3	Selector	29
8.1.4	Entry-information-selection	30
8.1.5	Entry-information	31
8.1.6	MS-submission-options	31
8.1.7	Common-submission-results.....	33
8.2	Retrieval Port abstract-operations	33
8.2.1	Summarize abstract-operation	34
8.2.1.1	Summarize-argument.....	34
8.2.1.2	Summarize-result.....	34
8.2.1.3	Summarize abstract-errors	35
8.2.2	List abstract-operation	35
8.2.2.1	List-argument	35
8.2.2.2	List-result.....	36
8.2.2.3	List abstract-errors.....	36
8.2.3	Fetch abstract-operation.....	36
8.2.3.1	Fetch-argument.....	36
8.2.3.2	Fetch-result	37
8.2.3.3	Fetch abstract-errors	37
8.2.4	Delete abstract-operation	37
8.2.4.1	Delete-argument.....	38
8.2.4.2	Delete-result.....	38
8.2.4.3	Delete abstract-errors.....	38
8.2.5	Register-MS abstract-operation.....	38
8.2.5.1	Register-MS-argument.....	39
8.2.5.2	Register-MS-result.....	43
8.2.5.3	Register-MS abstract-errors	43
8.2.6	Alert abstract-operation	43
8.2.6.1	Alert-argument	44
8.2.6.2	Alert-result	44
8.2.6.3	Alert abstract-errors.....	44
8.2.7	Modify abstract-operation.....	44
8.2.7.1	Modify-argument.....	44
8.2.7.2	Modify-result.....	46
8.2.7.3	Modify Abstract-errors	46

8.3	MS-submission Port abstract-operations	46
8.3.1	MS-message-submission abstract-operation	46
8.3.1.1	MS-message-submission-argument	47
8.3.1.2	MS-message-submission-result	47
8.3.1.3	MS-message-submission Abstract-errors	47
8.3.2	MS-probe-submission abstract-operation	48
8.3.2.1	MS-probe-submission-argument	48
8.3.2.2	MS-probe-submission-result	48
8.3.2.3	MS-probe-submission Abstract-errors	48
8.3.3	MS-cancel-deferred-delivery abstract-operation	48
8.3.4	MS-submission-control abstract-operation	49
9	Abstract-errors	49
9.1	Error precedence	49
9.2	Attribute-error	49
9.3	Auto-action-request-error	50
9.4	Delete-error	50
9.5	Fetch-restriction-error	51
9.6	Invalid-parameters-error	51
9.7	Range-error	52
9.8	Security-error	52
9.8.1	Security-error for Register-MS abstract-operation	52
9.8.2	Security-error for Delete, Fetch, List, Modify and Summarize abstract-operations	52
9.8.3	Security-error for MS-probe-submission and MS-message-submission abstract-operations	52
9.9	Sequence-number-error	53
9.10	Service-error	53
9.11	Message-group-error	53
9.12	MS-extension-error	54
9.13	Register-MS-error	54
9.14	Old-credentials-incorrectly-specified	55
9.15	New-credentials-unacceptable	55
9.16	Modify-error	55
9.17	Entry-class-error	55
SECTION 3 – GENERAL-ATTRIBUTE-TYPES, MATCHING-RULES AND AUTO-ACTION-TYPES		56
10	Overview	56
11	General-attribute-types	56
11.1	General-attribute-types overview	56
11.1.1	MS support requirements for general-attribute-types	59
11.1.2	MS-user support requirements for general-attribute-types	59
11.2	Description of the general-attribute-types	59
11.2.1	AC-correlated-report-list	59
11.2.2	AC-report-subject-entry	60
11.2.3	AC-report-summary	60
11.2.4	AC-uncorrelated-report-list	61
11.2.5	Auto-action-error	61
11.2.6	Auto-action-registration-identifier	62
11.2.7	Auto-action-subject-entry	62
11.2.8	Auto-action-type	62
11.2.9	Certificate-selectors	62
11.2.10	Child-sequence-numbers	62
11.2.11	Content	62
11.2.12	Content-confidentiality-algorithm-identifier	63
11.2.13	Content-correlator	63
11.2.14	Content-identifier	63
11.2.15	Content-integrity-check	63
11.2.16	Content-length	63

11.2.17	Content-returned.....	64
11.2.18	Content-type.....	64
11.2.19	Conversion-with-loss-prohibited.....	64
11.2.20	Converted-EITs.....	64
11.2.21	Creation-time.....	64
11.2.22	Deferred-delivery-cancellation-time.....	64
11.2.23	Deferred-delivery-time.....	65
11.2.24	Deletion-time.....	65
11.2.25	Delivered-EITs.....	65
11.2.26	Delivery-flags.....	65
11.2.27	DL-exempted-recipients.....	65
11.2.28	DL-expansion-history.....	66
11.2.29	DL-expansion-prohibited.....	66
11.2.30	Entry-type.....	66
11.2.31	Internal-trace-information.....	66
11.2.32	Latest-delivery-time.....	66
11.2.33	Locally-originated.....	66
11.2.34	Marked-for-deletion.....	67
11.2.35	Message-delivery-envelope.....	67
11.2.36	Message-delivery-time.....	67
11.2.37	Message-group-name.....	67
11.2.38	Message-identifier.....	67
11.2.39	Message-notes.....	68
11.2.40	Message-origin-authentication-check.....	68
11.2.41	Message-security-label.....	68
11.2.42	Message-submission-envelope.....	68
11.2.43	Message-submission-time.....	68
11.2.44	Message-token.....	68
11.2.45	MS-originated.....	69
11.2.46	MS-submission-error.....	69
11.2.47	Multiple-originator-certificates.....	69
11.2.48	Original-EITs.....	69
11.2.49	Originally-intended-recipient-name.....	69
11.2.50	Originating-MTA-certificate.....	70
11.2.51	Originator-certificate.....	70
11.2.52	Originator-name.....	70
11.2.53	Originator-report-request.....	70
11.2.54	Originator-return-address.....	70
11.2.55	Other-recipient-names.....	70
11.2.56	Parent-sequence-number.....	71
11.2.57	Per-message-indicators.....	71
11.2.58	Per-recipient-message-submission-fields.....	71
11.2.59	Per-recipient-probe-submission-fields.....	71
11.2.60	Per-recipient-report-delivery-fields.....	71
11.2.61	Priority.....	71
11.2.62	Probe-origin-authentication-check.....	72
11.2.63	Probe-submission-envelope.....	72
11.2.64	Proof-of-delivery-request.....	72
11.2.65	Proof-of-submission.....	72
11.2.66	Recipient-certificate.....	72
11.2.67	Recipient-names.....	72
11.2.68	Recipient-reassignment-prohibited.....	73
11.2.69	Redirection-history.....	73
11.2.70	Report-delivery-envelope.....	73
11.2.71	Reporting-DL-name.....	73
11.2.72	Reporting-MTA-certificate.....	73
11.2.73	Report-origin-authentication-check.....	73
11.2.74	Retrieval-status.....	74
11.2.75	Security-classification.....	74
11.2.76	Sequence-number.....	74

11.2.77	Signature-verification-status	74
11.2.78	Storage-period	75
11.2.79	Storage-time	75
11.2.80	Subject-submission-identifier	76
11.2.81	This-recipient-name	76
11.2.82	Trace-information	76
11.3	The Attribute-table information object set	76
11.4	Generation of the general-attributes	76
11.5	Attribute-types subscription	76
11.6	General-attribute-types subject to modification	81
12	General matching-rules	81
12.1	MS-string syntax	81
12.2	String matching-rules	81
12.2.1	MS-string-match	82
12.2.2	MS-string-ordering-match	82
12.2.3	MS-substrings-match	82
12.2.4	MS-single-substring-match	83
12.2.5	MS-string-case-sensitive-match	83
12.2.6	MS-string-list-match	83
12.2.7	MS-string-list-elements-match	83
12.2.8	MS-single-substring-list-match	83
12.2.9	MS-single-substring-list-elements-match	84
12.3	Syntax-based matching-rules	84
12.4	Matching-rules for complex Message Store attributes	84
12.4.1	OR-address-match	84
12.4.2	OR-address-elements-match	85
12.4.3	OR-address-substring-elements-match	85
12.4.4	OR-name-match	86
12.4.5	OR-name-elements-match	86
12.4.6	OR-name-substring-elements-match	86
12.4.7	OR-name-single-element-match	86
12.4.8	Redirection-or-DL-expansion-match	87
12.4.9	Redirection-or-DL-expansion-elements-match	87
12.4.10	Redirection-or-DL-expansion-substring-elements-match	87
12.4.11	Redirection-or-DL-expansion-single-element-match	87
12.4.12	Redirection-reason-match	87
12.4.13	MTS-identifier-match	88
12.4.14	Content-correlator-match	88
12.4.15	Content-identifier-match	88
12.5	Matching-rule support	88
12.6	The Matching-rule-table information object set	88
13	General-auto-actions	89
13.1	Auto-alert	90
13.2	Auto-modify	91
13.3	Auto-correlate-reports	92
13.4	Auto-delete	92
SECTION 4 – PROCEDURES FOR MESSAGE STORE AND PORT REALIZATION		93
14	Overview	93
15	Consumption of the Message Transfer abstract-service	93
15.1	Consumption of the Delivery Port abstract-services	93
15.1.1	Performance of the Message-delivery abstract-operation	93
15.1.2	Performance of the Report-delivery abstract-operation	94
15.1.3	Invocation of the Delivery-control abstract-operation	94
15.1.4	Generation rules for general-attributes	95

15.2	Consumption of the Submission Port abstract-services	95
15.2.1	Invocation of the Message-submission abstract-operation	95
15.2.2	Invocation of the Probe-submission abstract-operation	95
15.2.3	Invocation of the Cancel-deferred-delivery abstract-operation	96
15.2.4	Performance of the Submission-control abstract-operation	96
15.3	Consumption of the Administration Port abstract-services	96
15.3.1	Invocation of the Register abstract-operation	96
15.3.2	Invocation of the Change-credentials abstract-operation	96
15.3.3	Performance of the Change-credentials abstract-operation	97
16	Supply of the Message Store abstract-service	97
16.1	Supply of the Retrieval Port abstract-services	97
16.1.1	Performance of the Summarize abstract-operation	97
16.1.2	Performance of the List abstract-operation	97
16.1.3	Performance of the Fetch abstract-operation	98
16.1.4	Performance of the Delete abstract-operation	98
16.1.5	Performance of the Register-MS abstract-operation	99
16.1.6	Performance of the Modify abstract-operation	99
16.1.7	Invocation of the Alert abstract-operation	100
16.2	Supply of the MS-submission Port abstract-services	100
16.2.1	Performance of the MS-message-submission abstract-operation	100
16.2.2	Performance of the MS-probe-submission abstract-operation	101
16.2.3	Performance of the MS-cancel-deferred-delivery abstract-operation	102
16.2.4	Invocation of the Submission-control abstract-operation	102
16.2.5	Generation rules for general-attributes	103
16.3	Supply of the Administration Port abstract-services	103
16.3.1	Performance of the Register abstract-operation	103
16.3.2	Invocation of the Change-credentials abstract-operation	103
16.3.3	Performance of the Change-credentials abstract-operation	103
17	Ports realization	104
17.1	Retrieval Port	104
17.2	MS-submission Port	104
17.3	Administration Port	104
Annex A	Formal assignment of Object Identifiers	105
Annex B	Formal definition of the Message Store abstract-service	108
Annex C	Formal definition of general-attribute-types	121
Annex D	Formal definition of general matching-rules	131
Annex E	Formal definition of general-auto-action-types	134
Annex F	Additional MS capabilities	136
F.1	Modify capability	136
F.2	Modify retrieval-status capability	136
F.3	Protected change credentials capability	136
Annex G	Summary of Changes to previous editions	137
G.1	Changes introduced in the 1994 edition	137
G.2	Minimum changes required for support of 1994 Application Contexts	138
Annex H	Formal definition of MS parameter upper bounds	139

.....

Annex I – Message-grouping	140
H.1 Establishing a storage framework	140
H.2 Populating the storage framework	140
H.3 Interworking considerations	140
Annex J – Example of the Summarize abstract-operation	142
I.1 The entries in the example MS	142
I.2 An example of a request for summary	142
Annex K – Differences between ITU-T Rec. X.413 (1998) and ISO/IEC 10021-5:1999	143
INDEX	144

LIST OF FIGURES

Figure 1 – Message Store abstract-service	9
Figure 2 – The Components of an entry	12

LIST OF TABLES

Table 1 – Entry-types present in entry-classes	15
Table 2 – Message Store common general-attribute-types	57
Table 3 – General-attribute-types for the Auto-action-log entry-class	59
Table 4 – Generation of the General-attribute-types	77
Table 5 – Summary of general-auto-action registration and logging capabilities	90
Table I.1 – Stored-messages in the example	142
Table I.2 – Expected result from the Summarize abstract-operation	142

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 10021 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 10021-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.413.

This fourth edition cancels and replaces the third edition (ISO/IEC 10021-5:1996), which has been technically revised. It also incorporates Amendment 1:1998 and Technical Corrigendum 1:1998.

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

- *Part 1: System and Service Overview*
- *Part 2: Overall architecture*
- *Part 3: Abstract Service Definition Conventions*
- *Part 4: Message transfer system: Abstract service definition and procedures*
- *Part 5: Message store: Abstract service definition*
- *Part 6: Protocol specifications*
- *Part 7: Interpersonal messaging system*
- *Part 8: Electronic Data Interchange Messaging Service*
- *Part 9: Electronic Data Interchange Messaging System*
- *Part 10: MHS routing*
- *Part 11: Guide for Messaging Systems Managers*

Annexes A to G form a normative part of this part of ISO/IEC 10021. Annexes H to K are for information only.

Introduction

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

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

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

This Service Definition has been produced by joint ITU-T – ISO/IEC agreement. It is published as common text as ITU-T Rec. X.413 | ISO/IEC 10021-5. Annex K lists the differences between ITU-T and ISO/IEC texts.

...

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

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

SECTION 1 – GENERAL

1 Scope

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

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

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

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

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

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

2 Normative references

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

2.1 Reference Model references

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

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

2.2 Presentation references

This Recommendation | International Standard cites the following Presentation specifications:

- ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1: 1998, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.
- ITU-T Recommendation X.681 (1997) | ISO/IEC 8824-2: 1998, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*.
- ITU-T Recommendation X.682 (1997) | ISO/IEC 8824-3: 1998, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification*.
- ITU-T Recommendation X.690 (1997) | ISO/IEC 8825-1: 1998, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER)*.

2.3 Remote Operations references

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

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

2.4 Directory references

This Recommendation | International Standard cites the following Directory specifications:

- ITU-T Recommendation X.501 (1997) | ISO/IEC 9594-2: 1998, *Information technology – Open Systems Interconnection – The Directory: Models*.
- ITU-T Recommendation X.509 (1997) | ISO/IEC 9594-8: 1998, *Information technology – Open Systems Interconnection – The Directory: Authentication framework*.
- ITU-T Recommendation X.520 (1997) | ISO/IEC 9594-6: 1998, *Information technology – Open Systems Interconnection – The Directory: Selected attribute types*.

2.5 Message Handling references

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

- ITU-T Recommendation F.400/X.400 (1999), *Message handling services: Message handling system and service overview*.
ISO/IEC 10021-1: 1999, *Information technology – Message Handling Systems (MHS) – Part 1: System and service overview*.
- ITU-T Recommendation X.402 (1999) | ISO/IEC 10021-2: 1999, *Information technology – Message Handling Systems (MHS): Overall architecture*.
- ITU-T Recommendation X.411 (1999) | ISO/IEC 10021-4: 1999, *Information technology – Message Handling Systems (MHS): Message transfer system: Abstract service definition and procedures*.
- ITU-T Recommendation X.419 (1999) | ISO/IEC 10021-6: 1999, *Information technology – Message Handling Systems (MHS): Protocol specifications*.
- ITU-T Recommendation X.420 (1999) | ISO/IEC 10021-7: 1999, *Information technology – Message Handling Systems (MHS): Interpersonal messaging system*.

3 Definitions

3.1 Common Definitions for MHS

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

3.2 Message Store Definitions

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

- 3.2.1 abstract-association:** An abstract binding between two communication partners. In this Service Definition, the binding between an MS-user and an MS for the provision of the MS abstract-service, or between an MS and the MTS for the provision of the MTS abstract-service.
- 3.2.2 Administration Port:** A port which supports the administration services (of the MTS) within the MS abstract-service.
- 3.2.3 Alert abstract-operation:** An abstract-operation which enables the MS to inform the MS-user that a message or report has been delivered to the MS. May be issued only over an existing abstract-association.
- 3.2.4 attribute:** The information of a particular type appearing in an entry.
- 3.2.5 attribute-type:** That component of an attribute which indicates the class of information given by that attribute.
- 3.2.6 attribute-value:** A particular instance of the class of information indicated by an attribute type.
- 3.2.7 attribute-value-assertion:** A proposition, which may be *true*, *false*, or *undefined*, concerning the values of an attribute in an entry.
- 3.2.8 auto-action:** Actions that are performed automatically by the MS according to instructions previously registered by the MS-user.
- 3.2.9 Auto-action-log:** An entry-class which contains entries that record the performance of certain auto-actions by the MS.
- 3.2.10 auto-action-event:** An entry of the Auto-action-log entry-class which represents an auto-action execution.
- 3.2.11 auto-action-type:** The type of an auto-action, e.g., Auto-alert.
- 3.2.12 Auto-alert:** An auto-action which alerts the MS-user when a message or report is delivered.
- 3.2.13 Auto-correlate-reports:** An auto-action which correlates delivery reports with the originally submitted messages or probes to which they are related.
- 3.2.14 Auto-delete:** An auto-action that deletes messages whose storage period has expired.
- 3.2.15 Auto-forward:** A class of auto-actions which causes the MS to forward a delivered message to one or more recipients. As the definition of Auto-forward is content-specific, it is not defined in this Service Definition. Rather, each type of Auto-forward auto-action is defined in the Specification for the content-type concerned.
- 3.2.16 Auto-modify:** An auto-action which applies modifications to the attributes of newly created entries.
- 3.2.17 child-entry:** An entry immediately subordinate to another entry (its parent-entry) in a tree-structured relationship. An entry which is not a child-entry is a main-entry.
- 3.2.18 child-sequence-number:** A sequence-number in a parent-entry pointing to a child-entry. A parent-entry has one value of child-sequence-number for each child-entry it possesses.
- 3.2.19 constraining set:** An information object set used to constrain the values of related components within a Set or Sequence. See ITU-T Rec. X.682 | ISO/IEC 8824-3.
- 3.2.20 content-specific:** Describes a specification or action whose effect depends on the content-type of the message being handled.
- 3.2.21 creation-time:** An attribute which records the date and time at which the entry was created by the MS.
- 3.2.22 Delete abstract-operation:** An abstract-operation used to delete one or more entries of a specified entry-class.
- 3.2.23 delivered-EITs:** A multi-valued attribute which indicates the encoded-information-types present in the content of a delivered-message.
- 3.2.24 delivered-message:** An entry of the Delivery or Delivery-log entry-classes which represents a delivered-message.
- 3.2.25 delivered-report:** An entry of the Delivery or Delivery-log entry-classes which represents a delivered-report.
- 3.2.26 Delivery:** An entry-class which contains entries that represent messages and reports delivered by the MTS to the MS.
- 3.2.27 Delivery-log:** An entry-class which contains entries that provide, for logging purposes, a restricted representation of messages and reports delivered by the MTS to the MS.

- 3.2.28 Draft:** An entry-class which contains draft-message entries.
- 3.2.29 draft-message:** An entry of the Draft entry-class which represents a message not yet submitted to the MTS.
- 3.2.30 entry:** An information object stored in the MS. See 3.2.17, 3.2.45, and 3.2.61 for further classification of entries.
- 3.2.31 entry-class:** A category of entry which represents a particular type of information object. The principal entry-classes are the Stored-message, the Message-log and the Auto-action-log entry-classes.
- 3.2.32 entry-information:** A parameter, used in abstract-operations, which conveys selected information from an entry.
- 3.2.33 entry-information-selection:** A parameter, used in abstract-operations, which indicates what information from an entry is being requested.
- 3.2.34 entry-type:** An attribute which indicates whether an entry contains a delivered-message, a delivered-report, a returned-content, a submitted-message, a submitted-probe, a draft-message or an auto-action-event entry.
- 3.2.35 Fetch abstract-operation:** An abstract-operation which allows an unrestricted set of attribute information for one selected entry of a specified entry-class to be fetched from the MS by the MS-user.
- 3.2.36 fetch-restrictions:** Restrictions, imposed by the MS-user, on the type of information it is prepared to receive as a result of Fetch. The possible restrictions are on attribute-length, content-types and encoded information types.
- 3.2.37 filter:** A parameter, used in abstract-operations, which applies a test to a particular entry and is either satisfied or not by that entry.
- 3.2.38 filter-item:** An assertion about the presence or value(s) of an attribute of a particular type in an entry under test. Each such assertion is either *true*, *false*, or *undefined*.
- 3.2.39 forwarding-request:** A parameter that may be present in the argument of the MS-message-submission abstract-operation, invoked by the MS-user, to request that a stored message is forwarded from the MS.
- 3.2.40 general-attribute:** An attribute which is valid for all types of messages and reports, independent of content-type. Only attributes of this type are defined in this Service Definition.
- 3.2.41 general-auto-actions:** An auto-action which is valid for all types of messages and reports, independent of content-type. Only auto-actions of this type are defined in this Service Definition.
- 3.2.42 grade:** Defined in 5.2 of ITU-T Rec. X.402 | ISO/IEC 10021-2.
- 3.2.43 limit:** A component of the selector parameter which specifies the maximum number of selected entries to be returned in the result of an abstract-operation.
- 3.2.44 List abstract-operation:** An abstract-operation which allows the selection of entries of a specified entry-class and requests certain attribute information to be returned for those entries.
- 3.2.45 main-entry:** An entry which may form the root of a set of related entries, organized in a tree-structured fashion. Zero or more child-entries may be associated with a main-entry.
- 3.2.46 matching-rule:** A rule which allows entries to be selected by making assertions concerning their attribute-values.
- 3.2.47 message-group:** A set of related entries. An entry's message-group-name attribute indicates the message-groups of which it is a member.
- 3.2.48 Message-log:** An entry-class which incorporates all entries of the Submission-log and Delivery-log entry-classes.
- 3.2.49 Message-submission abstract-operation:** An abstract-operation the MS invokes to submit a message to the MTS when the MS-user invokes MS-message-submission, or when certain content-specific auto-actions are performed.
- 3.2.50 Modify abstract-operation:** An abstract-operation used to modify the attributes of one or more entries within an entry-class.
- 3.2.51 MS abstract-service:** The set of capabilities that the MS offers to its users by means of its ports.
- 3.2.52 MS abstract-service-user:** The user of the MS abstract-service. An end-user may employ different UAs, at different times, to act as the MS-abstract-service-user when accessing the MS abstract-service.

- 3.2.53 MS abstract-service-provider:** The MS which provides the MS abstract-service.
- 3.2.54 MS-message-submission abstract-operation:** An abstract-operation which enables the MS-user to submit a message to the MTS (and optionally store a copy), or to store a draft message.
- 3.2.55 MS-probe-submission abstract-operation:** An abstract-operation which enables the MS-user to submit a probe to the MTS, and optionally store a copy.
- 3.2.56 MS-submission Port:** A port which supports the MS-submission services of the MS abstract-service. The MS-submission abstract-service offers the same services as the Message-submission abstract-service (of the MTS abstract-service), and in addition, offers services for the storage of submitted messages, the logging of submitted messages, and the forwarding of messages residing in the MS.
- 3.2.57 MS-submission-options:** A parameter used in abstract-operations and auto-actions to determine whether a message or probe is to be submitted to the MTS, or stored in the MS, or both submitted and stored.
- 3.2.58 MS-user:** A shorter form of “MS abstract-service-user”.
- 3.2.59 multi-valued attribute:** An attribute which may have several values associated with it.
- 3.2.60 override:** A component of the selector parameter which indicates that previously registered restrictions for the performance of the Fetch abstract-operation shall not apply in this instance of the abstract-operation.
- 3.2.61 parent-entry:** An entry immediately superior to one or more child-entries. A parent-entry which is not a child-entry of some other entry is a main-entry.
- 3.2.62 parent-sequence-number:** A sequence-number which identifies a child-entry's parent-entry. Each child-entry has one value of parent-sequence-number.
- 3.2.63 partial-attribute-request:** A form of the entry-information-selection parameter which requests the return of selected values only, from a multi-valued attribute. See 3.2.33.
- 3.2.64 range:** A parameter used in abstract-operations to select a contiguous sequence of entries of a specified entry-class.
- 3.2.65 Register-MS abstract-operation:** An abstract-operation which allows the MS-user to register information relevant to its interworking with the MS.
- 3.2.66 registration:** Information which is registered with the MS and stored between abstract-associations (until changed by the Register-MS abstract-operation). See 3.2.65.
- 3.2.67 registration-identifier:** An identifier for one particular set of registration parameters for an auto-action-type.
- 3.2.68 Retrieval Port:** A port which supports the retrieval services of the MS abstract-service.
- 3.2.69 retrieval-status:** An attribute which records whether an entry has been retrieved from the MS by the MS-user.. Possible values are *new*, *listed* and *processed*.
- 3.2.70 returned-content:** An entry of the Delivery entry-class which contains the returned content of a previously submitted message.
- 3.2.71 selector:** A parameter used in abstract-operations to select entries of a specified entry-class.
- 3.2.72 sequence-number:** An attribute which unambiguously identifies an entry. Sequence-numbers are allocated in ascending order.
- 3.2.73 single-valued attribute:** An attribute which may have one value only associated with it.
- 3.2.74 Stored-message:** An entry-class which incorporates all entries of the Submission and Delivery entry-classes.
- 3.2.75 Submission:** An entry-class which contains entries that represent messages and probes which have been submitted to the MTS via the MS.
- 3.2.76 Submission-log:** An entry-class which contains entries that provide, for logging purposes, a restricted representation of messages and probes which have been submitted to the MTS.
- 3.2.77 submitted-message:** An entry of the Submission or Submission-log entry-classes which represents a submitted-message.
- 3.2.78 submitted-probe:** An entry of the Submission or Submission-log entry-classes which represents a submitted-probe.

- 3.2.79 subscription:** A long-term agreement between the MS supplier or administrator and the MS customers on the availability and use of optional MS features. This Service Definition assumes that such an agreement can be made, but does not prescribe the method by which it is made.
- 3.2.80 Summarize abstract-operation:** An abstract-operation which provides an overview of the kind and number of entries of some entry-class which are currently stored in the MS.
- 3.2.81 UA-registration:** Information registered with the MS which is specific to one of a user's UAs. See 3.2.66.

4 Abbreviations

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

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

5 Conventions

This Service Definition uses the descriptive conventions listed below.

5.1 Conventions for abstract-services

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

- To define the information objects of the MS abstract-service, and other data types and values, ASN.1 itself.
- To define the MS abstract-service, the MHS-OBJECT, PORT, ABSTRACT-OPERATION, and ABSTRACT-ERROR information object classes of ITU-T Rec. X.411 | ISO/IEC 10021-4. These are derived directly from the corresponding information object classes defined in ITU-T Rec. X.880 | ISO/IEC 13712-1, which also defines the CONTRACT and CONNECTION-PACKAGE information object classes.
- To define attributes, the ATTRIBUTE information object class of 6.3.3.3.
- To define auto-actions, the AUTO-ACTION and AUTO-ACTION-ERROR information object classes of 6.5.
- To define matching-rules, the MATCHING-RULE information object class of ITU-T Rec. X.501 | ISO/IEC 9594-2 (see also 6.3.9.3).

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

- Mandatory (M):** A mandatory component shall be present in every instance of the class.
- Optional (O):** An optional component may be present in an instance of the class at the discretion of the object (e.g. user) supplying that instance.
- Conditional (C):** A conditional component shall be present in an instance of the class as specified by this Service Definition.

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

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

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

- S Single-valued
- M Multi-valued

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

- M Mandatory
- O Optional
- Not supported

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

- P Always present in the entry because:
 - it is mandatory for generation by the MS; or
 - it is a mandatory or defaulted parameter in the relevant abstract-operation.
- C Conditionally present in the entry. It is present if:
 - it is supported by the MS and subscribed to by the user, and
 - it is present in an optional parameter in the relevant abstract-operation.
- Always absent, otherwise.

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

- N No
- Y Yes

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

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

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

- S Single-valued
- M Multi-valued

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

- Amod Auto-modify auto-action
- Md Message-delivery abstract-operation
- Mod Modify abstract-operation
- MS Message Store
- Ms Message-submission and MS-message-submission abstract-operations
- Ps Probe-submission and MS-probe-submission abstract-operations
- Rd Report-delivery abstract-operation.

5.4 General font conventions

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

5.5 Font conventions for ASN.1 definitions

Throughout this Service Definition, ASN.1 definitions appear in Courier font to highlight the difference between normal text and ASN.1 definitions. The font used for ASN.1 definitions is also one size smaller than normal text.

5.6 Rules for ASN.1 definitions

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

5.7 Conventions for previous editions of this Service Definition

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

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

5.8 ASN.1 Packed Encoding Rules

Although the abstract syntax in this Service Definition contains extension markers, it has not been verified that these are present in all instances that would be required before Packed Encoding Rules could safely be used.

5.9 Interpretation of UTC Time values

Dates and times in the MHS protocols are represented using the ASN.1 *UTCTime* type which uses only two decimal digits to represent the year, leaving the century unspecified. Since MHS systems must deal with dates both in the past (e.g. submission times of old messages which may be held in local storage or forwarded) and in the future (expiry time, deferred delivery time), it is important to observe a standard convention to avoid inaccurate display or malfunction of the MHS when dates from different centuries are compared.

The two decimal digits give 100 different years that can be expressed; an implementation has to associate each of these values with a particular century. The chosen convention is that dates up to ten years prior to the current time and up to forty years ahead of the current time should be associated with the corresponding century, with the interpretation of the remaining 49 values being implementation dependent. For example, for a system operating in 1996 the values "86" to "99" are interpreted as 1986 to 1999 and the values "00" to "36" are interpreted as 2000 to 2036, and the values "37" to "85" are implementation dependent.

NOTE – This convention permits two possible implementation strategies. An implementation can choose a fixed interpretation of all the year values, such that the convention is satisfied throughout the expected life of the product, or it can interpret the dates dynamically, based on the current date, such that the implementation remains valid indefinitely. For example, an implementation could choose the fixed range 1970 to 2069 for the available values, meaning that the implementation would require revision if it is still in use by the year 2029.

SECTION 2 – MESSAGE STORE ABSTRACT-SERVICE DEFINITION

6 Message Store model

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

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

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

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

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

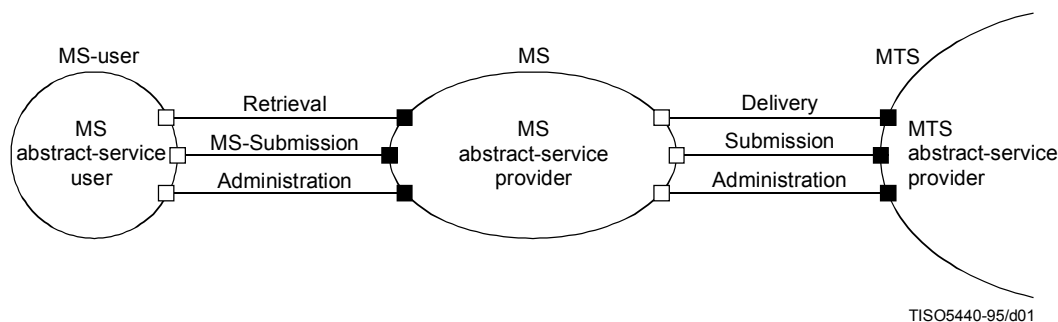


Figure 1 – Message Store abstract-service

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

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

6.1 Message Store objects and contracts

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

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

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

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

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

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

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

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

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

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

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

6.2 Message Store ports

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

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

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

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

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

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

6.2.1 Retrieval Ports

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

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

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

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

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

6.2.2 MS-submission Ports

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

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

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

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

6.2.3 Administration Ports

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

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

6.3 Information model

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

6.3.1 Entry-classes

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

- The **Stored-message** entry-class contains entries which correspond to messages, reports, probes, and draft messages.
- The **Message-log** entry-class contains entries that provide a restricted representation of messages, reports, and probes. These entries closely resemble the corresponding entries of the Stored-message entry-class, but contain only a subset of the attributes present in the latter in accordance with their logging function.
- The **Auto-action-log** entry-class contains entries that record the performance of auto-actions by the MS (see 6.3.7.3).

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

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

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

NOTES

- The term *information-base* used in CCITT Rec. X.413 (1988) and (1992) | ISO/IEC 10021-5:1990 has been replaced by *entry-class* in this Service Definition.
- In this Service Definition the description of the MS abstract-service assumes that all defined entry-classes are available for use. In practice, the behaviour of a given MS implementation will depend on its support for optional service components (e.g., the optional entry-classes and attribute-types) and on subscription. See 6.3.7.4.

6.3.2 Entries

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

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

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

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

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

6.3.3 Attributes

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

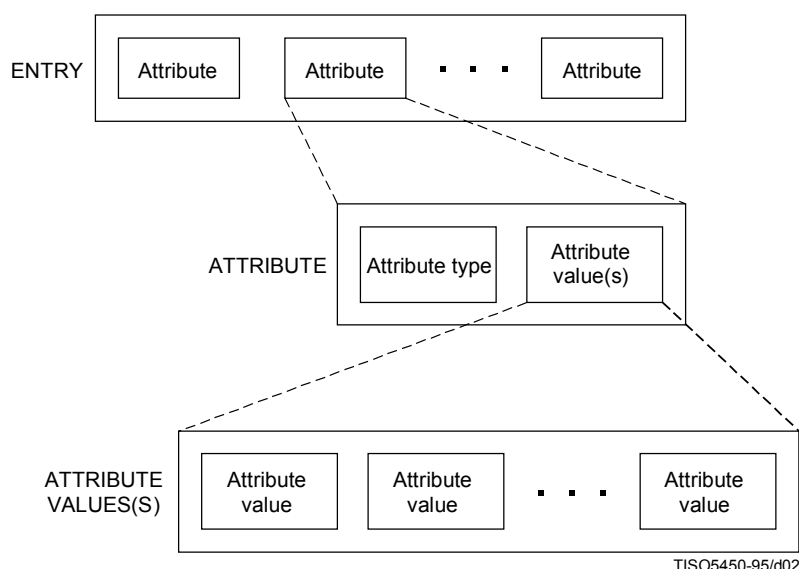


Figure 2 – The Components of an entry

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

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

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

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

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

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

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

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

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

6.3.3.1 Attribute-type

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

AttributeType ::= OBJECT IDENTIFIER

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

6.3.3.2 Attribute-values

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

6.3.3.3 The ATTRIBUTE information object class

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

- an **identifier** for the attribute which denotes the attribute-type;
- the **attribute-syntax** to which every value of the attribute shall conform;
- a collection of **matching-rules** identifying the types of matches which may be applied to the attribute (see 6.3.9);
- an indication of whether an attribute of this type shall contain only a single value, or may contain one or more values, all of the same ASN.1 type.

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

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

The field specifications have the following meaning:

- The **&id** value field is the identifier field for the class. It distinguishes a particular attribute-type from all others in the class.
- The **&Type** type field indicates the attribute-syntax to which every value of the attribute-type shall conform.
- The **&equalityMatch** value field identifies the matching-rule used to evaluate the equality component of a filter-item when applied to the attribute (see 8.1.2.2).

- d) The **&substringsMatch** value field identifies the matching-rule used to evaluate the substrings component of a filter-item when applied to the attribute (see 8.1.2.2).
- e) The **&orderingMatch** value field identifies the matching-rule used to evaluate the ordering components (greater-or-equal and less-or-equal) of a filter-item when applied to the attribute (see 8.1.2.2).
- f) The **&OtherMatches** value set field identifies any further matching-rules defined for the attribute-type.
- g) The **&numeration** value field specifies whether the attribute-type is single-valued or multi-valued.

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

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

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

- a) **parent-sequence-number**: This single-valued attribute gives the sequence-number of a child-entry's parent-entry. It is absent from a main-entry. Its definition is given in 11.2.51.
- b) **child-sequence-numbers**: This multi-valued attribute gives the sequence-numbers of all the child-entries of a parent-entry. It is absent from an entry which is not a parent-entry. Its definition is given in 11.2.8.

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

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

6.3.5 Content-specific Attributes

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

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

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

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

NOTES

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

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

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

6.3.6 Entry-types

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

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

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

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

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

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

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

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

6.3.7 Organization of entry-classes

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

Table 1 – Entry-types present in entry-classes

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

6.3.7.1 The Stored-message entry-class

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

- a) The **Delivery** entry-class contains entries with entry-type delivered-message, delivered-report, and returned-content, which are created by the performance of the Message-delivery and Report-delivery abstract-operations of the Delivery Port. The MS-user has no selective control over the creation of entries of the Delivery entry-class.
- b) The **Submission** entry-class contains entries with entry-type submitted-message and submitted-probe which are created by the performance of the MS-message-submission and MS-probe-submission abstract-operations, or by the performance of auto-actions which invoke the Message-submission abstract-operation. The MS shall create a submitted-message or submitted-probe entry only if the submission is successful, and if so requested by the MS-user. The request may be made explicitly in the argument of the submission or the registration-parameter of the auto-action, or may be registered for all submissions by means of the Register-MS abstract-operation.
- c) The **Draft** entry-class contains entries with entry-type draft-message. The value of the submission-options parameter (which is optionally present in the argument of the MS-message-submission abstract-operation) controls whether a Draft entry is created (with no submission to the MTS), or whether the message is actually submitted (see 8.1.6). The entries of the Draft entry-class differ mainly from those of the Submission entry-class in that they lack those attributes which are derived from the result parameter of the Message-submission abstract-operation returned by the MTS after successful submission.

6.3.7.2 The Message-log entry-class

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

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

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

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

- a) One Message-log entry exists for every Stored-message main-entry, each possessing identical sequence-number and creation-time attributes. However, a Message-log entry may be retained after the corresponding Stored-message entry has been deleted. A Message-log entry is created at the same time as the corresponding Stored-message entry is created.
- b) A Submission-log entry shall be created upon the submission of a message or probe, regardless of whether the submission is successful or of whether a Submission entry is also created (see 6.3.7.1). If the submission is not successful, the Submission-log entry shall not contain those attributes which are derived from Message-submission or Probe-submission result but shall contain an MS-submission-error attribute to indicate the reason for the failure of the submission. The MS-user has no selective control over the creation of Submission-log or Delivery-log entries.
- c) Every attribute of a Message-log entry (except for deletion-time and MS-submission-error, which are specific to Message-log entries) is identical with its counterpart in the corresponding Stored-message entry. The attributes of Message-log entries shall not be subject to modification directly (by means of the Modify abstract-operation or the Auto-modify auto-action), except for the message-group-name attribute which may be modified if no corresponding Stored-message entry exists. However, when an attribute of a Stored-message entry is modified the modification is also apparent in the corresponding Message-log entry.
- d) Certain attributes present in Stored-message entries shall be absent from the corresponding Message-log entries. These are message-delivery-envelope, message-submission-envelope, probe-submission-envelope, report-delivery-envelope, and content. The set of attribute-types available in Message-log entries is not prescribed, and is determined by subscription.
- e) A Message-log entry cannot be deleted before any corresponding Stored-message entry has been deleted. However, as a matter of local policy, the deletion of Message-log entries may be subject to additional restrictions. The MS shall continue to update attributes which hold correlation information in a Message-log entry even after the corresponding Stored-message entry has been deleted.

- f) The deletion-time attribute of a Message-log entry is created when the corresponding Stored-message entry is deleted. In the case of a Submission-log entry for which no Submission entry is created, the deletion-time attribute is generated when the Submission-log entry is created, and is assigned the same value as the entry's creation-time.
- g) Other Specifications which define message content-types may set down rules for the creation of a Message-log child-entry for each child-entry present in a Stored-message entry. No such rules are defined in this Service Definition. In particular, the Message-log entry-class shall not contain a child-entry representing the returned-content of a delivered-report.

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

6.3.7.3 The Auto-action-log entry-class

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

6.3.7.4 Entry-class support

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

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

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

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

6.3.8 Retrieval-status

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

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

The values of retrieval-status are as follows:

- a) *new*: The entry has neither been Listed by the MS-user nor has it been automatically *processed* by the MS.
- b) *listed*: Information about the entry has been returned to the MS-user in either a List abstract-operation or a Fetch abstract-operation, but the entry has not yet been completely *processed*.
- c) *processed*: Either an MS-user has 'completely fetched' the entry, or the MS has performed an auto-action on it, and the definition of that auto-action causes its retrieval-status to be set to *processed*. (Note that some auto-actions result in the message being deleted.) The meaning of 'completely fetched' depends on the message content-type and is defined in the relevant Specification.

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

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

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

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

6.3.9 Matching-rules

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

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

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

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

A matching-rule is defined through the specification of:

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

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

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

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

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

6.3.9.1 Generic Matching Rules

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

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

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

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

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

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

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

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

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

6.3.9.2 Matching Rule definition

The definition of a matching-rule involves:

- assigning an Object Identifier to the matching-rule;
- specifying the different types of matches supported by the rule;
- defining the syntax of an assertion of the matching-rule;
- defining the appropriate rules for evaluating a presented assertion with respect to target attribute-values held in an entry.

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

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

6.3.9.3 The MATCHING-RULE information object class

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

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

The object class contains the following fields:

- The **&AssertionType** type field indicates the assertion-syntax, i.e. the syntax to which every value in a presented matching-rule-assertion shall conform. It is always present in matching-rules used in this Service Definition.
- The **&id** value field is the identifier field for the class and contains an Object Identifier.

NOTES

- No support is provided in the defined syntax for actually defining the matching-rules themselves; this must be done by natural language or by other means.
- The defined syntax is used to define matching-rules in this and other Specifications.

6.4 Message grouping

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

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

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

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

- The MS-user shall register with the MS, by means of the Register-MS abstract-operation, the set of message-group-names that the user intends to employ for the classification of messages. This is analogous to the creation of a directory structure in a filestore. Subsequently, the MS shall verify that only registered message-group-names are assigned to entries, to ensure internal consistency. A message-group-descriptor may accompany each message-group-name registered to provide an informal description of its contents (see 8.2.5).

- b) The MS allows the registration of a message-group-name containing more than one group-name-part only if a registration exists for that message-group-name's parent, i.e. the message-group-name formed by omitting the last group-name-part of the name.
- c) The MS allows the deregistration of a registered message-group-name provided that the name is not present in the message-group-name attribute of any entry, that the name is not the parent of any other registered message-group, and that the name is not referenced in the modifications component of a registered Auto-modify auto-action.
- d) The MS may be instructed to automatically assign appropriate message-group-names to messages, controlled by the specification of selection criteria. This is performed by the Auto-modify auto-action (see 13.2).
- e) The MS-user may manually amend the membership of a message-group. This is performed by the Modify abstract-operation (see 8.2.7).
- f) The MS-user may retrieve details of the set of registered message-group-names for display to the end-user. This is performed by the Register-MS abstract-operation.
- g) The membership of a message-group may be determined by selection based on the value of the message-group-name attribute. The MS need not make any special provision for this as selection of entries based on the values of their attributes is a basic capability of the MS.

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

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

6.5 Auto-actions

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

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

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

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

6.5.1 The AUTO-ACTION information object class

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

```

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

AutoActionType ::= OBJECT IDENTIFIER

```

The object class contains the following fields:

- a) The **&id** is the identifier field for the class and contains an Object Identifier.
- b) The **&RegistrationParameter** type field defines the syntax of the registration-parameter associated with the auto-action-type (see 6.5.2). The field is optional as some auto-action-types require no registration-parameter.
- c) The **&Errors** value set field defines a set of errors, any of which may arise from an unsuccessful performance of the auto-action (see 6.5.3).

6.5.2 Auto-action registration

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

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

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

- a) **Auto-action-type** (M): The Object Identifier which identifies the auto-action-type. The constraining set (Auto-action-table) is defined in clause 13.
- b) **Registration-identifier** (O): Used to distinguish this auto-action-registration from other registrations of the same auto-action-type. This component also governs the order of execution of registered auto-actions of this auto-action-type. This component is disregarded in the case where the auto-action-type supports only a single registration.
- c) **Registration-parameter** (C): The parameter that the MS shall use when it performs this registered auto-action. This shall be present if the auto-action-type identifies an auto-action for which a registration parameter is defined, and is absent otherwise.

6.5.3 Auto-action errors

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

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

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

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

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

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

6.5.4 Auto-action execution

Auto-actions are performed autonomously by the MS (possibly in the absence of an abstract-association with the MS-user) when given conditions are fulfilled. The first stage of this process is the identification of an entry which has become a candidate for auto-action processing as a consequence of some event. The most common event which initiates auto-action processing is the creation of an entry (as the result of Message-delivery or Message-submission, for example). Other events, such as the expiry of a timer, may also initiate auto-action processing.

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

6.6 MS extensions

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

```
MS-EXTENSION ::= TYPE-IDENTIFIER
```

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

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

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

```
MSExtensionItem ::= INSTANCE OF MS-EXTENSION
```

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

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

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

7 MS-bind and MS-unbind operations

7.1 MS-bind abstract-operation

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

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

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

7.1.1 MS-bind-argument

The **MS-bind-argument** parameters are used to identify, authenticate and set the security-context for an MS-user. They also contain a set of restrictions for entries to be returned as the result of the Fetch abstract-operation, identify a set of registrations associated with this instance of the MS-user, and may contain a request to be informed of the entry-classes, auto-action-types, attribute-types, matching-rules, message-group support and content-types to which the MS-user has subscribed.

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

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

- a) **Initiator-name** (C): This parameter contains the name of the MS-user and is defined further in 8.1.1.1.1.1 of ITU-T Rec. X.411 | ISO/IEC 10021-4.
- b) **Initiator-credentials** (M): This parameter contains the *credentials* of the MS-user. The **initiator-credentials** may be used by the MS to authenticate the identity of the MS-user (see ITU-T Rec. X.509 | ISO/IEC 9594-8).

If simple-authentication is used, the initiator-credentials comprise a simple password. The password is defined in 8.5.11 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

If protected-authentication is used, the initiator-credentials comprise a password protected as described in clause 6 of ITU-T Rec. X.509 | ISO/IEC 9594-8 and further described in Annex H of ITU-T Rec. X.402 | ISO/IEC 10021-2.

If strong-authentication is used, the initiator-credentials comprise an initiator-bind-token, and, optionally, an initiator-certificate or certificate-selector. The initiator-bind-token, initiator-certificate, and certificate-selector are defined in 8.1.1.1.1.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4. The initiator-certificate shall contain the OR-address of the MS-user in the *x400Address* component in its subject alternative name field (see 12.3.2.1 of ITU-T Rec. X.509 | ISO/IEC 9594-8), unless the security-policy provides an alternative binding of the certificate to the MS-user. 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 ITU-T Rec. X.411 | ISO/IEC 10021-4.

- c) **Security-context** (O): This parameter identifies the **security-context** within which the MS-user proposes to operate. The security-context is defined in 8.1.1.1.1.3 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

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

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

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

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

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

The parameter contains the following components:

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

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

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

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

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

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

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

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

NOTE 3 – In 1988 Application Contexts, the maximum-attribute-length restriction applies only to an entry's content attribute.

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

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

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

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

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

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

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

7.1.2 MS-bind-result

```
MSBindResult ::= SET {
    responder-credentials          [2] ResponderCredentials,
    available-auto-actions         [3] SET SIZE (1..ub-auto-actions) OF AUTO-ACTION.&id
                                   ({AutoActionTable}) OPTIONAL,
    available-attribute-types      [4] SET SIZE (1..ub-attributes-supported) OF ATTRIBUTE.&id
                                   ({AttributeTable}) OPTIONAL,
    alert-indication               [5] BOOLEAN DEFAULT FALSE,
    content-types-supported        [6] SET SIZE (1..ub-content-types) OF OBJECT IDENTIFIER OPTIONAL,
    -- 1994 extensions --
    entry-classes-supported        [7] SET SIZE(1..ub-entry-classes) OF EntryClass OPTIONAL,
    matching-rules-supported       [8] SET SIZE(1..ub-matching-rules) OF OBJECT IDENTIFIER OPTIONAL,
    bind-result-extensions         [9] MSExtensions OPTIONAL,
    message-group-depth           [10] INTEGER (1..ub-group-depth) OPTIONAL,
    auto-action-error-indication   [11] AutoActionErrorIndication OPTIONAL,
    unsupported-extensions         [12] SET SIZE(1..ub-extensions) OF OBJECT IDENTIFIER OPTIONAL,
    ua-registration-id-unknown     [13] BOOLEAN DEFAULT FALSE,
    service-information            [14] GeneralString (SIZE(1..ub-service-information-length))
                                   OPTIONAL }
```

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

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

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

If protected-authentication is used, the responder-credentials comprise a password protected as described in clause 6 of ITU-T Rec. X.509 | ISO/IEC 9594-8 and further described in Annex H of ITU-T Rec. X.402 | ISO/IEC 10021-2.

If strong-authentication is used, the responder-credentials comprise a responder-bind-token generated by the MS and, optionally, a responder-certificate or certificate-selector. The responder-bind-token, responder-certificate, and certificate-selector are defined in 8.1.1.1.2.2 of ITU-T Rec. X.411 | ISO/IEC 10021-4. The responder-certificate shall contain the OR-address of the MS in the *x400Address* component in its subject alternative name field (see 12.3.2.1 of ITU-T Rec. X.509 | ISO/IEC 9594-8), unless the security-policy provides an alternative binding of the certificate to the MS."

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

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

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

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

- f) **Entry-classes-supported (C):** This identifies the entry-classes supported by the MS. Support for an entry-class implies that it shall be available to the Summarize, List, Fetch, Delete, and Modify abstract-operations, and to the auto-action-types supported by the MS. In addition, support for the Submission entry-class implies support for its use in the MS-message-submission and MS-probe-submission abstract-operations. Support for the Draft entry-class implies support for its use in the MS-message-submission abstract-operation. This parameter shall be present if the MS-configuration-request parameter of the MS bind-argument is *true*; it is absent otherwise.
- g) **Matching-rules-supported (C):** This identifies the matching-rules supported by the MS. Support for a matching-rule implies that it shall be available for use with an attribute for which the matching-rule is defined, where support for that attribute-type is also claimed. This parameter shall be present if the MS configuration-request parameter of the MS-bind-argument is *true* and at least one matching-rule is supported; it is absent otherwise.

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

- h) **Bind-result-extensions (C):** This parameter allows for future general and content-specific extensions to MS-bind-result. If the MS supports one or more additional capabilities whose specification defines an MS-extension to indicate that capability, the specified extensions shall be present. MS-extensions to indicate additional capabilities defined in this Service Definition are listed in Annex F; further extensions may be defined in future editions of this Service Definition, in content-specific Specifications, or to indicate proprietary capabilities.
- i) **Message-group-depth (C):** This indicates the maximum number of group-name-parts that may be present in a message-group-name. It shall be present if the MS-configuration-request parameter of the MS-bind-argument is *true* and the MS supports the message-group-name attribute-type; it is absent otherwise. If the MS-user presents, in the argument of an abstract-operation or in an auto-action registration-parameter, a message-group-name which contains more group-name-parts than are specified in this parameter, then a message-group-error results.
- j) **Auto-action-error-indication (C):** This shall be present if an auto-action performed by the MS since the previous abstract-association was established caused the generation of an auto-action-error. Since this

information cannot be reported when a 1988 Application Context is in use, the establishment of such an association shall not affect the setting of this parameter which shall be presented when a 1994 Application Context is next used.

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

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

- k) **Unsupported-extensions (C)**: If the MS-user supplied a bind-extensions parameter in the MS-bind-argument and one or more of these extensions are not supported by the MS, this parameter identifies all such unsupported-extensions. The parameter contains the Object Identifier values of each of these unsupported-extensions, drawn from the bind-extensions parameter. The parameter is present if the bind-extensions parameter of the MS-bind-argument is present and at least one of the extensions is not supported; it is absent otherwise.
- l) **UA-registration-id-unknown (C)**: This parameter is present and has the value *true* if the MS-bind-argument specifies a UA-registration-identifier which has not been registered previously by means of the Register-MS abstract-operation. It is absent otherwise.
- m) **Service-information (C)**: This provides information concerning the operation of the service. An MS implementation may use this to report operational matters such as service availability and resource usage. The parameter shall be present at the discretion of the MS service provider.

7.1.3 MS-bind-error

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

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

BindProblem ::= ENUMERATED {
    authentication-error          (0),
    unacceptable-security-context (1),
    unable-to-establish-association (2),
    ... -- 1994 extension addition -- ,
    bind-extension-problem        (3),
    inadequate-association-confidentiality (4) }
```

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

- a) **Unqualified-error (C)**: This indicates the nature of the error. If *authentication-error* is indicated, the abstract-association cannot be established because the initiator's credentials are not acceptable or are improperly specified. If *unacceptable-security-context* is indicated, then the security-context proposed by the initiator of the abstract-association is unacceptable to the responder. If *unable-to-establish-association* is indicated, then the responder has rejected the initiator's attempt to establish an abstract-association for an unspecified reason. If *bind-extension-problem* is indicated, the abstract-association cannot be established because the bind-extensions parameter specified in the MS-bind-argument contains an improperly specified extension. If *inadequate-association-confidentiality* is indicated, the abstract-association cannot be established because the underlying connection does not provide the necessary confidentiality.
- b) **Qualified-error (C)**: This parameter indicates the nature of the error, and may, in addition, specify **supplementary-information** which expands on the error. Where one or more bind-extensions parameters have been improperly specified, the Object Identifier values of these extensions shall be reported in **bind-extension-errors**.

7.2 MS-unbind abstract-operation

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

ms-unbind ABSTRACT-OPERATION ::= emptyUnbind

8 Abstract-operations

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

8.1 Common data-types used in abstract-operations

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

- a) Range;
- b) Filter;
- c) Selector;
- d) Entry Information Selection;
- e) Entry Information;
- f) MS Submission Options;
- g) Common Submission Results.

8.1.1 Range

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

```

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

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

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

CreationTime ::= UTCTime
  
```

The components of range have the following meaning:

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

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

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

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

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

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

The values of sequence-number and creation-time specified in range need not identify existing entries.

8.1.2 Filters

8.1.2.1 Filter

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

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

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

- A filter which is an **item** has the value of the filter-item (i.e. *true*, *false*, or *undefined*).
- A filter which is the **and** of a set of filters is *true* if the set is empty or if each filter is *true*; it is *false* if at least one filter is *false*; otherwise it is *undefined* (i.e. at least one filter is *undefined* and no filters are *false*).
- A filter which is the **or** of a set of filters is *false* if the set is empty or if each filter is *false*; it is *true* if at least one filter is *true*; otherwise it is *undefined* (i.e. at least one filter is *undefined* and no filters are *true*).
- A filter which is the **not** of a filter is *true* if the filter is *false*; *false* if it is *true*; and *undefined* if it is *undefined*.

8.1.2.2 Filter-item

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

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

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

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

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

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

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

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

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

- a) **equality**, it is *true* if and only if there is a value of the attribute for which the attribute's equality matching-rule applied to that value and the presented value returns *true*;
- b) **substrings**, it is *true* if and only if there is a value of the attribute in which the attribute's **substring** matching-rule applied to that value and the presented value in **strings** returns *true*. The meaning of the components of **strings** is defined in the appropriate matching-rule definition; see 12.2.3 for an example of a substrings matching-rule.
- c) **greater-or-equal**, it is *true* if and only if there is a value of the attribute for which the attribute's **ordering** matching-rule applied to that value and the presented value returns *false*, i.e. there is a value of the attribute which is *greater than or equal to* the presented value;
- d) **less-or-equal**, it is *true* if and only if there is a value of the attribute for which either the attribute's **equality** matching-rule or **ordering** matching-rule applied to that value and the presented value returns *true*, i.e. there is a value of the attribute which is *less than or equal to* the presented value;
- e) **present**, it is *true* if and only if such an attribute is present in the entry;
- f) **approximate-match**, it is *true* if and only if there is a value of the attribute for which the **approximate-match** matching-rule applied to that value and the presented value returns *true*. The approximate-match matching-rule may provide phonetic matching to accommodate spelling variations, and is locally defined; there are no specific guide-lines for approximate matching in this Service Definition. If **approximate-match** is not supported, this filter-item shall be treated as a match for **equality**;
- g) **other-match**, it is *true* if and only if there is a value of the attribute with the indicated attribute-type for which the matching-rule, applied to that value and the presented-value, returns *true*. If the matching-rule is one of those specified as an other-matching-rule in the definition of the specified attribute-type, and is supported by the MS, then matching proceeds as specified in the matching-rule definition; otherwise an attribute-error of inappropriate-matching may result. As a local matter, an MS may support matching-rules in addition to those defined in this Service Definition. The constraining set (MatchingRuleTable) is defined in clause 12.

8.1.2.3 Attribute-value-assertion

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

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

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

- a) *true*, if the entry contains an attribute of that attribute-type, one of whose attribute-values matches the presented attribute-value;
- b) *undefined*, if the entry contains an attribute of that attribute-type, and the specified matching-rule declares the result of the match to be undefined;
- c) *false*, otherwise.

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

8.1.3 Selector

A **selector** parameter is used to select entries of a specified entry-class. The selection operates in three stages. Firstly, the total set of entries of the entry-class may be restricted to a particular contiguous set by specifying its range. Secondly, entries from within this set may be selected by specifying a filter which the selected entries shall satisfy. Thirdly, a limit may be placed on the number of entries thus selected. If the range is defined in ascending order (or is omitted), those

entries with the lowest sequence-numbers are selected; if the range is defined in descending order, those entries with the highest sequence numbers are selected (see 8.1.1). Abstract-operations are applied to selected entries according to their range order, i.e. where the range is defined in ascending order, the selected entry with the lowest sequence-number is operated on first.

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

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

- Child-entries (O)**: If *false*, only main-entries are considered for selection. If *true*, both main-entries and child-entries are considered for selection. In the absence of this component, only main-entries are considered for selection.
- Range (O)**: The abstract-syntax-notation of range is given in 8.1.1. In the absence of this component, the range is unbounded.
- Filter (O)**: The abstract-syntax-notation of filter is given in 8.1.2.1. In the absence of this component, all entries within the specified range are selected.
- Limit (O)**: This allows the specification of an upper limit to the number of entries selected. In the absence of this component, there is no limit to the number of entries selected.

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

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

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

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

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

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

8.1.4 Entry-information-selection

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

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

```
AttributeSelection ::= SET {
  type ATTRIBUTE.&id ({AttributeTable}),
  from [0] INTEGER (1..ub-attribute-values) OPTIONAL --used if type is multi valued--,
  count [1] INTEGER (0..ub-attribute-values) OPTIONAL
  -- for 1988 Application Contexts the lower bound is one-- }
```

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

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

- b) **From** (O): When an attribute is multi-valued, this Integer gives the relative position of the first value to be returned. If it specifies a value beyond those present in the attribute, no values are returned. This component shall not be present unless the attribute-type is multi-valued. If this component is omitted, values starting at the first value are returned.
- c) **Count** (O): This Integer specifies the maximum number of values to be returned. It is not an error if **count** is greater than the number of values present in the entry. If **count** is zero, then information is requested on the total number of values present in the attribute but no actual values are returned. If this component is omitted, there is no limit as to how many values are returned.

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

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

8.1.5 Entry-information

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

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

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

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

- a) **Sequence-number** (M): The sequence-number identifying the entry (see 6.3.2).
- b) **Attributes** (C): The set of selected attributes from the entry. Where the corresponding attribute-selection constitutes a partial-attribute-request, only the requested subset of the attribute-values actually present in that attribute are returned. This component is absent if the requested-attributes parameter was omitted from the list-argument or fetch-argument, and no (UA-specific or general) list-attribute-defaults or fetch-attribute-defaults are in force. The component is also absent if none of the requested-attributes are present in the entry, or if all the requested-attributes present were specified in entry-information-selections in which the count component indicated that zero attribute-values were to be returned.
- c) **Value-count-exceeded** (C): This component is present where, in the performance of the List or Fetch abstract-operation, **count** is specified in the attribute-selection, and the specified attribute contains values beyond the limit set by count. The **type** indicates the attribute-type concerned, and **total** indicates the total number of values possessed by the attribute. One value of the component is present for each attribute-selection which exceeds the limit set by count. The component is absent if none of the attribute-selections exceeded their count limits.

8.1.6 MS-submission-options

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

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

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

- a) **Object-entry-class** (O): This specifies the entry-class of the entry to be created, that will contain details of the associated message or probe submission. Three values are defined:
 - 1) **Submission**. The message or probe shall be submitted to the MTS and an entry created in the Submission, and, subject to subscription, the Submission-log entry-classes.
 - 2) **Submission-log**. The message or probe shall be submitted to the MTS and an entry created in the Submission-log entry-class only.

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

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

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

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

- d) **MS-submission-extensions** (O): This component allows for future general and content-specific extensions to MS-submission-options. The Specification for a given content-type defines its use of this component. In the absence of the component no MS-submission-extensions are specified. This Service Definition defines the following extensions:
- 1) **Originator-token** (O): This extension is used where the submitted message contains a message-token which contains encrypted-data that has been encrypted such that it cannot subsequently be decrypted by the originator. This extension enables the originator to supply a message-token constructed as if the originator were a recipient of the message, to be stored in the submitted-message entry but not submitted to the MTS. Subsequently, the originator may retrieve this information and use it to recover the original message.

```
originator-token MS-EXTENSION ::= {
    OriginatorToken IDENTIFIED BY id-ext-originator-token}

OriginatorToken ::= MessageToken
    (CONSTRAINED BY {-- Must contain an asymmetric-token with an
        -- encrypted-data component --} )
```

The **originator-token** contains a *message-token* argument of the Message-submission abstract-operation (see 8.2.1.1.1.26 of ITU-T Rec. X.411 | ISO/IEC 10021-4) which contains an *encrypted-data* component that is encrypted using the public key of the message originator.

NOTES

1 When storage-on-submission is used, the originator retains a copy of the message in the MS, but is not treated as a recipient in the message submission envelope. This extension provides the originator with the security arguments that are encrypted on a per-recipient basis for the conventional recipients of the message. Note that content-integrity-check does not need to be duplicated here, as it is only a signature, and so the originator is implicitly able to use any of the values provided for the other recipients.

2 If Content Confidentiality is provided using a symmetric encryption algorithm with a content-confidentiality-key (session key) which is itself encrypted such that it requires each recipient's private key to decrypt it, then the message's originator would have no means of decrypting the copy of the message as stored in the MS on submission. This MS-submission-extension enables the MS-user to supply a value with the submitted message, which is stored in the submitted-message entry but is not included in the message submitted to the MTS. This contains the session key for the message, encrypted with the public key of the submitting MS-user. When the MS-user subsequently retrieves the submitted-message, his private key may be used to decrypt the session key, and hence decrypt the message.

3 Where Content Confidentiality is provided by use of a symmetric algorithm, and a method other than the message-token is used to distribute the key, then the originating MS-user must employ a different mechanism to retain the key and so enable subsequent decryption of the stored-message.

4 Where Content Confidentiality is provided directly by use of an asymmetric algorithm, it is unlikely that storage on submission will be useful, except where the key pair is shared between users, including the originating MS-user, who thus has access to both public and private keys.

- 2) **Originator-certificate-selectors-override (O)**: This extension contains information sufficient to identify one certificate, where the originator has more than one certificate with the same algorithm-identifier. The extension enables the originator to identify a certificate to validate specific digital signatures within the content-integrity-check or message-token arguments of Message-submission, or to be used for key agreement or asymmetric encryption. It also allows a certificate of each recipient to be identified for key agreement or asymmetric encryption. It is constructed as if the originator were a recipient of the message, and is stored in the submitted-message entry rather than being submitted to the MTS. Subsequently, the originator may retrieve this information and use it to recover the original message.

```
originator-certificate-selectors-override MS-EXTENSION ::= {
  CertificateSelectors (WITH COMPONENTS{
    ...,
    message-origin-authentication ABSENT}) IDENTIFIED BY
    id-ext-originator-certificate-selectors-override }
```

If this extension is present and MS-submission-options request storage of the submitted message, then when generating the certificate-selectors attribute the value in each component that is present replaces the value in the corresponding component of certificate-selectors (see 11.2.9).

8.1.7 Common-submission-results

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

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

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

- Created-entry (C)**: This shall be present if the MS-user requested storage of a submitted message or probe in the Submission or Submission-log entry-classes, or storage of a draft-message entry in the Draft entry-class. It indicates the sequence-number of the newly created entry.
- Auto-action-error-indication (C)**: This shall be present if an auto-action performed by the MS concerning the submitted-message or probe, or stored draft-message, caused the generation of an auto-action-error. If the Auto-action-log entry-class is subscribed to, this parameter indicates the sequence-number of the first entry which records these unsuccessful auto-action executions. Otherwise it contains a Null (see 7.1.2, item j). The parameter shall be absent if no auto-action-errors occurred.
- MS-submission-result-extensions (C)**: This component allows for future general and content-specific extensions to common-submission-results. No extensions are defined in this Service Definition. It shall be present if the MS-user specified MS-submission-extensions in the submission-options parameter of the MS-message-submission or MS-probe-submission argument, and those extensions caused the generation of this corresponding result.

8.2 Retrieval Port abstract-operations

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

- Summarize;
- List;
- Fetch;
- Delete;
- Register-MS;
- Alert;
- Modify.

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

- the Delete, Register-MS, and Modify abstract-operations shall not be performed until all outstanding abstract-operations have been completed;
- the Delete, Register-MS, and Modify abstract-operations are performed in the order in which they are invoked and are allowed to complete before any other abstract-operation is performed.

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

8.2.1 Summarize abstract-operation

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

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

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

```

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

```

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

8.2.1.1 Summarize-argument

```

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

```

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

- Entry-class (O)**: This specifies which entry-class is addressed by the abstract-operation (see 6.3.1).
- Selector (M)**: This is a set of selection criteria to determine which entries shall be summarized (see 8.1.3).
- Summary-requests (O)**: This is the sequence of attribute-types for which summaries are requested. This parameter is present only if a summary is requested.
- Summarize-extensions (O)**: This parameter allows for future general and content-specific extensions to summarize-argument. No extensions are defined in this Service Definition.

8.2.1.2 Summarize-result

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

```

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

```

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

- Next (C)**: This is returned in the case where the number of entries selected would have been greater if it were not for the limit specified in the selector. The parameter contains the sequence-number of the next entry that would have been selected, as determined by the range order which determines the direction of searching (see 8.1.3).
- Count (M)**: This is an integer giving the count of entries that match the selection criteria.
- Span (C)**: This contains the lowest and highest sequence-numbers of the entries that match the selection criteria. It is absent if there are no such entries.

```

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

```

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

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

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

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

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

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

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

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

- i) **Type** (M): The type of the attribute.
 - ii) **Value** (M): The attribute-value for which the count is given.
 - iii) **Count** (M): A count of entries with this attribute-value.
- e) **Summarize-result-extensions** (C): This parameter allows for future general and content-specific extensions to summarize-result. No extensions are defined in this Service Definition.

8.2.1.3 Summarize abstract-errors

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

8.2.2 List abstract-operation

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

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

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

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

8.2.2.1 List-argument

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

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

- a) **Entry-class** (O): This specifies which entry-class is addressed by the abstract-operation (see 6.3.1).
- b) **Selector** (M): This is a set of selection criteria to determine which entries shall be returned (see 8.1.3).
- c) **Requested-attributes** (O): This indicates what information from the selected entries is to be returned in the result (see 8.1.4). If this parameter is absent, the registered set of UA-list-attribute-defaults is used. If no UA-registration was specified when the abstract-association was established, or no UA-list-attribute-defaults are registered for the currently active UA-registration, then the general (non UA-specific) list-attribute-defaults are used (see 8.2.5.1, items c and g).
- d) **List-extensions** (O): This parameter allows for future general and content-specific extensions to list-argument. No extensions are defined in this Service Definition.

8.2.2.2 List-result

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

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

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

- a) **Next** (C): This is returned in the case where the number of entries selected would have been greater if it were not for the limit specified in the selector. The parameter contains the sequence-number of the next entry that would have been selected, as determined by the range order which determines the direction of searching (see 8.1.3).
- b) **Requested** (C): This conveys the requested entry-information (see 8.1.5) from each selected entry. Each item of entry-information appears in the same order (ascending or descending) as the range order (see 8.1.3). The parameter is absent if no entries were selected.
- e) **List-result-extensions** (C): This parameter allows for future general and content-specific extensions to list-result. No extensions are defined in this Service Definition.

8.2.2.3 List abstract-errors

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

8.2.3 Fetch abstract-operation

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

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

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

8.2.3.1 Fetch-argument

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