INTERNATIONAL STANDARD

ISO/IEC 9506-2

First edition
1990-10-15

Industrial automation systems — Manufacturing Message Specification —

Part 2:
Protocol specification

Systèmes d'automatisation industrielle — Spécification de messagerie industrielle — Partie 2: Spécification de protocole

Reference number
ISO 9506-2 : 1990 (E)

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

International Standard ISO/IEC 9506-2 was prepared by Technical Committee ISO/IEC TC 184, *Industrial automation systems*, Sub-Committee SC 5, *System integration and communication.*

ISO/IEC 9506 consists of the following parts, under the general title *Industrial automation systems — Manufacturing Message Specification*:

— *Part 1: Service definition*

— *Part 2: Protocol specification.*

Introduction

This part of ISO/IEC 9506 provides a wide variety of services useful for various manufacturing and process control devices. It is designed to be used both by itself and in conjunction with Companion Standards, which describe the application of subsets of these services to particular device types.

The services provided by the Manufacturing Message Specification (MMS) range from simple to highly complex. It is not expected that all of these services will be supported by all devices. The subset to be supported is limited in some cases by Companion Standards, and in all cases may be limited by the implementor. Characteristics important in selection of a subset of services to be supported include:

a) applicability of the service to the device;

b) the complexity of services and requirements;

c) the complexity of provision of a particular class of service via the network versus the complexity of the device.

Security considerations

When implementing MMS in secure or safety critical applications, features of the OSI security architecture may need to be implemented. Appropriate features should be selected from ISO 7498-2 covering safety architectures and features. Those of particular interest cover the position (in OSI) of

a) access control;

b) authentication;

c) non-repudiation.

Specific implementation methods shall be at the discretion of the implementor.

Complexity of services and requirements

Some MMS services are quite complex and should be considered as advanced functions. Devices used in very simple applications often will not require such advanced functions, and hence will not support such MMS services.

Keywords

Application Interworking
Application Layer Protocol
Information Processing Systems
Manufacturing Communications Network
Manufacturing Message Specification
Numerical Control System
Open Systems Interconnection
OSI Reference Model
Process Control System
Programmable Controller
Programmable Device
Robotics Control System
Virtual Manufacturing Device

General

This part of ISO/IEC 9506 is one of a set of International Standards produced to facilitate the interconnection of information processing systems. It is positioned within the application layer of the Open Systems Interconnection Environment as an Application Service Element (ASE) with respect to other related standards by the Basic Reference Model for Open Systems Interconnection (ISO 7498).

The aim of Open Systems Interconnection is to allow, with a minimum of technical agreement outside the interconnection standards, the interconnection of information processing systems:

a) from different manufacturers;

b) under different managements;

c) of different levels of complexity;

d) of different ages.

Purpose

The purpose of this part of ISO/IEC 9506 is to define the Manufacturing Message Specification Protocol. It is most closely related to and lies within the field of application of the Manufacturing Message Specification Service Definition, ISO/IEC 9506-1. It also uses and references the Association Control Service Element Definition (ISO 8649) and the Presentation Layer Service Definition (ISO 8822), whose provisions it assumes in order to accomplish the aims of the Manufacturing Message Specification Protocol. The inter-relationship of these International Standards is depicted in Figure 1.

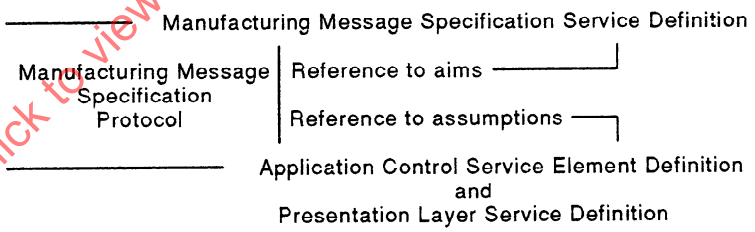


Figure 1 — Relationship between the Manufacturing Message Specification Protocol and adjacent services

The MMS protocol is structured so that subsets of protocol can be defined. The variations and options available within this part of ISO/IEC 9506 are essential to enable a Manufacturing Message Specification to be provided for a wide variety of applications. Thus, a minimally conforming implementation will not be suitable for use in all possible circumstances. It is important, therefore, to qualify all references to this part of ISO/IEC 9506 with statements of the options provided or required with statements of the intended purpose of provision or use.

NOTE — The services of this part of ISO/IEC 9506 are generic, and intended to be referenced by Companion Standards, each of which is directed to a more specific class of application. The services of this part of ISO/IEC 9506 may also be used in a stand-alone manner (without the use of Companion Standards).

It should be noted that, as the number of valid protocol sequences is very large, it is not possible with current technology to verify that an implementation will operate the protocol defined in this part of ISO/IEC 9506 correctly under all circumstances. It is possible by means of testing to establish confidence that an implementation correctly operates the protocol in a representative sample of circumstances. It is, however, intended that this part of ISO/IEC 9506 can be used in circumstances where two implementations fail to communicate in order to determine whether one or both have failed to operate the protocol correctly.

Intended users

The primary aims of this part of ISO/IEC 9506 is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer MMS entities at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

a) as a guide for implementors and designers;

b) for use in the testing and procurement of equipment;

c) as part of an agreement for the admittance of systems into the open systems environment;

d) as a refinement to the understanding of OSI.

This part of ISO/IEC 9506 is concerned, in particular, with the communication and interworking of programmable devices on the plant floor. By using this part of ISO/IEC 9506 together with other standards positioned within the OSI Reference Model, otherwise incompatible systems may work together in any combination.

Industrial automation systems - Manufacturing Message Specification - Part 2: Protocol specification

1 Scope

The Manufacturing Message Specification is an application layer standard designed to support messaging communications to and from programmable devices in a Computer Integrated Manufacturing (CIM) environment.

1.1 Specifications

This part of ISO/IEC 9506 specifies:

a) procedures for a single protocol for the transfer of data and control information from one application entity to a peer application entity in the MMS-context;

b) the means of selecting the services to be used by the application entities while communicating in the MMS-context;

c) the structure of the Manufacturing Messaging Specification Protocol Data Units used for the transfer of data and control information.

1.2 Procedures

The procedures are defined in terms of

a) the interactions between peer application entities through the exchange of Manufacturing Message Specification Application Protocol Data Units;

b) the interactions between an MMS-provider and the MMS-user in the same system through the exchange of MMS primitives;

c) the interactions between an MMS-provider and the Association Control Service Element through the exchange of association control service primitives;

d) the interactions between an MMS-provider and a presentation service provider through the exchange of Presentation service primitives.

1.3 Applicability

These procedures are applicable to instances of communication between systems which support MMS within the application layer of the OSI Reference Model, and which require the ability to interconnect in an open systems interconnection environment.

1.4 Conformance

This part of ISO/IEC 9506 also specifies conformance requirements for systems implementing these procedures. This part of ISO/IEC 9506 does not contain tests to demonstrate compliance with such requirements.

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9506. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9506 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 646 : 1983, *Information processing — ISO 7-bit coded character set for information interchange.*

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

ISO 7498-2 : 1989, *Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture.*

ISO 7498-3 : 1989, *Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 3: Naming and addressing.*

ISO 8326 : 1987, *Information processing systems — Open Systems Interconnection — Basic connection oriented session service definition.*

ISO/TR 8509 : 1987, *Information processing systems — Open Systems Interconnection — Service conventions.*

ISO 8571 : 1988, *Information processing systems — Open Systems Interconnection — File Transfer, Access and Management.*

ISO 8649: 1988, Information processing systems — Open Systems Interconnection — Service definition for the Association Control Service Element.

ISO 8650 : 1988, *Information processing systems — Open Systems Interconnection — Protocol specification for the Association Control Service Element.*

ISO 8822 : 1988, *Information processing systems — Open Systems Interconnection — Connection oriented presentation service definition.*

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

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

ISO 8825 : 1987, *Information processing systems — Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

ISO 8825/Add 1 : — [1], *Information processing systems — Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) Addendum 1: ASN.1 Extensions.*

ISO 9040 : — [1], *Information processing systems — Open Systems Interconnection — Virtual terminal service — Basic class.*

ISO 9041 : — [1], *Information processing systems — Open Systems Interconnection — Virtual terminal protocol — Basic class.*

ISO/IEC 9506-1 : 1990, *Industrial automation systems — Manufacturing Message Specification — Part 1: Service definition.*

ISO/IEC 9545-1 : 1989, *Information technology — Open Systems Interconnection — Application Layer Structure.*

ISO/IEC 9594 : — [1], *Information processing systems — Open Systems Interconnection — The Directory.*

IEEE 754 : 1985, *IEEE Standard for Binary Floating-Point Arithmetic.*

3 Definitions

NOTE — The definitions contained in this clause make use of abbreviations defined in clause 4.

For the purposes of this part of ISO/IEC 9506, the following definitions apply.

1) To be published.

3.1 Reference Model definitions

This part of ISO/IEC 9506 is based on the concepts developed in the Basic Reference Model for Open Systems Interconnection (ISO 7498), and makes use of the following terms defined in that International Standard:

a) application-entity;

b) application-process;

c) application service element;

d) open system;

e) (N)-protocol;

f) (N)-protocol-data-unit;

g) (N)-service-access-point;

h) (N)-layer;

i) system;

j) (N)-user-data.

3.2 Service Convention definitions

This part of ISO/IEC 9506 makes use of the following terms defined in the OSI Service Conventions (ISO TR 8509) as they apply to the Manufacturing Message Specification:

a) confirm;

b) indication;

c) primitive;

d) request;

e) response;

f) service primitive;

g) service provider;

h) service user.

3.3 Abstract Syntax Notation definitions

This part of ISO/IEC 9506 makes use of the following terms defined in the Abstract Syntax Notation One (ASN.1) Specification (ISO 8824):

1) value;

2) type;

3) simple type;

4) structure type;

5) component type;

6) tag;

7) tagging;

8) type (or value) reference name;

9) character string type;

10) boolean type;

11) true;

12) false;

13) integer type;

14) bitstring type;

15) octetstring type;

16) null type;

17) sequence type;

18) sequence-of type;

19) tagged type;

20) choice type;

21) selection type;

22) real type;

23) object identifier type;

24) MACRO;

25) module;

26) production;

27) ASN.1 encoding rules;

28) ASN.1 character set;

29) external type.

3.4 Other definitions

For the purpose of this part of ISO/IEC 9506, the following definitions also apply:

3.4.1 AA-specific (Application Association specific):

An adjective used to describe an object whose name has a scope that is a single application association (i.e. the name may be referenced only on the application association over which the object was defined).

3.4.2 attribute:

A data element, having a defined meaning, together with a statement of the set of possible values it may take.

3.4.3 Called MMS-user:

The MMS-user that issues the Initiate.response service primitive.

3.4.4 Calling MMS-user:

The MMS-user that issues the Initiate.request service primitive.

3.4.5 Client:

The peer communicating entity which makes use of a VMD for some particular purpose via a service request instance.

3.4.6 conformance building block (CBB):

An atomic unit used to describe MMS conformance requirements.

3.4.7 data:

Any representation to which meaning is or might be assigned (e.g. characters).

3.4.8 domain:

An abstract object that represents a subset of the capabilities of a VMD which is used for a specific purpose.

3.4.9 Domain-specific:

An adjective used to describe an object whose name has a scope that is a single domain (i.e. the name can be referenced over all application associations established with the VMD that may reference this domain).

3.4.10 download:

The process of transferring the content of a domain, including any subordinate objects, via load data to an MMS-user.

3.4.11 event management:

The management of event conditions, event actions, and event enrollments.

3.4.12 file:

An unambiguously named collection of information having a common set of attributes.

3.4.13 file operation:

The transfer of files between open systems, the inspection, modification, or replacement of part of a file's content, or the management of a file and its attributes.

3.4.14 filestore:

An organized collection of files, including their attributes and names, residing at a particular open system.

3.4.15 information:

The combination of data and the meaning that it conveys.

3.4.16 invalid PDU:

A PDU which does not comply with the requirements of this part of ISO/IEC 9506 with respect to structure, semantic meaning, or both.

3.4.17 journal:

A set of recorded, time-tagged event transitions, variable data, and/or comments, which may be logically ordered during retrieval.

3.4.18 local matter:

A decision made by a system concerning its behaviour in the Manufacturing Message Specification that is not subject to the requirements of this part of ISO/IEC 9506.

3.4.19 Manufacturing Message Protocol Machine (MMPM):

An abstract machine that carries out the procedures specified in this part of this part of ISO/IEC 9506.

3.4.20 MMS-context:

A specification of the service elements of MMS and semantics of communication to be used during the lifetime of an application association.

3.4.21 MMS-provider:

That part of the application entity that conceptually provides the MMS service through the exchange of MMS PDUs.

3.4.22 MMS-user:

That portion of the application process which conceptually invokes the Manufacturing Message Specification.

3.4.23 monitored event:

A detected change in the state of an event condition.

3.4.24 network-triggered event:

A trigger which occurs due to an explicit solicitation by a client.

3.4.25 operator station:

An abstract object representing equipment associated with a VMD that provides for input/output interaction with an operator.

3.4.26 predefined object:

An object, whose name is of VMD-specific or Domain-specific scope, that is instantiated through the use of some mechanism other than an MMS service.

3.4.27 program invocation:

An abstract object representing a dynamic element which most closely corresponds to an execution thread in a multi-tasking environment, which is composed of a set of domains.

3.4.28 protocol error:

A PDU that does not comply with the requirements of this part of ISO/IEC 9506.

3.4.29 Receiving MMPM:

The MMPM that receives an MMS PDU.

3.4.30 Receiving MMS-user:

The MMS-user that receives an indication or confirmation service primitive.

3.4.31 remote device control and monitoring:

The manipulation or inspection of the state of a device attached to the responder of a service request.

3.4.32 Requesting MMS-user:

The MMS-user that issues the request service primitive for a service.

3.4.33 Responding MMS-user:

The MMS-user that issues the response service primitive for a service.

3.4.34 semaphore:

A conceptual lock associated with a logical or physical resource that permits access to that resource only by an owner of the lock.

3.4.35 semaphore management:

The control of semaphores.

3.4.36 Sending MMPM:

The MMPM that sends an MMS PDU.

3.4.37 Sending MMS-user:

The MMS-user that issues a request or response service primitive.

3.4.38 Server:

The peer communicating entity which behaves as a VMD for a particular service request instance.

3.4.39 standardized object:

An object instantiation, whose name is of VMD-specific or Domain-specific scope, whose definition is provided in this part of ISO/IEC 9506 or an MMS Companion Standard.

3.4.40 type:

An abstract description of the class of data which may be conveyed by the value of a variable.

3.4.41 upload:

The process of transferring the content of a domain, including any subordinate objects, via load data from a remote user, in such a manner as to allow subsequent download.

3.4.42 valid PDU:

A PDU which complies with the requirements of this part of ISO/IEC 9506 with respect to structure and semantic meaning.

3.4.43 variable:

One or more data elements that are referred to together by a single name or description.

3.4.44 variable access:

The inspection or modification of variables or components of variables defined at a VMD.

3.4.45 Virtual Manufacturing Device (VMD):

An abstract representation of a specific set of resources and functionality at a real manufacturing device and a mapping of this abstract representation to the physical and functional aspects of the real manufacturing device.

3.4.46 VMD-specific:

An adjective used to describe an object whose name has a scope that is a single VMD (i.e. the name may be referenced by all application associations established with the VMD).

4 Abbreviations

AA : application association
ACSE : Association Control Service Element
AE : application entity
AP : application process
APDU : application protocol data unit
ASE : application service element
ASN.1 : Abstract Syntax Notation One
CBB : conformance building block
FRSM : file read state machine

FTAM : File Transfer, Access and Management
MMPM : Manufacturing Message Protocol Machine
MMS : Manufacturing Message Specification
NC : Numerical Control
OCS : operator communication services
OSI : Open Systems Interconnection
PC : Programmable Controller
PDU : protocol data unit
PSAP : presentation service access point
SAP : service access point
SDU : service data unit
ULSM : upload state machine
VMD : Virtual Manufacturing Device
VT : Virtual Terminal

5 Conventions

5.1 Service Conventions

This part of ISO 9506 uses the descriptive conventions contained in the OSI Service Conventions (ISO TR 8509). The model defines the interactions between the MMS-user and the MMS-provider. Information is passed between an MMS-user and an MMS-provider by service primitives, which may convey parameters.

5.2 Base of Numeric Values

This part of ISO/IEC 9506 uses a decimal representation for all numeric values unless otherwise noted.

5.3 Notation

This part of ISO/IEC 9506 uses the abstract syntax notation defined in ISO 8824 (ASN.1 Specification). In keeping with the intent and requirements of the ASN.1 Standard, all type reference symbols begin with an upper case letter. All value references begin with a lower case letter.

5.4 Supporting Productions

Supporting productions introduced in the various clauses of this part of ISO/IEC 9506 are described where they are referenced if they are used primarily in one place. When supporting productions are referenced multiple times from different places, they are collected at the end of the most relevant clause. In any case, an index of productions with page numbers may be found at the end of this part of ISO/IEC 9506.

5.5 Pass-through Parameters

Many of the parameters of the various MMS services are passed from the request primitive via the service's request PDU to the indication primitive or from the response primitive via the service's response PDU to the confirm primitive, without other action being taken by the MMS-provider relative to the parameter.

5.5.1 Pass-through Request Parameters

The type identified by the type reference name shall be the parameter of the same name (see 5.5) from the service's request primitive, and shall appear as the parameter of the same name in the service's indication primitive, if issued. The value of the parameter in the request primitive, indication primitive, and the request PDU are semantically equivalent.

If the parameter is optional and it is omitted from the request service primitive, it shall be absent in the request PDU. If an optional parameter is absent in the request PDU, it shall be absent in the indication service primitive.

If a parameter has a default in the request PDU and this default value is provided in the request service primitive, then the parameter may be absent in the request PDU. If a parameter has a default value in the request PDU and this parameter is absent in the request PDU, then the parameter shall specify the default value in the indication service primitive.

5.5.2 Pass-through Response Parameters

The type identified by the type reference name shall be the parameter of the same name (see 5.5) from the service's response primitive, and shall appear as the parameter of the same name in the service's confirm primitive, if issued. The value of the parameter in the response primitive, confirm primitive, and the response PDU shall be semantically equivalent.

If the parameter is optional and it is omitted from the response service primitive, it shall be absent in the response PDU. If an optional parameter is absent in the response PDU, it shall be absent in the confirm service primitive.

If a parameter has a default in the response PDU and this default value is provided in the response service primitive, then the parameter may be absent in the response PDU. If a parameter has a default value in the response PDU and this parameter is absent in the response PDU, then the parameter shall specify the default value in the confirm service primitive.

5.5.3 Enumerated Values in Parameters

For those parameters in the service description that have enumerated values, the value specified for the corresponding protocol parameter shall be the value of the same name (see 5.5) from the service primitive containing the parameter. The values conveyed in the service primitive, resulting PDU, and the service primitive that results from receipt of the service primitive shall be semantically equivalent.

NOTE – The correspondence between such values is identified in this part of ISO/IEC 9506 through the use of the same names in the service primitives and protocol. In the service specification, such values are specified in all upper case characters. In the protocol specification, the case of the name is chosen so as to satisfy ASN.1 syntax requirements, with the name in upper case characters following the usage in the protocol in a comment.

5.6 Negative Confirmation

Most confirmed MMS services provide for negative confirmation in the case that an error occurs in the processing of the service request by the responding MMS-user. Such negative confirmation shall be indicated by a Result(-) parameter and an "ErrorType" parameter in the service's response primitive. A Result(-) parameter and an "ErrorType" parameter that is semantically equivalent to those parameters in the response primitive shall appear in the confirm service primitive.

The abstract syntax for a negative confirmation shall be the ErrorPDU of the service, with the "error" field derived from the "Problem" parameter in the response service primitive.

5.7 Modifiers to a Service Request

MMS services allow modifiers to be used with instances of service requests.

In instances of requests of services which make use of modifiers, the modifiers specified in a RequestPDU shall be semantically equivalent to, and in the same order as, those modifiers specified in the request service primitive. The indication primitive shall contain a list of modifiers that is semantically equivalent to, and in the same order as, the modifiers in the RequestPDU.

5.8 Presentation of Errors

For each service presented in the body of this part of ISO/IEC 9506, the errors that may result from the use of that service are not presented with the protocol for the service. Errors are specified in a separate clause.

5.9 Use of Companion Standard Fields

In a number of places in the MMS protocol, a parameter is specified as for use by Companion Standards. This allows the definition of the content of this field to be specified via some means outside the scope of this part of ISO/IEC 9506. Use of such fields is specified in clause 19.

5.10 Calling and Called MMS-user

This part of ISO/IEC 9506 makes use of the terms Calling and Called MMS-user. The Calling MMS-user is the MMS-user that issues the Initiate.request service primitive. The Called MMS-user is the MMS-user that issues the Initiate.response service primitive.

NOTE – The use of the term "called" in MMS is not the same as the general usage of the term in OSI. The MMS usage of the term "called" corresponds to the OSI usage of the term "responding". This distinction has been introduced in order to avoid confusion with the Requesting/Responding MMS-user definition given below.

5.11 Sending and Receiving MMS-user and MMPM

This part of ISO/IEC 9506 makes use of the terms Sending and Receiving MMS-user. The Sending MMS-user is the MMS-user that issues a request or response service primitive. The Receiving MMS-user is the MMS-user that receives an indication or confirmation service primitive.

NOTE – It is important to note that, in the course of completion of a confirmed MMS service, both MMS-users will be senders and receivers at one time. The first MMS-user sends the request and receives the confirmation, while the second MMS-user receives the indication and sends the response.

This part of ISO/IEC 9506 makes use of the terms Sending and Receiving MMPM. The Sending MMPM is the MMPM that sends an MMS PDU. The Receiving MMPM is the MMPM that receives an MMS PDU.

5.12 Requesting and Responding MMS-user

This part of ISO/IEC 9506 makes use of the terms Requesting and Responding MMS-user. The Requesting MMS-user is the MMS-user that issues the request service primitive for a service, while the Responding MMS-user is the MMS-user that issues the response service primitive for a service.

NOTE – It is important to note that the use of the term Responding MMS-user differs from the use of the term Responding entity in ACSE and other standards. In those standards, the term is used to reference the entity that responds to a connection request.

5.13 Client and Server of a Service

This part of ISO/IEC 9506 makes use of the terms Client and Server in order to describe the model of the MMS VMD. The Server is defined as the peer communicating entity which behaves as a VMD for a particular service request instance. The Client is the peer communicating entity which makes use of the VMD for some particular purpose via a service request instance. The VMD model is primarily useful in describing the actions of the Server, and thus in describing the commands and responses that a Client may use. A real end system may adopt the Client role, or the Server role, or both during the lifetime of an application association. Use of MMS in the OSI environment is further described in clause 6.

5.14 ASN.1 Definitions

All ASN.1 definitions provided in this part of ISO/IEC 9506, clauses 7 to 16, and annexes B and C, are part of the ASN.1 Module "MMS-General-Module-1". All ASN.1 definitions provided in this part of ISO/IEC 9506, clause 19, are part of the ASN.1 Module "ISO-9506-MMS-1". The beginning and closing statements indicating that each ASN.1 definition provided is a part of this module is omitted in order to make reading of the document easier. Each ASN.1 definition provided implicitly contains the statement:

```
<ModuleName> DEFINITIONS ::= BEGIN
```

at the beginning of the definition and contains the keyword "END" at the end of the definition, where <ModuleName> is the name of the ASN.1 Module of which the definition forms a part.

NOTE – ISO-9506-MMS-1 represents major revision number 1 of the MMS core abstract syntax provided by this part of this part of ISO/IEC 9506.

MMS-General-Module-1 represents major revision number 1 of the MMS general definitions, which are referenced by the Module ISO-9506-MMS-1 and all MMS Companion Standards.

6 Elements of Protocol Procedure

This clause describes the elements of protocol procedure related to the sending and receiving of MMS PDUs and their relation to service primitive events at the MMS-user to MMS-provider boundary.

6.1 Descriptive Conventions

The figures in this clause use a standard state diagram descriptive mechanism. The following text summarizes this mechanism. Note that all state diagrams are shown from the viewpoint of the MMS-provider.

Each state is represented by a box. The name of the state is shown inside the box. Each arrow represents a transition into or out of a state. The head of the arrow points to the output state, which is the state entered as a result of the transition.

Each transition is labeled with the input action that causes the transition, and the output actions to be taken upon the transition. The inputs are shown above the outputs, and are separated from the outputs by a solid horizontal line.

Service primitives to which a "+" is appended indicate a service primitive containing a Result(+) parameter. Service primitives to which a "-" is appended indicate a service primitive containing a Result(-) parameter.

6.2 Entering and Leaving the MMS Environment

The initiate, conclude, and abort services provide the mechanisms for entering and leaving the MMS environment. The model for these services (which describes allowable sequences of events) is described in ISO/IEC 9506-1, clause 8.

6.3 Operating in the MMS Environment

Once in the MMS environment, there may be a number of services outstanding at any instant in time. ISO/IEC 9506 describes the state diagram for each such service request instance independently.

NOTE – Other clauses of this part of ISO/IEC 9506 define additional limitations on allowable sequences of service primitives, and may further restrict the MMS-user.

6.3.1 Confirmed MMS Services

This clause describes the state transitions for all confirmed services that may be invoked within the MMS environment. This set of services consists of all services that are requested through the use of the Confirmed-RequestPDU.

The state transition diagrams in Figure 2 and Figure 3 are applicable for each of the above services, and are applied separately to each instance of each service request. Multiple concurrent service request instances may be outstanding at any given instant in time, subject to sequencing rules stated in other clauses of this Standard.

All PDUs associated with the execution of a single MMS confirmed service instance (these PDUs are the Confirmed-RequestPDU, the Confirmed-ResponsePDU, the Confirmed-ErrorPDU, the Cancel-RequestPDU, the Cancel-ResponsePDU, the Cancel-ErrorPDU, and the RejectPDU) shall be sent in the same presentation context.

6.3.1.1 The Service Requester

NOTE 1 – The order of receipt of the Cancel-ResponsePDU and Confirmed-ErrorPDU(x) in transition 5 of Figure 2 shall not be relevant.

Figure 2 depicts the progression of a confirmed MMS service request from the service requester's point of view. Before the service request primitive is issued, the service is considered to be in the "Requester Idle" state. Upon receipt of a request primitive for any of the MMS confirmed services, the MMS-provider sends a Confirmed-RequestPDU (specifying the invoke ID that unambiguously identifies the service request instance on the application association) and enters the state "Service Pending Requester".

Upon receipt of a Confirmed-ResponsePDU specifying the service previously requested and the invoke ID that specifies the service instance, the MMS-provider issues a confirmation service primitive (specifying the service type and the invoke ID previously requested) to the MMS-user containing a Result(+) parameter. A state transition into the "Requester Idle" state then occurs.

Upon receipt of a Confirmed-ErrorPDU specifying the previously requested service and the invoke ID that specifies the service instance, the MMS-provider issues a confirmation service primitive (specifying the service type and the invoke ID previously requested) to the MMS-user containing a Result(-) parameter. A state transition into the "Requester Idle" state then occurs.

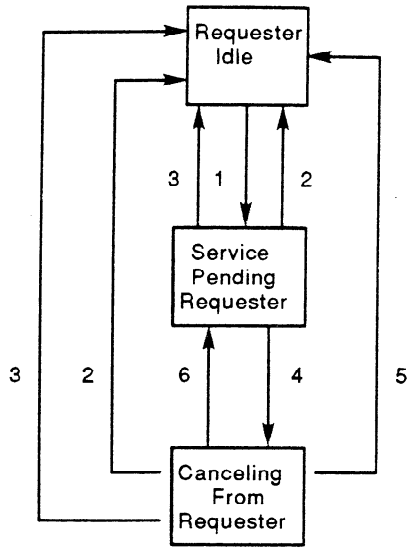
Upon receipt of a cancel request service primitive from the MMS-user, the MMS-provider sends a Cancel-RequestPDU containing the invoke ID of the service request to be cancelled (this information is obtained from the cancel request primitive parameters). The state "Canceling From Requester" is then entered.

The state "Canceling From Requester" is exited upon receipt of any one of four possible input actions. These are described below.

If a Cancel-ErrorPDU is received which specifies an invoke ID that matches the proper instance of the cancel service request, the MMS-provider issues a cancel confirm service primitive to the MMS-user containing a Result(-) parameter and returns to the "Service Pending Requester" state. In this case, the cancel request is considered to have failed.

In the case of a successful cancel request, the following events occur:

a) a Cancel-ResponsePDU is received whose invoke ID matches the proper instance of the cancel service request;



Transitions:

1 - x.request / Confirmed-RequestPDU(x)

2 - Confirmed-ResponsePDU(x) / x.confirm+

3 - Confirmed-ErrorPDU(x) / x.confirm-

4 - cancel.request / Cancel-RequestPDU

5 - Cancel-ResponsePDU and Confirmed-ErrorPDU(x) / cancel.confirm+ and x.confirm-

6 - Cancel-ErrorPDU / cancel.confirm-

Figure 2 – Confirmed Service Request as seen by the Service Requester

b) a Confirmed-ErrorPDU is received which specifies the service type of the service being cancelled and the invoke ID matches that of the service being cancelled;

c) the MMS-provider issues a cancel confirm service primitive to the MMS-user containing the Result(+) parameter and a confirm service primitive for the service being cancelled containing a Result(-) parameter (and specifying the cause as cancellation);

d) the MMS-provider transitions to the "Requester Idle" state.

If a Confirmed-ResponsePDU is received which specifies the service type of the service being cancelled and the invoke ID matches that of the service being cancelled, the MMS-provider issues a confirm service primitive containing a Result(+) parameter for the service that was in the process of being cancelled. In this case, the cancel request is considered to have failed, and a Cancel-ErrorPDU will be received for the cancel service invocation.

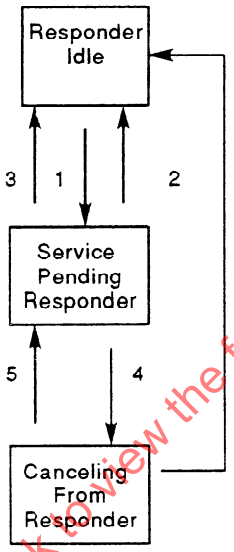
NOTE 2 – This case generally occurs when the Confirmed-ResponsePDU for the service being cancelled and the cancel-RequestPDU are issued simultaneously by the two MMS-users in a two-way simultaneous dialogue.

If a Confirmed-ErrorPDU is received which specifies the service type of the service being cancelled whose invoke ID matches that of the service being cancelled, and the cause for the error is not error class SERVICE-PREEMPT and error code CANCEL, the MMS-provider issues a confirm service primitive containing the Result(-) parameter for the service that was in the process of being cancelled. In this case, the cancel request is considered to have failed, and a Cancel-ErrorPDU will be received for the cancel service invocation.

NOTE 3 – This case generally occurs when the Confirmed-ErrorPDU for the service being cancelled and the Cancel-RequestPDU are issued simultaneously by the two MMS-users in a two-way simultaneous dialogue.

The handling of erroneous cancels is described in 6.4.

6.3.1.2 The Service Responder



Transitions:

1 – Confirmed–RequestPDU(x) / x.indication

2 – x.response+ / Confirmed–ResponsePDU(x)

3 – x.response– / Confirmed–ErrorPDU(x)

4 – Cancel–RequestPDU / Cancel.indication

5 – Cancel.response– / Cancel–ErrorPDU

6 – Cancel.response+ and x.response– / Cancel–ResponsePDU and Confirmed–ErrorPDU(x)

Figure 3 – Confirmed Service Request as seen by the Service Responder

NOTE 1 – The order in which the Cancel.response+ and x.response- service primitives are issued in transition 6 of Figure 3 shall not be relevant.

Figure 3 depicts the progression of a confirmed MMS service request from the service responder's point of view. Before the service Confirmed-RequestPDU is received, the service is considered to be in the "Responder Idle" state. Upon receipt of a Confirmed-RequestPDU for any of the confirmed services identified above, the MMS-provider issues an indication primitive (specifying the particular service being requested and an invoke ID that specifies the service instance) and enters the state "Service Pending Responder".

Upon receipt of a response service primitive containing a Result(+) parameter (specifying the service previously indicated and an invoke ID that specifies the service instance), the MMS-provider sends a Confirmed-ResponsePDU (specifying the service type and the invoke ID from the response primitive). A state transition into the "Responder Idle" state then occurs.

Upon receipt of a response service primitive containing a Result(-) parameter (specifying the service previously indicated and an invoke ID that specifies the service instance), the MMS-provider sends a Confirmed-ErrorPDU (specifying the service type and the invoke ID from the response primitive). A state transition into the "Responder Idle" state then occurs.

Upon receipt of a Cancel-RequestPDU specifying the invoke ID of the matching service instance, the MMS-provider issues a cancel indication service primitive specifying the invoke ID of the service request to be cancelled (this information is obtained from the Cancel-RequestPDU parameters). The state "Canceling Service Responder" is then entered.

NOTE 2 – Actions to be taken upon receipt of a Cancel-RequestPDU whose invoke ID does not match any outstanding service instance are specified in 6.4.

The state "Canceling Service Responder" is exited upon receipt of either one of two possible input actions. These are described in the next two paragraphs.

When a cancel request succeeds at the responding MMS-user, the following sequence of events occurs:

a) the responding MMS-user issues a cancel response specifying the invoke ID of the matching service instance and containing a Result(+) parameter to the MMS-provider, and issues a response service primitive containing a Result(-) parameter (specifying the error class SERVICE and error code CANCEL) for the service being cancelled;

b) the MMS-provider sends a Cancel-ResponsePDU and a Confirmed-ErrorPDU specifying the service instance cancelled (with error class SERVICE and error code CANCEL);

c) the MMS-provider returns to the "Responder Idle" state.

The MMS-user shall not issue a cancel response service primitive containing a Result(+) parameter without also issuing a response service primitive containing a Result(-) parameter that specifies the error class SERVICE-PREEMPT and error code CANCEL. Conversely, the MMS-user shall not issue a response service primitive containing a Result(-) parameter that specifies the error class SERVICE-PREEMPT and error code CANCEL without also issuing a cancel response service primitive containing a Result(+) parameter. Hence, these two events logically occur together.

If a cancel response specifying the invoke ID of the matching service instance containing a Result(-) parameter is received, the MMS-provider sends a Cancel-ErrorPDU and returns to the "service pending" state. In this case, the cancel request is considered to have failed.

NOTE 3 – The handling of erroneous cancel requests and invalid PDUs is described in 6.4.

6.3.2 Unconfirmed MMS Services

This clause describes the operation of unconfirmed MMS services. This set of services is defined as those services that make up the UnconfirmedService choice defined in clause 7.

The state transition diagrams in Figures 4 and 5 are applicable for each of the above identified services, and are applied separately to each instance of each service request.

6.3.2.1 The Service Requester

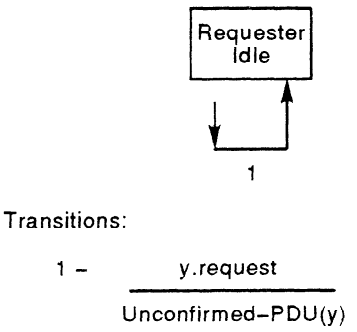


Figure 4 – Unconfirmed Service as seen by the Service Requester

Figure 4 depicts the progression of an unconfirmed MMS service from the service requester's point of view. Before the service request primitive is issued, the service is considered to be in the "Requester Idle" state. Upon receipt of a request primitive for any of the above unconfirmed services, the MMS-provider sends an Unconfirmed-PDU (specifying the particular service being requested) and transitions back to the "Requester Idle" state.

For unconfirmed MMS services, no response PDU or error PDU will be received. Further, it is not possible to cancel an unconfirmed MMS service.

6.3.2.2 The Service Responder

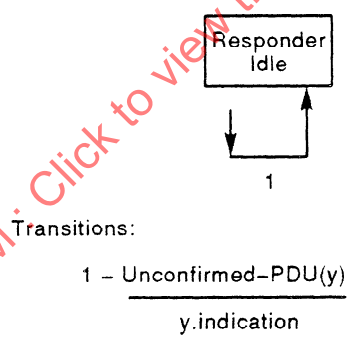


Figure 5 – Unconfirmed Service as seen by the Service Responder

Figure 5 depicts the progression of an unconfirmed MMS service from the service responder's point of view. Before the Unconfirmed-PDU is received, the service is considered to be in the "Responder Idle" state. Upon receipt of a request primitive for any of the above unconfirmed services, the MMS-provider issues an indication service primitive (specifying the particular service being requested based upon information in the Unconfirmed-PDU received) and transitions from the "Responder Idle" state to the "Responder Idle" state (into the same state).

For unconfirmed MMS services, the MMS-user may not issue any response primitive. Further, it is not possible to cancel an unconfirmed MMS service.

6.3.3 The Cancel Service

The Cancel service, while it is a confirmed service, does not operate in the same manner as other confirmed services. When the Cancel service is invoked, no new state machine is created. Rather, the state machine of the service being cancelled is affected. A Cancel service request cannot be cancelled by another invocation of the Cancel service. The invoke ID specified in the Cancel service request may not be that of another invocation of the Cancel service, since the Cancel service only operates on those services that make up the ConfirmedServiceRequest choice defined in clause 7.

At any given instant, only one Cancel service invocation may be outstanding for any given service request instance. An invocation of the Cancel service does not affect the limit on the number of service requests that may be outstanding at any given instant as negotiated by the initiate service. Cancel service requests are not counted in determining if the limit has been reached.

The operation of the Cancel service is described in the previous clauses of this document describing the service requester and service responder for MMS confirmed services.

6.4 Handling of Error Conditions

Upon receipt of an invalid PDU, the MMS-provider shall issue a reject indication service primitive to the MMS-user identifying the error detected, and shall send a RejectPDU to the system from which the invalid PDU was received. In this case, no state change shall occur.

Upon receipt of a Cancel-RequestPDU that attempts to cancel an unknown service request, for example, when the invoke ID specified is not an outstanding confirmed service, the MMS-provider shall send a Cancel-ErrorPDU to the sender of the Cancel request. In this case, the MMS-user is not notified of the erroneous cancel attempt.

NOTE 1 – It is possible for a Cancel-RequestPDU and a Confirmed-ResponsePDU or Confirmed-ErrorPDU for the service requested to be cancelled to be issued concurrently by the two communicating MMS-users. Thus, one side considers the service to have completed, while the other considers it to be awaiting cancellation. In this case, the Cancel request fails and the service completes normally.

Upon receipt of a Cancel-ErrorPDU, where the state machine referred to by the invoke ID of the service to be cancelled is in the "Requester Idle" state, the MMS-provider issues a cancel confirm- service primitive to the MMS-user.

NOTE 2 – This case occurs when a Confirmed-ResponsePDU or Confirmed-ErrorPDU for the service to be cancelled passes a Cancel-RequestPDU for that service.

6.5 The Reject Service and RejectPDU

The reject service is used to notify an MMS-user of a protocol error that has occurred. The operation of this service is described in ISO/IEC 9506-1, clauses 8 and 17.

NOTE – The actions to be taken by an MMS-user upon receipt of a reject indication service primitive are a local matter. It is important to note that, as a result of the possibility that a request, response, or error may be rejected, the two MMS-users involved in a dialogue may not have a common understanding of the state of outstanding transaction objects (see ISO/IEC 9506-1, clause 7). The abort service may be used at any time by an MMS-user to terminate the MMS-environment and the application association.

7 MMS PDU

This clause describes the PDUs used to operate the MMS protocol. The mapping of these PDUs to underlying services is described in clause 17. The mapping of MMS services to these PDUs is described in clauses 8 to 16.

This module defines all the common syntax to MMS and its Companion Standards.

```
MMS-General-Module-1 {iso standard 9506 part(2) mms-general-module-version1(2)}
DEFINITIONS ::= BEGIN

    EXPORTS MMSpdu;
    -- By this statement we imply that all productions subordinate
    -- to MMSpdu are exported also.

    IMPORTS AP-title,AP-invocation-id, AE-qualifier, AE-invocation-id
      FROM  ISO-8650-ACSE-1
        {iso standard 8650 abstract-syntax(2) acse-pdi(1)};

    MMSpdu ::= CHOICE {
         confirmed-RequestPDU     [0] IMPLICIT Confirmed-RequestPDU,
         confirmed-ResponsePDU    [1] IMPLICIT Confirmed-ResponsePDU,
         confirmed-ErrorPDU       [2] IMPLICIT Confirmed-ErrorPDU,
         unconfirmed-PDU          [3] IMPLICIT Unconfirmed-PDU,
         rejectPDU                [4] IMPLICIT RejectPDU,
         cancel-RequestPDU        [5] IMPLICIT Cancel-RequestPDU,
         cancel-ResponsePDU       [6] IMPLICIT Cancel-ResponsePDU,
         cancel-ErrorPDU          [7] IMPLICIT Cancel-ErrorPDU,
         initiate-RequestPDU      [8] IMPLICIT Initiate-RequestPDU,
         initiate-ResponsePDU     [9] IMPLICIT Initiate-ResponsePDU,
         initiate-ErrorPDU       [10] IMPLICIT Initiate-ErrorPDU,
         conclude-RequestPDU     [11] IMPLICIT Conclude-RequestPDU,
         conclude-ResponsePDU    [12] IMPLICIT Conclude-ResponsePDU,
         conclude-ErrorPDU       [13] IMPLICIT Conclude-ErrorPDU
         }
```

There are fourteen types of PDUs in MMS. The Initiate-RequestPDU, the Initiate-ResponsePDU, the Initiate-ErrorPDU, the Conclude-RequestPDU, the Conclude-ResponsePDU, the Conclude-ErrorPDU, the RejectPDU, the Cancel-RequestPDU, the Cancel-ResponsePDU, and the Cancel-ErrorPDU are defined in clause 8. The remaining PDU types are defined in 7.1 to 7.4.

7.1 The Confirmed-RequestPDU

```
Confirmed-RequestPDU ::= SEQUENCE {
     invokeID Unsigned32,
     listOfModifier SEQUENCE OF Modifier OPTIONAL,
     ConfirmedServiceRequest,
     [79] CS-Request-Detail OPTIONAL
     -- shall not be transmitted if value is NULL
     }
```

The Confirmed-RequestPDU is a sequence containing four elements, an invokeID, an optional list of modifiers, a ConfirmedServiceRequest, and a CS-Request-Detail.

The invokeID shall be a 32-bit unsigned integer, which shall unambiguously identify a service request among all outstanding confirmed service requests from a particular MMS-user on a given application association. At any instant in time, there shall be at most one service request outstanding from a particular MMS-user on an application association for any given invokeID. The value of the invokeID shall be the value provided by the MMS-user in the request service primitive (see ISO/IEC 9506-1, clause 5). The invokeID provided in the Confirmed-ResponsePDU and Confirmed-ErrorPDU allows the MMS-provider and the MMS-user to correlate these PDUs with the appropriate service request.

The list of modifiers shall be used to prescribe modifiers to the execution of service requests. A modifier specified in a list of modifiers shall be successfully executed before the next modifier in the list of modifiers or before execution of the service request. The order of the modifiers in the list is therefore important. If no list of modifiers is present, then service request execution may begin immediately upon receipt of the request and without pre-conditions.

The ConfirmedServiceRequest shall be used to identify a confirmed service and the argument for that confirmed service. This parameter is further described in 7.5.2.

The CS-Request-Detail shall be an additional parameter to be supplied by a Companion Standard or the abstract syntax defined in clause 19. The identifier of the CHOICE in this parameter shall be identical to the identifier of the CHOICE in the ConfirmedServiceRequest. This parameter is further described in 7.5.2.

7.2 The Unconfirmed-PDU

```
Unconfirmed-PDU ::= SEQUENCE {
     UnconfirmedService,
     [79] CS-Unconfirmed-Detail OPTIONAL
     -- shall not be transmitted if value is NULL
     }
```

The Unconfirmed-PDU shall be a sequence containing an UnconfirmedService and a CS-Unconfirmed-Detail.

The UnconfirmedService shall be used to identify an unconfirmed service and its argument.

The CS-Unconfirmed-Detail shall be an additional parameter to be supplied by a Companion Standard or the abstract syntax defined in clause 19. The identifier of the CHOICE in this parameter shall be identical to the identifier of the CHOICE in the UnconfirmedService. This parameter is further described in 7.5.3.

7.3 The Confirmed-ResponsePDU

```
Confirmed-ResponsePDU ::= SEQUENCE {
     invokeID Unsigned32,
     ConfirmedServiceResponse,
     [79] CS-Response-Detail OPTIONAL
     -- shall not be transmitted if value is NULL
     }
```

The Confirmed-ResponsePDU shall be a sequence containing three elements, an invokeID, a ConfirmedServiceResponse and a CS-Response-Detail.

The invokeID shall be a 32-bit unsigned integer that shall unambiguously identify a service request among all outstanding confirmed service requests from a particular MMS-user on an application association. At any instant in time, there may be at most one service request outstanding from a particular MMS-user on an application association for any given invokeID. The value of the invokeID shall be the value provided by the MMS-user in the response service primitive (see ISO/IEC 9506-1, clause 5), and shall identify the request instance that caused the service to be carried out. The invokeID in this PDU allows the MMS-provider and MMS-user to correlate this PDU with the appropriate service request.

The ConfirmedServiceResponse shall be used to identify a confirmed service and the response for that confirmed service. This parameter is further described in 7.5.4.

The CS-Response-Detail shall be an additional parameter to be supplied by a Companion Standard or the abstract syntax defined in clause 19. The identifier of the CHOICE in this parameter shall be identical to the identifier of the CHOICE in the ConfirmedServiceResponse. This parameter is further described in 7.5.4.

7.4 The Confirmed-ErrorPDU

```
Confirmed-ErrorPDU ::= SEQUENCE {
     invokeID          [0] IMPLICIT Unsigned32,
     modifierPosition  [1] IMPLICIT Unsigned32 OPTIONAL,
     serviceError      [2] IMPLICIT ServiceError
     }
```

The Confirmed-ErrorPDU shall be a sequence containing three elements, an invokeID, an optional modifierPosition, and a ServiceError.

The invokeID shall be a 32-bit unsigned integer that shall unambiguously identify a service request among all outstanding confirmed service requests from a particular MMS-user on an application association. At any instant in time, there may be at most one service request outstanding from a particular MMS-user on an application association for any given invokeID. The value of the invokeID shall be the value provided by the MMS-user in the response service primitive (see ISO/IEC 9506-1, clause 5), and shall identify the request instance that caused the service to be carried out. The invokeID in this PDU allows the MMS-provider and MMS-user to correlate this PDU with the appropriate service request.

The modifierPosition shall be a 32-bit unsigned integer that shall unambiguously identify a modifier among all modifiers which were specified in the listOfModifier sequence in the Confirmed-RequestPDU identified by the invokeID. This parameter is derived from the Modifier Position sub-parameter in the Service Error parameter from the response service primitive (see ISO/IEC 9506-1, clause 17).

The ServiceError shall be used to identify the error class and error code for either the modifier of the confirmed service, or the confirmed service. The ServiceError parameter is described further in 7.5.5.

7.5 Supporting Productions

The following supporting productions provide definitions used in specifying the MMS PDUs.

7.5.1 Modifier

```
Modifier ::= CHOICE {
     attachToEventCondition [0] IMPLICIT AttachToEventCondition,
     attachToSemaphore      [1] IMPLICIT AttachToSemaphore
     }
```

The Modifier parameter shall be a choice of one of the MMS modifiers. Each modifier shall serve to modify the normal execution of an MMS confirmed service by placing a pre-condition on the execution of a service request instance which shall be successfully executed before service execution can begin. The effect of a modifier is modeled by the service invocation state machine provided in clause 6. Definitions of modifiers may be found in the protocol descriptions in the following clauses of this part of ISO/IEC 9506.

7.5.2 ConfirmedServiceRequest

```
ConfirmedServiceRequest ::= CHOICE {
  status                          [0] IMPLICIT Status-Request,
  getNameList                     [1] IMPLICIT GetNameList-Request,
  identify                        [2] IMPLICIT Identify-Request,
  rename                          [3] IMPLICIT Rename-Request,
  read                            [4] IMPLICIT Read-Request,
  write                           [5] IMPLICIT Write-Request,
  getVariableAccessAttributes [6] GetVariableAccessAttributes-Request,
  defineNamedVariable             [7] IMPLICIT DefineNamedVariable-Request,
  defineScatteredAccess           [8] IMPLICIT DefineScatteredAccess-Request,
  getScatteredAccessAttributes [9] GetScatteredAccessAttributes-Request,
  deleteVariableAccess            [10] IMPLICIT DeleteVariableAccess-Request,
  defineNamedVariableList         [11] IMPLICIT DefineNamedVariableList-Request,
  getNamedVariableListAttributes [12]
                                      GetNamedVariableListAttributes-Request,
  deleteNamedVariableList     [13] IMPLICIT DeleteNamedVariableList-Request,
  defineNamedType             [14] IMPLICIT DefineNamedType-Request,
  getNamedTypeAttributes      [15]          GetNamedTypeAttributes-Request,
  deleteNamedType             [16] IMPLICIT DeleteNamedType-Request,
  input                       [17] IMPLICIT Input-Request,
  output                      [18] IMPLICIT Output-Request,
  takeControl                 [19] IMPLICIT TakeControl-Request,
  relinquishControl           [20] IMPLICIT RelinquishControl-Request,
  defineSemaphore             [21] IMPLICIT DefineSemaphore-Request,
  deleteSemaphore             [22]          DeleteSemaphore-Request,
  reportSemaphoreStatus       [23]          ReportSemaphoreStatus-Request,
  reportPoolSemaphoreStatus   [24] IMPLICIT
                                      ReportPoolSemaphoreStatus-Request,
  reportSemaphoreEntryStatus [25] IMPLICIT
                                      ReportSemaphoreEntryStatus-Request,
  initiateDownloadSequence    [26] IMPLICIT
                                      InitiateDownloadSequence-Request,
  downloadSegment             [27] IMPLICIT DownloadSegment-Request,
  terminateDownloadSequence   [28] IMPLICIT
                                      TerminateDownloadSequence-Request,
  initiateUploadSequence      [29] IMPLICIT InitiateUploadSequence-Request,
  uploadSegment               [30] IMPLICIT UploadSegment-Request,
  terminateUploadSequence     [31] IMPLICIT TerminateUploadSequence-Request,
  requestDomainDownload       [32] IMPLICIT RequestDomainDownload-Request,
  requestDomainUpload         [33] IMPLICIT RequestDomainUpload-Request,
  loadDomainContent           [34] IMPLICIT LoadDomainContent-Request,
  storeDomainContent          [35] IMPLICIT StoreDomainContent-Request,
  deleteDomain                [36] IMPLICIT DeleteDomain-Request,
  getDomainAttributes         [37] IMPLICIT GetDomainAttributes-Request,
  createProgramInvocation     [38] IMPLICIT CreateProgramInvocation-Request,
  deleteProgramInvocation     [39] IMPLICIT DeleteProgramInvocation-Request,
  start                       [40] IMPLICIT Start-Request,
  stop                        [41] IMPLICIT Stop-Request,
  resume                      [42] IMPLICIT Resume-Request,
  reset                       [43] IMPLICIT Reset-Request,
  kill                        [44] IMPLICIT Kill-Request,
  getProgramInvocationAttributes [45] IMPLICIT
                                      GetProgramInvocationAttributes-Request,
  obtainFile                  [46] IMPLICIT ObtainFile-Request,
  defineEventCondition        [47] IMPLICIT DefineEventCondition-Request,
  deleteEventCondition        [48]          DeleteEventCondition-Request,
  getEventConditionAttributes    [49]   GetEventConditionAttributes-Request,
  reportEventConditionStatus     [50]   ReportEventConditionStatus-Request,
  alterEventConditionMonitoring [51] IMPLICIT
                                     AlterEventConditionMonitoring-Request,
  triggerEvent                [52] IMPLICIT TriggerEvent-Request,
  defineEventAction           [53] IMPLICIT DefineEventAction-Request,
  deleteEventAction           [54]          DeleteEventAction-Request,
```

```
    getEventActionAttributes    [55]          GetEventActionAttributes-Request,
    reportEventActionStatus     [56]           ReportEventActionStatus-Request,
    defineEventEnrollment       [57] IMPLICIT DefineEventEnrollment-Request,
    deleteEventEnrollment       [58]          DeleteEventEnrollment-Request,
    alterEventEnrollment        [59] IMPLICIT AlterEventEnrollment-Request,
    reportEventEnrollmentStatus  [60] ReportEventEnrollmentStatus-Request,
    getEventEnrollmentAttributes [61] IMPLICIT
                                       GetEventEnrollmentAttributes-Request,
    acknowledgeEventNotification [62] IMPLICIT
                                       AcknowledgeEventNotification-Request,
    getAlarmSummary              [63] IMPLICIT GetAlarmSummary-Request,
    getAlarmEnrollmentSummary    [64] IMPLICIT
                                       GetAlarmEnrollmentSummary-Request,
    readJournal                  [65] IMPLICIT ReadJournal-Request,
    writeJournal                 [66] IMPLICIT WriteJournal-Request,
    initializeJournal            [67] IMPLICIT InitializeJournal-Request,
    reportJournalStatus          [68]          ReportJournalStatus-Request,
    createJournal                [69] IMPLICIT CreateJournal-Request,
    deleteJournal                [70] IMPLICIT DeleteJournal-Request,
    getCapabilityList            [71] IMPLICIT GetCapabilityList-Request,
    fileOpen                     [72] IMPLICIT FileOpen-Request,
    fileRead                     [73] IMPLICIT FileRead-Request,
    fileClose                    [74] IMPLICIT FileClose-Request,
    fileRename                   [75] IMPLICIT FileRename-Request,
    fileDelete                   [76] IMPLICIT FileDelete-Request,
    fileDirectory                [77] IMPLICIT FileDirectory-Request,
    additionalService            [78] AdditionalService-Request
          -- choices [72] through [77] are reserved for use by services
          -- defined in annex C
          -- choice [79] is reserved
    }

CS-Request-Detail ::= CHOICE {
    status                      [0]  IMPLICIT CS-Status-Request,
    getNameList                 [1]  IMPLICIT CS-GetNameList-Request,
    input                       [17] IMPLICIT CS-Input-Request,
    output                      [18] IMPLICIT CS-Output-Request,
    initiateDownloadSequence    [26] IMPLICIT
                                     CS-InitiateDownloadSequence-Request,
    downloadSegment             [27] IMPLICIT CS-DownloadSegment-Request,
    terminateDownloadSequence   [28] IMPLICIT
                                     CS-TerminateDownloadSequence-Request,
    initiateUploadSequence      [29] IMPLICIT
                                     CS-InitiateUploadSequence-Request,
    uploadSegment               [30] IMPLICIT CS-UploadSegment-Request,
    terminateUploadSequence     [31] IMPLICIT
                                     CS-TerminateUploadSequence-Request,
    requestDomainDownload       [32] IMPLICIT
                                     CS-RequestDomainDownload-Request,
    requestDomainUpload         [33] IMPLICIT CS-RequestDomainUpload-Request,
    loadDomainContent           [34] IMPLICIT CS-LoadDomainContent-Request,
    storeDomainContent          [35] IMPLICIT CS-StoreDomainContent-Request,
    deleteDomain                [36] IMPLICIT CS-DeleteDomain-Request,
    getDomainAttributes         [37] IMPLICIT CS-GetDomainAttributes-Request,
    createProgramInvocation     [38] IMPLICIT
                                     CS-CreateProgramInvocation-Request,
    deleteProgramInvocation     [39] IMPLICIT
                                     CS-DeleteProgramInvocation-Request,
    start                       [40] IMPLICIT CS-Start-Request,
    stop                        [41] IMPLICIT CS-Stop-Request,
    resume                      [42] IMPLICIT CS-Resume-Request,
    reset                       [43] IMPLICIT CS-Reset-Request,
    kill                        [44] IMPLICIT CS-Kill-Request,
```

```
getProgramInvocationAttributes [45] IMPLICIT
                              CS-GetProgramInvocationAttributes-Request,
defineEventCondition          [47] IMPLICIT CS-DefineEventCondition-Request,
deleteEventCondition          [48] IMPLICIT CS-DeleteEventCondition-Request,
getEventConditionAttributes    [49] IMPLICIT
                                 CS-GetEventConditionAttributes-Request,
reportEventConditionStatus     [50] IMPLICIT
                                 CS-ReportEventConditionStatus-Request,
alterEventConditionMonitoring [51] IMPLICIT
                                 CS-AlterEventConditionMonitoring-Request,
triggerEvent                  [52] IMPLICIT CS-TriggerEvent-Request,
defineEventAction             [53] IMPLICIT CS-DefineEventAction-Request,
deleteEventAction             [54] IMPLICIT CS-DeleteEventAction-Request,
getEventActionAttributes      [55] IMPLICIT
                                 CS-GetEventActionAttributes-Request,
reportEventActionStatus       [56] IMPLICIT
                                 CS-ReportEventActionStatus-Request,
defineEventEnrollment         [57] IMPLICIT
                                 CS-DefineEventEnrollment-Request,
deleteEventEnrollment         [58] IMPLICIT
                                 CS-DeleteEventEnrollment-Request,
alterEventEnrollment          [59] IMPLICIT CS-AlterEventEnrollment-Request,
reportEventEnrollmentStatus   [60] IMPLICIT
                                 CS-ReportEventEnrollmentStatus-Request,
getEventEnrollmentAttributes [61] IMPLICIT
                                 CS-GetEventEnrollmentAttributes-Request,
acknowledgeEventNotification [62] IMPLICIT
                                 CS-AcknowledgeEventNotification-Request,
getAlarmSummary               [63] IMPLICIT CS-GetAlarmSummary-Request,
getAlarmEnrollmentSummary     [64] IMPLICIT
                                 CS-GetAlarmEnrollmentSummary-Request,
readJournal                   [65] IMPLICIT CS-ReadJournal-Request,
writeJournal                  [66] IMPLICIT CS-WriteJournal-Request,
initializeJournal             [67] IMPLICIT CS-InitializeJournal-Request,
reportJournalStatus           [68] IMPLICIT
                                 CS-ReportJournalStatus-Request,
createJournal                 [69] IMPLICIT CS-CreateJournal-Request,
deleteJournal                 [70] IMPLICIT CS-DeleteJournal-Request,
getCapabilityList             [71] IMPLICIT CS-GetCapabilityList-Request
}
```

The ConfirmedServiceRequest parameter shall identify the service type and the argument for that service. The context tags provided identify the service type. Definitions for each individual service specify the form of the argument for the service through definition of a type, which is referenced by the ConfirmedServiceRequest production. Each of the services in the ConfirmedServiceRequest choice is a confirmed service.

The AdditionalServiceRequest shall be reserved for definition of additional services by MMS Companion Standards.

NOTE – All such additional services shall meet the following requirements as described in annex A:

a) no additional service shall violate any of the state transition rules provided by ISO/IEC 9506-1 and ISO/IEC 9506-2. This limits Companion Standard to definition of substates within existing MMS states;

b) no additional service shall violate any of the object models provided by ISO/IEC 9506-1;

c) no additional service shall be used to circumvent the spirit and intent of ISO/IEC 9506-1 and ISO/IEC 9506-2.

7.5.3 UnconfirmedService

```
UnconfirmedService ::= CHOICE {
      informationReport   [0] IMPLICIT InformationReport,
      unsolicitedStatus   [1] IMPLICIT UnsolicitedStatus,
      eventNotification   [2] IMPLICIT EventNotification,
      additionalService   [3] AdditionalUnconfirmedService
              }

CS-Unconfirmed-Detail ::= CHOICE {
      unsolicitedStatus   [1] IMPLICIT CS-UnsolicitedStatus,
      eventNotification   [2] IMPLICIT CS-EventNotification
      }
```

The UnconfirmedService parameter shall identify the service type and the argument for that service. The context tags provided identify the service type. Definitions for each individual service specify the form of the argument for the service through definition of a type, which is referenced by the UnconfirmedService production. Each of the services in the UnconfirmedService choice is an unconfirmed service.

The AdditionalUnconfirmedService shall be reserved for definition of additional services by MMS Companion Standards.

NOTE – All such additional services shall meet the following requirements as described in annex A:

a) no additional service shall violate any of the state transition rules provided by ISO/IEC 9506-1 and ISO/IEC 9506-2. This limits Companion Standard to definition of substates within existing MMS states;

b) no additional service shall violate any of the object models provided by ISO/IEC 9506-1;

c) no additional service shall be used to circumvent the spirit and intent of ISO/IEC 9506-1 and ISO/IEC 9506-2.

7.5.4 ConfirmedServiceResponse

```
ConfirmedServiceResponse ::= CHOICE {
  status                        [0] IMPLICIT Status-Response,
  getNameList                   [1] IMPLICIT GetNameList-Response,
  identify                      [2] IMPLICIT Identify-Response,
  rename                        [3] IMPLICIT Rename-Response,
  read                          [4] IMPLICIT Read-Response,
  write                         [5] IMPLICIT Write-Response,
  getVariableAccessAttributes [6] IMPLICIT
                                    GetVariableAccessAttributes-Response,
  defineNamedVariable           [7] IMPLICIT DefineNamedVariable-Response,
  defineScatteredAccess         [8] IMPLICIT DefineScatteredAccess-Response,
  getScatteredAccessAttributes [9] IMPLICIT
                                    GetScatteredAccessAttributes-Response,
  deleteVariableAccess          [10] IMPLICIT DeleteVariableAccess-Response,
  defineNamedVariableList       [11] IMPLICIT
                                    DefineNamedVariableList-Response,
  getNamedVariableListAttributes [12] IMPLICIT
                                    GetNamedVariableListAttributes-Response,
  deleteNamedVariableList       [13] IMPLICIT
                                    DeleteNamedVariableList-Response,
  defineNamedType               [14] IMPLICIT DefineNamedType-Response,
  getNamedTypeAttributes        [15] IMPLICIT GetNamedTypeAttributes-Response,
  deleteNamedType               [16] IMPLICIT DeleteNamedType-Response,
  input                         [17] IMPLICIT Input-Response,
  output                        [18] IMPLICIT Output-Response,
  takeControl                   [19]          TakeControl-Response,
  relinquishControl             [20] IMPLICIT RelinquishControl-Response,
  defineSemaphore               [21] IMPLICIT DefineSemaphore-Response,
  deleteSemaphore               [22] IMPLICIT DeleteSemaphore-Response,
  reportSemaphoreStatus         [23] IMPLICIT ReportSemaphoreStatus-Response,
```

```
reportPoolSemaphoreStatus   [24] IMPLICIT
                                  ReportPoolSemaphoreStatus-Response,
reportSemaphoreEntryStatus [25] IMPLICIT
                                  ReportSemaphoreEntryStatus-Response,
initiateDownloadSequence    [26] IMPLICIT
                                  InitiateDownloadSequence-Response,
downloadSegment             [27] IMPLICIT DownloadSegment-Response,
terminateDownloadSequence   [28] IMPLICIT
                                  TerminateDownloadSequence-Response,
initiateUploadSequence      [29] IMPLICIT InitiateUploadSequence-Response,
uploadSegment               [30] IMPLICIT UploadSegment-Response,
terminateUploadSequence     [31] IMPLICIT
                                  TerminateUploadSequence-Response,
requestDomainDownload       [32] IMPLICIT RequestDomainDownload-Response,
requestDomainUpload         [33] IMPLICIT RequestDomainUpload-Response,
loadDomainContent           [34] IMPLICIT LoadDomainContent-Response,
storeDomainContent          [35] IMPLICIT StoreDomainContent-Response,
deleteDomain                [36] IMPLICIT DeleteDomain-Response,
getDomainAttributes         [37] IMPLICIT GetDomainAttributes-Response,
createProgramInvocation     [38] IMPLICIT
                                  CreateProgramInvocation-Response,
deleteProgramInvocation     [39] IMPLICIT
                                  DeleteProgramInvocation-Response,
start                       [40] IMPLICIT Start-Response,
stop                        [41] IMPLICIT Stop-Response,
resume                      [42] IMPLICIT Resume-Response,
reset                       [43] IMPLICIT Reset-Response,
kill                        [44] IMPLICIT Kill-Response,
getProgramInvocationAttributes [45] IMPLICIT
                                  GetProgramInvocationAttributes-Response,
obtainFile                  [46] IMPLICIT ObtainFile-Response,
defineEventCondition        [47] IMPLICIT DefineEventCondition-Response,
deleteEventCondition        [48] IMPLICIT DeleteEventCondition-Response,
getEventConditionAttributes  [49] IMPLICIT
                                  GetEventConditionAttributes-Response,
reportEventConditionStatus   [50] IMPLICIT
                                  ReportEventConditionStatus-Response,
alterEventConditionMonitoring [51] IMPLICIT
                                  AlterEventConditionMonitoring-Response,
triggerEvent                [52] IMPLICIT TriggerEvent-Response,
defineEventAction           [53] IMPLICIT DefineEventAction-Response,
deleteEventAction           [54] IMPLICIT DeleteEventAction-Response,
getEventActionAttributes    [55] IMPLICIT
                                  GetEventActionAttributes-Response,
reportEventActionStatus     [56] IMPLICIT
                                  ReportEventActionStatus-Response,
defineEventEnrollment       [57] IMPLICIT DefineEventEnrollment-Response,
deleteEventEnrollment       [58] IMPLICIT DeleteEventEnrollment-Response,
alterEventEnrollment        [59] IMPLICIT AlterEventEnrollment-Response,
reportEventEnrollmentStatus  [60] IMPLICIT
                                  ReportEventEnrollmentStatus-Response,
getEventEnrollmentAttributes [61] IMPLICIT
                                  GetEventEnrollmentAttributes-Response,
acknowledgeEventNotification [62] IMPLICIT
                                  AcknowledgeEventNotification-Response,
getAlarmSummary             [63] IMPLICIT GetAlarmSummary-Response,
getAlarmEnrollmentSummary   [64] IMPLICIT
                                  GetAlarmEnrollmentSummary-Response,
readJournal                 [65] IMPLICIT ReadJournal-Response,
writeJournal                [66] IMPLICIT WriteJournal-Response,
initializeJournal           [67] IMPLICIT InitializeJournal-Response,
reportJournalStatus         [68] IMPLICIT ReportJournalStatus-Response,
createJournal               [69] IMPLICIT CreateJournal-Response,
```

```
  deleteJournal                 [70] IMPLICIT DeleteJournal-Response,
  getCapabilityList             [71] IMPLICIT GetCapabilityList-Response,
  fileOpen                      [72] IMPLICIT FileOpen-Response,
  fileRead                      [73] IMPLICIT FileRead-Response,
  fileClose                     [74] IMPLICIT FileClose-Response,
  fileRename                    [75] IMPLICIT FileRename-Response,
  fileDelete                    [76] IMPLICIT FileDelete-Response,
  fileDirectory                 [77] IMPLICIT FileDirectory-Response,
  additionalService             [78] AdditionalService-Response
        -- choices [72] through [77] are reserved for use by services
        -- defined in annex C
        -- choice [79] is reserved
  }


CS-Response-Detail ::= CHOICE {
  status                        [0]  IMPLICIT CS-Status-Response,
  getNameList                   [1]  IMPLICIT CS-GetNameList-Response,
  input                         [17] IMPLICIT CS-Input-Response,
  output                        [18] IMPLICIT CS-Output-Response,
  initiateDownloadSequence      [26] IMPLICIT
                                     CS-InitiateDownloadSequence-Response,
  downloadSegment               [27] IMPLICIT CS-DownloadSegment-Response,
  terminateDownloadSequence     [28] IMPLICIT
                                     CS-TerminateDownloadSequence-Response,
  initiateUploadSequence        [29] IMPLICIT
                                     CS-InitiateUploadSequence-Response,
  uploadSegment                 [30] IMPLICIT CS-UploadSegment-Response,
  terminateUploadSequence       [31] IMPLICIT
                                     CS-TerminateUploadSequence-Response,
  requestDomainDownload         [32] IMPLICIT
                                     CS-RequestDomainDownload-Response,
  requestDomainUpload           [33] IMPLICIT CS-RequestDomainUpload-Response,
  loadDomainContent             [34] IMPLICIT CS-LoadDomainContent-Response,
  storeDomainContent            [35] IMPLICIT CS-StoreDomainContent-Response,
  deleteDomain                  [36] IMPLICIT CS-DeleteDomain-Response,
  getDomainAttributes           [37] IMPLICIT CS-GetDomainAttributes-Response,
  createProgramInvocation       [38] IMPLICIT
                                     CS-CreateProgramInvocation-Response,
  deleteProgramInvocation       [39] IMPLICIT
                                     CS-DeleteProgramInvocation-Response,
  start                         [40] IMPLICIT CS-Start-Response,
  stop                          [41] IMPLICIT CS-Stop-Response,
  resume                        [42] IMPLICIT CS-Resume-Response,
  reset                         [43] IMPLICIT CS-Reset-Response,
  kill                          [44] IMPLICIT CS-Kill-Response,
  getProgramInvocationAttributes [45] IMPLICIT
                                CS-GetProgramInvocationAttributes-Response,
  defineEventCondition          [47] IMPLICIT
                                     CS-DefineEventCondition-Response,
  deleteEventCondition          [48] IMPLICIT
                                     CS-DeleteEventCondition-Response,
  getEventConditionAttributes   [49] IMPLICIT
                                     CS-GetEventConditionAttributes-Response,
  reportEventConditionStatus    [50] IMPLICIT
                                     CS-ReportEventConditionStatus-Response,
  alterEventConditionMonitoring [51] IMPLICIT
                                CS-AlterEventConditionMonitoring-Response,
  triggerEvent                  [52] IMPLICIT CS-TriggerEvent-Response,
  defineEventAction             [53] IMPLICIT CS-DefineEventAction-Response,
  deleteEventAction             [54] IMPLICIT CS-DeleteEventAction-Response,
  getEventActionAttributes      [55] IMPLICIT
```

```
                                    CS-GetEventActionAttributes-Response,
reportEventActionStatus     [56] IMPLICIT
                                    CS-ReportEventActionStatus-Response,
defineEventEnrollment       [57] IMPLICIT
                                    CS-DefineEventEnrollment-Response,
deleteEventEnrollment       [58] IMPLICIT
                                    CS-DeleteEventEnrollment-Response,
alterEventEnrollment        [59] IMPLICIT
                                    CS-AlterEventEnrollment-Response,
reportEventEnrollmentStatus  [60] IMPLICIT
                                    CS-ReportEventEnrollmentStatus-Response,
getEventEnrollmentAttributes [61] IMPLICIT
                                    CS-GetEventEnrollmentAttributes-Response,
acknowledgeEventNotification [62] IMPLICIT
                                    CS-AcknowledgeEventNotification-Response,
getAlarmSummary              [63] IMPLICIT CS-GetAlarmSummary-Response,
getAlarmEnrollmentSummary    [64] IMPLICIT
                                    CS-GetAlarmEnrollmentSummary-Response,
readJournal                  [65] IMPLICIT CS-ReadJournal-Response,
writeJournal                 [66] IMPLICIT CS-WriteJournal-Response,
initializeJournal            [67] IMPLICIT CS-InitializeJournal-Response,
reportJournalStatus          [68] IMPLICIT
                                    CS-ReportJournalStatus-Response,
createJournal                [69] IMPLICIT CS-CreateJournal-Response,
deleteJournal                [70] IMPLICIT CS-DeleteJournal-Response,
getCapabilityList            [71] IMPLICIT CS-GetCapabilityList-Response
}
```

The ConfirmedServiceResponse parameter shall identify the service type and the response for that service. The context tags provided identify the service type. Definitions for each individual service specify the form of the response for the service through definition of a type, which is referenced by the ConfirmedServiceResponse production.

The AdditionalService-Response shall be reserved for definition of additional responses to services by MMS Companion Standards.

NOTE – All such additional services shall meet the following requirements as described in annex A:

a) no additional service shall violate any of the state transition rules provided by ISO/IEC 9506-1 and ISO/IEC 9506-2. This limits Companion Standards to definition of substates within existing MMS states;

b) no additional service shall violate any of the object models provided by ISO/IEC 9506-1 and ISO/IEC 9506-2;

c) no additional service shall be used to circumvent the spirit and intent of ISO/IEC 9506-1 and ISO/IEC 9506-2.

7.5.5 ServiceError

```
ServiceError ::= SEQUENCE {
     errorClass [0] CHOICE {
          vmd-state [0] IMPLICIT INTEGER { -- VMD-STATE,
               other (0),                    -- OTHER
               vmd-state-conflict (1),       -- VMD-STATE-CONFLICT
               vmd-operational-problem (2), -- VMD-OPERATIONAL-PROBLEM
               domain-transfer-problem (3),
                                             -- DOMAIN-TRANSFER-PROBLEM
               state-machine-id-invalid (4) -- STATE-MACHINE-ID-INVALID
               },
          application-reference [1] IMPLICIT INTEGER {
                                        -- APPLICATION REFERENCE
               other (0),                    -- OTHER
               application-unreachable (1), -- APPLICATION-UNREACHABLE
               connection-lost (2),          -- CONNECTION-LOST
               application-reference-invalid (3),
                                        -- APPLICATION-REFERENCE-INVALID
               context-unsupported (4)      -- CONTEXT-UNSUPPORTED
               },
          definition [2] IMPLICIT INTEGER { -- DEFINITION
               other (0),                    -- OTHER
               object-undefined (1),         -- OBJECT-UNDEFINED
               invalid-address (2),          -- INVALID-ADDRESS
               type-unsupported (3),         -- TYPE-UNSUPPORTED
               type-inconsistent (4),        -- TYPE-INCONSISTENT
               object-exists (5),            -- OBJECT-EXISTS
               object-attribute-inconsistent (6)
                              -- OBJECT-ATTRIBUTE-INCONSISTENT
               },
          resource [3] IMPLICIT INTEGER { -- RESOURCE
               other (0),      -- OTHER
               memory-unavailable (1), -- MEMORY-UNAVAILABLE
               processor-resource-unavailable (2),
                              -- PROCESSOR-RESOURCE-UNAVAILABLE
               mass-storage-unavailable (3),
                              -- MASS-STORAGE-UNAVAILABLE
               capability-unavailable (4), -- CAPABILITY-UNAVAILABLE
               capability-unknown (5)       -- CAPABILITY-UNKNOWN
               },
          service [4] IMPLICIT INTEGER {  -- SERVICE
               other (0),                    -- OTHER
               primitives-out-of-sequence (1),
                              -- PRIMITIVES-OUT-OF-SEQUENCE
               object-state-conflict (2),    -- OBJECT-STATE-CONFLICT
               -- Value 3 reserved for further definition
               continuation-invalid (4),     -- CONTINUATION-INVALID
               object-constraint-conflict (5)
                              -- OBJECT-CONSTRAINT-CONFLICT
               },
          service-preempt [5] IMPLICIT INTEGER { -- SERVICE-PREEMPT
               other (0),                    -- OTHER
               timeout (1),                  -- TIMEOUT
               deadlock (2),                 -- DEADLOCK
               cancel (3)                    -- CANCEL
               },
          time-resolution [6] IMPLICIT INTEGER { -- TIME-RESOLUTION
               other (0),                   -- OTHER
               unsupportable-time-resolution (1)
                                            -- UNSUPPORTABLE-TIME-RESOLUTION
               },
          access [7] IMPLICIT INTEGER {        -- ACCESS
               other (0),                    -- OTHER
               object-access-unsupported (1), -- OBJECT-ACCESS-UNSUPPORTED
```

```
            object-non-existent (2),      -- OBJECT-NON-EXISTENT
            object-access-denied (3),     -- OBJECT-ACCESS-DENIED
            object-invalidated (4)        -- OBJECT-INVALIDATED
            },
        initiate [8] IMPLICIT INTEGER {     -- INITIATE
            other (0),                       -- OTHER
              -- Values 1 and 2 are reserved for further definition
            max-services-outstanding-calling-insufficient (3),
                        -- MAX-SERVICES-OUTSTANDING-CALLING-INSUFFICIENT
            max-services-outstanding-called-insufficient (4),
                        -- MAX-SERVICES-OUTSTANDING-CALLED-INSUFFICIENT
            service-CBB-insufficient (5),    -- SERVICE-CBB-INSUFFICIENT
            parameter-CBB-insufficient (6),
                                -- PARAMETER-CBB-INSUFFICIENT
            nesting-level-insufficient (7)
                                -- NESTING-LEVEL-INSUFFICIENT
            },
        conclude [9] IMPLICIT INTEGER {     -- CONCLUDE
            other (0),                      -- OTHER
            further-communication-required (1)
                                -- FURTHER-COMMUNICATION-REQUIRED
            },
        cancel [10] IMPLICIT INTEGER {       -- CANCEL
            other (0),                      -- OTHER
            invoke-id-unknown (1),          -- INVOKE-ID-UNKNOWN
            cancel-not-possible (2)         -- CANCEL-NOT-POSSIBLE
            },
        file [11] IMPLICIT INTEGER {          -- FILE
            other (0),                      -- OTHER
            filename-ambiguous (1),         -- FILENAME-AMBIGUOUS
            file-busy (2),                  -- FILE-BUSY
            filename-syntax-error (3),      -- FILENAME-SYNTAX-ERROR
            content-type-invalid (4),       -- CONTENT-TYPE-INVALID
            position-invalid (5),           -- POSITION-INVALID
            file-access-denied (6),         -- FILE-ACCESS-DENIED
            file-non-existent (7),          -- FILE-NON-EXISTENT
            duplicate-filename (8),         -- DUPLICATE-FILENAME
            insufficient-space-in-filestore (9)
                                -- INSUFFICIENT-SPACE-IN-FILESTORE
            },
        others    [12] IMPLICIT INTEGER,          -- OTHERS
        cs-error [13] CS-Service-Error
        -- shall not be chosen in the abstract syntax defined in
        -- Clause 19 of this Part of this International Standard
        },
    additionalCode              [1] IMPLICIT INTEGER OPTIONAL,
    additionalDescription       [2] IMPLICIT VisibleString OPTIONAL,
    serviceSpecificInformation [3] CHOICE {
        obtainFile              [0] IMPLICIT ObtainFile-Error,
        start                   [1] IMPLICIT Start-Error,
        stop                    [2] IMPLICIT Stop-Error,
        resume                  [3] IMPLICIT Resume-Error,
        reset                   [4] IMPLICIT Reset-Error,
        deleteVariableAccess    [5] IMPLICIT DeleteVariableAccess-Error,
        deleteNamedVariableList
                                [6] IMPLICIT DeleteNamedVariableList-Error,
        deleteNamedType         [7] IMPLICIT DeleteNamedType-Error,
        defineEventEnrollment-Error
                                [8] DefineEventEnrollment-Error,
        fileRename              [9] IMPLICIT FileRename-Error,
                                -- Reserved for use by annex C Rename service
        additionalService      [10] AdditionalService-Error
        } OPTIONAL
```

}

The ServiceError parameter shall identify the class and code for the error, may provide locally defined error code and error message information, and shall provide service specific information for services which require additional information to be conveyed in the case of errors. The errorClass, distinguished values of the errorClass, additionalCode, and additionalDescription parameters are derived in accordance with the conventions in this part of ISO/IEC 9506, 5.5, and the definitions in ISO/IEC 9506-1, clause 17.

The selection of an errorClass CHOICE shall be based on the Error Class subparameter of the ErrorType parameter specified in the response service primitive. The selection of a value for the Error Class selected shall be based on the Error Code subparameter of the ErrorType parameter specified in the response service primitive. The additionalCode and additionalDescription parameters shall be derived from the subparameters in the ErrorType parameter that have the same name.

The serviceSpecificInformation choice shall not be present if the modifierPosition parameter is present in the Confirmed-ErrorPDU. If the modifierPosition parameter is not present in the Confirmed-ErrorPDU, then the serviceSpecificInformation choice shall be derived from other parameters specified as sub-parameters of the Result(-) parameter for particular services (if such service specific sub-parameters exist).

NOTE – Service specific information is not specified when a modifier causes the Confirmed-ErrorPDU to be returned. When the Confirmed-ErrorPDU is returned as a result of an error that occurs during the processing of a confirmed service request, the service specific information is returned for those services that provide such information as required by the respective service procedures.

The AdditionalService-Error shall be reserved for definition of service specific error information for additional services by MMS Companion Standards.

NOTE – All such additional services shall meet the following requirements as described in annex A:

a) no additional service shall violate any of the state transition rules provided by ISO/IEC 9506-1 and ISO/IEC 9506-2. This limits Companion Standards to definition of substates within existing MMS states;

b) no additional service shall violate any of the object models provided by ISO/IEC 9506-1 and ISO/IEC 9506-2;

c) no additional service shall be used to circumvent the spirit and intent of ISO/IEC 9506-1 and ISO/IEC 9506-2.

7.6 Common MMS Types

This clause defines a number of types which are referenced in various other clauses in this part of ISO/IEC 9506.

7.6.1 TimeOfDay

```
TimeOfDay ::= OCTET STRING (SIZE(4|6))
```

The TimeOfDay type shall be an OCTET STRING. A value of the TimeOfDay type may contain either four (4) or six (6) octets. The first form specifies the time as the number of milliseconds since midnight on the current date (the date is not contained in the value), while the second form contains the time and a date, expressed as the relative day since January 1, 1984. The first four octets shall contain a value indicating the number of milliseconds since midnight for the current date in both forms. The value of the time field shall be derived by numbering the bits of these octets, starting with the least significant bit of the last octet as bit zero and ending the numbering with the most significant bit of the first octet as bit thirty-one. Each bit shall be assigned a numerical value of 2**N, where N is the position of the bit in this numbering sequence. The value of the time shall be obtained by summing the numerical values assigned to each bit for those bits which are set to one. Bits twenty-eight through thirty-one shall always be set to zero.

The octets of the TimeOfDay type shall be specified as follows. If the value contains the Date (six octet content), it shall be represented (using ASN.1 bstring notation) as:

```
'0000ttttttttttttttttttttttttttttddddddddddddddd'B
```

If the value does not contain the Date (four octet content length), the last two octets ("d...d") are omitted. In the bstring representation given above, "t...t" is the relative millisecond of the reported day, with midnight represented by 0, and "d...d" is the relative day, with January 1, 1984 equal to 0. All values are binary values.

The most significant bit of a sub-field value occurs earlier in the bstring with bit significance decreasing for later bits of the bstring.

A system which implements the TimeOfDay type shall specify the granularity of the "t...t" sub-field in the Protocol Implementation Conformance Statement (see clause 18).

7.6.2 Identifiers and Integer Types

The types "Identifier", "Integer8", "Integer16", "Integer32", "Unsigned8", "Unsigned16", and "Unsigned32" are used throughout this part of ISO/IEC 9506. These types are defined as follows.

```
Identifier ::= VisibleString FROM
               ("A"|"a"|"B"|"b"|"C"|"c"|"D"|"d"|"E"|"e"|"F"|"f"|
                "G"|"g"|"H"|"h"|"I"|"i"|"J"|"j"|"K"|"k"|"L"|"l"|
                "M"|"m"|"N"|"n"|"O"|"o"|"P"|"p"|"Q"|"q"|"R"|"r"|
                "S"|"s"|"T"|"t"|"U"|"u"|"V"|"v"|"W"|"w"|"X"|"x"|
                "Y"|"y"|"Z"|"z"|"$"|"_"|"0"|"1"|"2"|"3"|"4"|"5"|
                "6"|"7"|"8"|"9") (SIZE(1..32))
                -- An Identifier shall not begin with a digit.

Integer8   ::= INTEGER(-128..127)          -- range -128 <= i <= 127

Integer16  ::= INTEGER(-32768..32767)      -- range -32,768 <= i <= 32,767

Integer32  ::= INTEGER(-2147483648..2147483647)
                                           -- range -2**31 <= i <= 2**31 - 1

Unsigned8  ::= INTEGER(0..127)             -- range 0 <= i <= 127

Unsigned16 ::= INTEGER(0..32767)           -- range 0 <= i <= 32767

Unsigned32 ::= INTEGER(0..2147483647)      -- range 0 <= i <= 2**31 - 1
```

MMS defines various types of names (variable names, type names, etc) in terms of the Identifier production. An Identifier shall be limited in length to 32 octets which shall be chosen from the set characters defined by the VisibleString type as specified in ASN.1 (ISO 8824). Identifiers shall be case sensitive. An Identifier shall not begin with a digit.

Integer8, Integer16, Integer32, Unsigned8, Unsigned16 and Unsigned32 are used throughout this part of ISO/IEC 9506 to represent integers of restricted range, where the minimum and maximum representable values are as specified in the comments following their type declaration.

7.6.3 ObjectName

```
ObjectName ::= CHOICE {
     vmd-specific    [0] IMPLICIT Identifier,
     domain-specific [1] IMPLICIT SEQUENCE {
          domainID Identifier,
          itemID   Identifier
          },
     aa-specific     [2] IMPLICIT Identifier
     }
```

The ObjectName parameter shall be derived according to the rules provided in this part of ISO/IEC 9506, 5.5, using the definition of the ObjectName service parameter defined in ISO/IEC 9506-1, clause 7.

7.6.4 ApplicationReference

```
ApplicationReference ::= SEQUENCE {
     ap-title           [0] ISO-8650-ACSE-1.AP-title OPTIONAL,
     ap-invocation-id   [1] ISO-8650-ACSE-1.AP-invocation-id OPTIONAL,
     ae-qualifier       [2] ISO-8650-ACSE-1.AE-qualifier OPTIONAL,
     ae-invocation-id   [3] ISO-8650-ACSE-1.AE-invocation-id OPTIONAL
    }
```

The ApplicationReference parameter shall be derived according to the rules provided in this part of ISO/IEC 9506, 5.5, using the definition of the Application Reference service parameter defined in ISO/IEC 9506-1, clause 6. This ASN.1 definition makes use of the AP-title, AP-invocation-id, AE-qualifier, and AE-invocation-id types defined in the ISO-8650-ACSE-1 module definition provided in ISO 8650.

The values used in the ApplicationReference in any instance of its use shall be chosen such that the ApplicationReference is sufficient to identify uniquely and unambiguously the Application Process, Application Process Invocation, Application Entity, or Application Entity Invocation as required by the referencing MMS service.

NOTE 1 – Additional information on application layer naming and addressing can be found in ISO 7498-3, ISO/IEC 9545, ISO 8649, and ISO 8650.

NOTE 2 – The exact definition and values used for AP-title, AP-invocation-id, AE-qualifier, and AE-invocation-id should be chosen taking into account the ongoing work in the areas of the Association Control Service Element, naming, directories, application layer structure, and the establishment of the Registration Authority for AE-titles.

7.6.5 FileName

```
FileName ::= SEQUENCE OF GraphicString
```

The Filename type shall consist of a sequence of graphic strings. Determination of the semantics of elements in the sequence of graphic strings of a file name shall be a local matter. Any restrictions imposed by a system conforming to ISO/IEC 9506-1 and ISO/IEC 9506-2 on lengths and legal characters in a FileName shall be specified in the Protocol Implementation Conformance Statement (see clause 18). As a minimum, each implementation making use of the filename type defined in this clause shall support filenames containing a single element consisting of one to eight upper case letters or numbers that start with a letter.

NOTE – ISO/IEC 9506-1 and ISO/IEC 9506-2 do not define any interpretation for the components of a filename; such components provide a transparent naming mechanism to the initiator and the responder. The relation between the components defined in the virtual filestore and any division into components in the real system environment is a local implementation choice. An implementation may map a local component structure on to the components of the filename, or it may choose to map its existing filename syntax into a filename with only one component name. An implementation may reflect the MMS filename components in selecting an access path to the real file, but this choice is not in itself visible for interconnection purposes through MMS.

7.6.6 Priority

```
Priority ::= Unsigned8

normalPriority Priority ::= 64
```

The Priority type shall consist of an Unsigned8. The Priority type is an integer with values ranging from zero (0) to one hundred twenty seven (127), inclusive. Zero represents the highest priority. One hundred twenty seven represents the lowest priority. By convention, sixty four (64) represents the "normal" priority.

7.6.7 Severity

```
Severity ::= Unsigned8

normalSeverity Severity ::= 64
```

The Severity type shall consist of an Unsigned8. The Severity type is an integer with values ranging from zero (0) to one hundred twenty seven (127), inclusive. Zero represents the greatest severity. One hundred twenty seven represents the least severity. By convention, sixty four (64) represents the "normal" severity.

8 Environment and General Management Protocol

8.1 Introduction

This clause describes the PDUs of the services that make up the Environment and General Management services. This clause specifies the protocol required for realization of the following services:

a) Initiate;

b) Conclude;

c) Abort;

d) Cancel;

e) Reject.

8.2 Initiate

The abstract syntax of the Initiate service request, response, and error are specified by the Initiate-RequestPDU, Initiate-ResponsePDU, and Initiate-ErrorPDU types respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause. Any additional valid tagged ASN.1 values received as sequence elements in the Initiate-RequestPDU, Initiate-ResponsePDU, or Initiate-ErrorPDU shall be ignored for upward compatibility purposes.

NOTE – The type definitions for InitRequestDetail and InitResponseDetail are described in clause 19.

```
Initiate-RequestPDU ::= SEQUENCE {
    localDetailCalling                  [0]  IMPLICIT Integer32 OPTIONAL,
    proposedMaxServOutstandingCalling   [1]  IMPLICIT Integer16,
    proposedMaxServOutstandingCalled    [2]  IMPLICIT Integer16,
    proposedDataStructureNestingLevel   [3]  IMPLICIT Integer8 OPTIONAL,
    initRequestDetail                   [4]  IMPLICIT InitRequestDetail
    }

Initiate-ResponsePDU ::= SEQUENCE {
    localDetailCalled                   [0]  IMPLICIT Integer32 OPTIONAL,
    negotiatedMaxServOutstandingCalling [1]  IMPLICIT Integer16,
    negotiatedMaxServOutstandingCalled  [2]  IMPLICIT Integer16,
    negotiatedDataStructureNestingLevel [3]  IMPLICIT Integer8 OPTIONAL,
    initResponseDetail                  [4]  IMPLICIT InitResponseDetail
    }

Initiate-ErrorPDU  ::= ServiceError
```

```
ServiceSupportOptions ::= BIT STRING {
    status                              (0),
    getNameList                         (1),
    identify                            (2),
    rename                              (3),
    read                                (4),
    write                               (5),
    getVariableAccessAttributes         (6),
    defineNamedVariable                 (7),
    defineScatteredAccess               (8),
    getScatteredAccessAttributes        (9),
    deleteVariableAccess                (10),
    defineNamedVariableList             (11),
    getNamedVariableListAttributes      (12),
    deleteNamedVariableList             (13),
    defineNamedType                     (14),
    getNamedTypeAttributes              (15),
    deleteNamedType                     (16),
    input                               (17),
    output                              (18),
    takeControl                         (19),
    relinquishControl                   (20),
    defineSemaphore                     (21),
    deleteSemaphore                     (22),
    reportSemaphoreStatus               (23),
    reportPoolSemaphoreStatus           (24),
    reportSemaphoreEntryStatus          (25),
    initiateDownloadSequence            (26),
    downloadSegment                     (27),
    terminateDownloadSequence           (28),
    initiateUploadSequence              (29),
    uploadSegment                       (30),
    terminateUploadSequence             (31),
    requestDomainDownload               (32),
    requestDomainUpload                 (33),
    loadDomainContent                   (34),
    storeDomainContent                  (35),
    deleteDomain                        (36),
    getDomainAttributes                 (37),
    createProgramInvocation             (38),
    deleteProgramInvocation             (39),
    start                               (40),
    stop                                (41),
    resume                              (42),
    reset                               (43),
    kill                                (44),
    getProgramInvocationAttributes      (45),
    obtainFile                          (46),
    defineEventCondition                (47),
    deleteEventCondition                (48),
    getEventConditionAttributes         (49),
    reportEventConditionStatus          (50),
    alterEventConditionMonitoring       (51),
    triggerEvent                        (52),
    defineEventAction                   (53),
    deleteEventAction                   (54),
    getEventActionAttributes            (55),
    reportEventActionStatus             (56),
    defineEventEnrollment               (57),
    deleteEventEnrollment               (58),
    alterEventEnrollment                (59),
    reportEventEnrollmentStatus         (60),
    getEventEnrollmentAttributes        (61),
```

```
        acknowledgeEventNotification    (62),
        getAlarmSummary                 (63),
        getAlarmEnrollmentSummary       (64),
        readJournal                     (65),
        writeJournal                    (66),
        initializeJournal               (67),
        reportJournalStatus             (68),
        createJournal                   (69),
        deleteJournal                   (70),
        getCapabilityList               (71),
        fileOpen                        (72),
               -- this bit is reserved for use of a service defined in annex C
        fileRead                        (73),
               -- this bit is reserved for use of a service defined in annex C
        fileClose                       (74),
               -- this bit is reserved for use of a service defined in annex C
        fileRename                      (75),
               -- this bit is reserved for use of a service defined in annex C
        fileDelete                      (76),
               -- this bit is reserved for use of a service defined in annex C
        fileDirectory                   (77),
               -- this bit is reserved for use of a service defined in annex C
        unsolicitedStatus               (78),
        informationReport               (79),
        eventNotification               (80),
        attachToEventCondition          (81),
        attachToSemaphore               (82),
        conclude                        (83),
        cancel                          (84)
        }

    ParameterSupportOptions ::= BIT STRING {
        str1 (0),
        str2 (1),
        vnam (2),
        valt (3),
        vadr (4),
        vsca (5),
        tpy  (6),
        vlis (7),
        real (8),
        akec (9),
        cei (10)
        }
```

8.2.1 Initiate-RequestPDU

The abstract syntax of the Initiate service request shall be the Initiate-RequestPDU.

8.2.2 Initiate-ResponsePDU

The abstract syntax of the Initiate service response shall be the Initiate-ResponsePDU.

8.2.3 Initiate-ErrorPDU

The abstract syntax of the Initiate service error shall be the Initiate-ErrorPDU.

8.3 Conclude

The abstract syntax of the Conclude service request, response, and error are specified by the Conclude-RequestPDU, Conclude-ResponsePDU, and Conclude-ErrorPDU types respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Conclude-RequestPDU ::= NULL

Conclude-ResponsePDU ::= NULL

Conclude-ErrorPDU  ::= ServiceError
```

8.3.1 Conclude-RequestPDU

The abstract syntax of the Conclude service request shall be the Conclude-RequestPDU.

8.3.2 Conclude-ResponsePDU

The abstract syntax of the Conclude service response shall be the Conclude-ResponsePDU.

8.3.3 Conclude-ErrorPDU

The abstract syntax of the Conclude service error shall be the Conclude-ErrorPDU.

8.4 Abort

The abort service is directly mapped to the ACSE A-ABORT service (see clause 17).

8.5 Cancel

The abstract syntax of the Cancel service request, response, and error are specified by the Cancel-RequestPDU, Cancel-ResponsePDU, and Cancel-ErrorPDU types respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Cancel-RequestPDU ::= Unsigned32 -- originalInvokeID

Cancel-ResponsePDU ::=  Unsigned32 -- originalInvokeID

Cancel-ErrorPDU  ::=  SEQUENCE  {
    originalInvokeID [0] IMPLICIT Unsigned32,
    serviceError     [1] IMPLICIT ServiceError
    }
```

8.5.1 Cancel-RequestPDU

The abstract syntax of the Cancel service request shall be the Cancel-RequestPDU.

8.5.2 Cancel-ResponsePDU

The abstract syntax of the Cancel service response shall be the Cancel-ResponsePDU.

8.5.3 Cancel-ErrorPDU

The abstract syntax of the Cancel service error shall be the Cancel-ErrorPDU.

8.6 Reject

The abstract syntax of the Reject service is specified by the RejectPDU. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
RejectPDU ::= SEQUENCE {
    originalInvokeID            [0] IMPLICIT Unsigned32 OPTIONAL,
    rejectReason  CHOICE {
       confirmed-requestPDU [1] IMPLICIT INTEGER { -- CONFIRMED-REQUESTPDU
          other (0),                          -- OTHER
          unrecognized-service (1),           -- UNRECOGNIZED-SERVICE
          unrecognized-modifier (2),          -- UNRECOGNIZED-MODIFIER
          invalid-invokeID (3),               -- INVALID-INVOKEID
          invalid-argument (4),               -- INVALID-ARGUMENT
          invalid-modifier (5),               -- INVALID-MODIFIER
          max-serv-outstanding-exceeded (6),
                                  -- MAX-SERV-OUTSTANDING-EXCEEDED
            -- Value 7 reserved for further definition
          max-recursion-exceeded (8),         -- MAX-RECURSION-EXCEEDED
          value-out-of-range (9)              -- VALUE-OUT-OF-RANGE
          },
       confirmed-responsePDU [2] IMPLICIT INTEGER { --CONFIRMED-RESPONSEPDU
          other (0),                          -- OTHER
          unrecognized-service (1),           -- UNRECOGNIZED-SERVICE
          invalid-invokeID (2),               -- INVALID-INVOKEID
          invalid-result   (3),               -- INVALID-RESULT
            -- Value 4 reserved for further definition
          max-recursion-exceeded (5),         -- MAX-RECURSION-EXCEEDED
          value-out-of-range (6)              -- VALUE-OUT-OF-RANGE
          },
       confirmed-errorPDU [3] IMPLICIT INTEGER { -- CONFIRMED-ERRORPDU
          other (0),                          -- OTHER
          unrecognized-service (1),           -- UNRECOGNIZED-SERVICE
          invalid-invokeID (2),               -- INVALID-INVOKEID
          invalid-serviceError (3),           -- INVALID-SERVICEERROR
          value-out-of-range (4)              -- VALUE-OUT-OF-RANGE
          },
       unconfirmedPDU [4] IMPLICIT INTEGER { -- UNCONFIRMEDPDU
          other (0),                          -- OTHER
          unrecognized-service (1),           -- UNRECOGNIZED-SERVICE
          invalid-argument (2),               -- INVALID-ARGUMENT
          max-recursion-exceeded (3),         -- MAX-RECURSION-EXCEEDED
          value-out-of-range (4)              -- VALUE-OUT-OF-RANGE
          },
       pdu-error [5] IMPLICIT INTEGER {       -- PDU-ERROR
          unknown-pdu-type (0),               -- UNKNOWN-PDU-TYPE
          invalid-pdu (1),                    -- INVALID-PDU
          illegal-acse-mapping (2)            -- ILLEGAL-ACSE-MAPPING
          },
       cancel-requestPDU [6] IMPLICIT INTEGER { -- CANCEL-REQUESTPDU
          other (0),                          -- OTHER
          invalid-invokeID (1)                -- INVALID-INVOKEID
          },
       cancel-responsePDU [7] IMPLICIT INTEGER { -- CANCEL-RESPONSEPDU
          other (0),                          -- OTHER
          invalid-invokeID (1)                -- INVALID-INVOKEID
          },
       cancel-errorPDU [8] IMPLICIT INTEGER { -- CANCEL-ERRORPDU
```

```
            other (0),                          -- OTHER
            invalid-invokeID (1),               -- INVALID-INVOKEID
            invalid-serviceError (2),           -- INVALID-SERVICEERROR
            value-out-of-range (3)              -- VALUE-OUT-OF-RANGE
            },
        conclude-requestPDU [9] IMPLICIT INTEGER { -- CONCLUDE-REQUESTPDU
            other (0),                          -- OTHER
            invalid-argument (1)                -- INVALID-ARGUMENT
            },
        conclude-responsePDU [10] IMPLICIT INTEGER { -- CONCLUDE-RESPONSEPDU
            other (0),                          -- OTHER
            invalid-result (1)                  -- INVALID-RESULT
            },
        conclude-errorPDU [11] IMPLICIT INTEGER { -- CONCLUDE-ERRORPDU
            other (0),                          -- OTHER
            invalid-serviceError (1),           -- INVALID-SERVICEERROR
            value-out-of-range (2)              -- VALUE-OUT-OF-RANGE
            }
        }
    }
```

The abstract syntax for the Reject Service shall be the Reject PDU. The reject reason parameter is derived from the Reject PDU Type and the Reject Code parameters in the service specification. The choice selected shall match the Reject PDU Type in the service parameter as indicated in the comments. The value for choice selected shall be chosen to match the Reject code value in the service parameter as indicated in the comments.

9 VMD Support Protocol

9.1 Introduction

This clause describes the PDUs of the VMD Support Services. This clause specifies the protocol required for realization of the following services:

a) Status;

b) UnsolicitedStatus;

c) GetNameList;

d) Identify;

e) Rename;

f) GetCapabilityList.

9.2 Status

The abstract syntax of the status choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the Status-Request and Status-Response types, respectively. These types are specified below and described in the paragraphs which follow. The derivation of all parameters for which explicit derivations are not provided in this clause is described in 5.5.

```
Status-Request ::= BOOLEAN -- Extended Derivation
```

```
Status-Response ::= SEQUENCE {
   vmdLogicalStatus        [0] IMPLICIT INTEGER {
       state-changes-allowed      (0),
       no-state-changes-allowed   (1),
       limited-services-permitted (2),
       support-services-allowed   (3)
       },
   vmdPhysicalStatus       [1] IMPLICIT INTEGER {
       operational           (0),
       partially-operational (1),
       inoperable            (2),
       needs-commissioning   (3)
       },
   localDetail              [2] IMPLICIT BIT STRING (SIZE(0..128)) OPTIONAL
                              -- not to exceed 128 bits in length
   }
```

9.2.1 Status-Request

The abstract syntax of the status choice of the ConfirmedServiceRequest shall be the Status-Request.

9.2.2 Status-Response

The abstract syntax of the status choice of the ConfirmedServiceResponse shall be the Status-Response.

9.3 UnsolicitedStatus

The abstract syntax of the unsolicitedStatus choice of the UnconfirmedService is specified by the UnsolicitedStatus type. This type is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
UnsolicitedStatus ::= Status-Response
```

9.3.1 UnsolicitedStatus

The abstract syntax of the unsolicitedStatus choice of the UnconfirmedService shall be the UnsolicitedStatus.

9.4 GetNameList

The abstract syntax of the getNameList choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the GetNameList-Request and GetNameList-Response types respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetNameList-Request ::= SEQUENCE {
   extendedObjectClass [0] CHOICE {
       objectClass     [0] IMPLICIT INTEGER {
          namedVariable        (0),
          scatteredAccess      (1),
          namedVariableList    (2),
          namedType            (3),
          semaphore            (4),
          eventCondition       (5),
          eventAction          (6),
          eventEnrollment      (7),
          journal              (8),
          domain               (9),
          programInvocation    (10),
          operatorStation      (11)
          },
       csObjectClass [1] CsAdditionalObjectClasses
     -- Shall not be chosen in the abstract syntax
     -- defined in clause 19 in this part of this Standard
       },
   objectScope      [1] CHOICE {
       vmdSpecific     [0] IMPLICIT NULL,
       domainSpecific  [1] IMPLICIT Identifier,
       aaSpecific      [2] IMPLICIT NULL
       },
   continueAfter [2] IMPLICIT Identifier OPTIONAL
   }

GetNameList-Response ::= SEQUENCE {
   listOfIdentifier [0] IMPLICIT SEQUENCE OF Identifier,
   moreFollows      [1] IMPLICIT BOOLEAN DEFAULT TRUE
   }
```

9.4.1 GetNameList-Request

The abstract syntax of the getNameList choice of the ConfirmedServiceRequest shall be the GetNameList-Request.

9.4.1.1 Extended Object Class

The objectClass choice within the GetNameList-Request shall be chosen if the value of the Extended Object Class parameter of the service request primitive is OBJECT-CLASS.

The csObjectClass choice within the GetNameList-Request shall be chosen if the value of the Extended Object Class parameter of the service request primitive is CS-OBJECT-CLASS.

9.4.1.2 Object Scope

The vmdSpecific choice within the GetNameList-Request shall be chosen if the value of the Object Scope parameter of the service request primitive is VMD-Specific.

The domainSpecific choice within the GetNameList-Request shall be chosen if the value of the Object Scope parameter of the service request primitive is Domain-Specific. The value of the domainSpecific type shall be derived from the value of the Domain Name parameter of the service request primitive.

The aaSpecific choice within the GetNameList-Request shall be chosen if the value of the Object Scope parameter of the service request primitive is AA-Specific.

9.4.2 GetNameList-Response

The abstract syntax of the getNameList choice of the ConfirmedServiceResponse shall be the GetNameList-Response.

9.4.3 CsAdditionalObjectClasses

The CsAdditionalObjectClasses type conveys the CS Object Class parameter of the GetNameList-Request and the Rename-Request. This type shall be specified by a Companion Standard and shall not be conveyed in PDU's identified by the presentation context derived by the abstract syntax defined in clause 19.

9.5 Identify

The abstract syntax of the identify choice of the ConfirmedServiceRequest and ConfirmedServiceResponse, is specified by the Identify-Request and Identify-Response types respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Identify-Request ::= NULL

Identify-Response ::= SEQUENCE {
   vendorName              [0] IMPLICIT VisibleString,
   modelName               [1] IMPLICIT VisibleString,
   revision                [2] IMPLICIT VisibleString,
   listOfAbstractSyntaxes  [3] IMPLICIT SEQUENCE OF
                               OBJECT IDENTIFIER OPTIONAL
   }
```

9.5.1 Identify-Request

The abstract syntax of the identify choice of the ConfirmedServiceRequest shall be the Identify-Request.

9.5.2 Identify-Response

The abstract syntax of the identify choice of the ConfirmedServiceResponse shall be the Identify-Response.

9.6 Rename

The abstract syntax of the rename choice of the ConfirmedServiceRequest and ConfirmedServiceResponse, is specified by the Rename-Request and Rename-Response types, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Rename-Request ::= SEQUENCE {
   extendedObjectClass [0] CHOICE {
      objectClass     [0] IMPLICIT INTEGER {
         namedVariable        (0),
         scatteredAccess      (1),
         namedVariableList    (2),
         namedType            (3),
         semaphore            (4),
         eventCondition       (5),
         eventAction          (6),
         eventEnrollment      (7),
         journal              (8),
         domain               (9),
         programInvocation    (10),
         operatorStation      (11)
         },
      csObjectClass [1] CsAdditionalObjectClasses
      -- Shall not be chosen in the abstract syntax
      -- defined in clause 19 of this Standard
      },
   currentName         [1] ObjectName,
   newIdentifier       [2] IMPLICIT Identifier
   }

Rename-Response ::= NULL
```

9.6.1 Rename-Request

The abstract syntax of the rename choice of the ConfirmedServiceRequest shall be the Rename-Request.

9.6.1.1 Extended Object Class

The objectClass choice within the Rename-Request shall be chosen if the value of the Extended Object Class parameter of the service request primitive is OBJECT-CLASS.

The csObjectClass choice within the Rename-Request shall be chosen if the value of the Extended Object Class parameter of the service request primitive is CS-OBJECT-CLASS.

9.6.2 Rename-Response

The abstract syntax of the rename choice of the ConfirmedServiceResponse shall be the Rename-Response.

9.7 GetCapabilityList

The abstract syntax of the getCapabilityList choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the GetCapabilityList-Request and GetCapabilityList-Response types, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetCapabilityList-Request ::= SEQUENCE {
   continueAfter VisibleString OPTIONAL
   }

GetCapabilityList-Response ::= SEQUENCE {
   listOfCapabilities     [0] IMPLICIT SEQUENCE OF VisibleString,
   moreFollows            [1] IMPLICIT BOOLEAN DEFAULT TRUE
   }
```

9.7.1 GetCapabilityList-Request

The abstract syntax of the getCapabilityList choice of the ConfirmedServiceRequest shall be the GetCapabilityList-Request.

9.7.2 GetCapabilityList-Response

The abstract syntax of the getCapabilityList choice of the ConfirmedServiceResponse shall be the GetCapabilityList-Response.

10 Domain Management Protocol

10.1 Introduction

This clause describes the service-specific protocol elements of the services which are defined by the Domain Management clause of the MMS service definition. This includes:

InitiateDownloadSequence
DownloadSegment
TerminateDownloadSequence
InitiateUploadSequence
UploadSegment
TerminateUploadSequence
RequestDomainDownload
RequestDomainUpload
LoadDomainContent
StoreDomainContent
DeleteDomain
GetDomainAttributes

10.2 InitiateDownloadSequence

The abstract syntax of the initiateDownloadSequence choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the InitiateDownloadSequence-Request and the InitiateDownloadSequence-Response, respectively. These types are specified below and described in the paragraphs which follow. The derivation of all parameters for which explicit derivations are not provided in this clause is described in 5.5.

```
InitiateDownloadSequence-Request ::= SEQUENCE {
    domainName          [0] IMPLICIT Identifier,
    listOfCapabilities  [1] IMPLICIT SEQUENCE OF VisibleString,
    sharable            [2] IMPLICIT BOOLEAN
    }

InitiateDownloadSequence-Response ::= NULL
```

10.2.1 InitiateDownloadSequence-Request

The abstract syntax of the initiateDownloadSequence choice of the ConfirmedServiceRequest shall be the InitiateDownloadSequence-Request.

10.2.2 InitiateDownloadSequence-Response

The abstract syntax of the initiateDownloadSequence choice of the ConfirmedServiceResponse shall be the InitiateDownloadSequence-Response.

10.3 DownloadSegment

The abstract syntax of the downloadSegment choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the DownloadSegment-Request and the DownloadSegment-Response respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DownloadSegment-Request ::= Identifier -- Domain Name

DownloadSegment-Response ::= SEQUENCE {
    loadData                CHOICE {
       non-coded                [0] IMPLICIT OCTET STRING,
       coded                    EXTERNAL
       },
    moreFollows             [1] IMPLICIT BOOLEAN DEFAULT TRUE
    }
```

10.3.1 DownloadSegment-Request

The abstract syntax of the downloadSegment choice of the ConfirmedServiceRequest shall be the DownloadSegment-Request.

10.3.2 DownloadSegment-Response

The abstract syntax of the downloadSegment choice of the ConfirmedServiceResponse shall be the DownloadSegment-Response.

10.3.2.1 Load Data

The abstract syntax of the Load Data parameter of the DownloadSegment service response shall be a choice between an OCTET STRING, indicating that the value of this parameter is not further described and its interpretation is a local matter, and an EXTERNAL, indicating that the abstract syntax referenced by the EXTERNAL contains coding rules for interpreting the value of this parameter.

10.4 TerminateDownloadSequence

The abstract syntax of the terminateDownloadSequence choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the TerminateDownloadSequence-Request and the TerminateDownloadSequence-Response respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
TerminateDownloadSequence-Request ::= SEQUENCE {
    domainName              [0] IMPLICIT Identifier,
    discard                 [1] IMPLICIT ServiceError OPTIONAL
    }

TerminateDownloadSequence-Response ::= NULL
```

10.4.1 TerminateDownloadSequence-Request

The abstract syntax of the terminateDownloadSequence choice of the ConfirmedServiceRequest shall be the Terminate-DownloadSequence-Request.

10.4.1.1 Discard

The abstract syntax of the Discard parameter is provided by ServiceError. If the Discard parameter is present in the TerminateDownloadSequence service request, the ServiceError type shall be included providing a reason for the discard. If the Discard parameter is not present in the TerminateDownloadSequence service request, the ServiceError field shall not be included.

10.4.2 TerminateDownloadSequence-Response

The abstract syntax of the terminateDownloadSequence choice of the ConfirmedServiceResponse shall be the Terminate-DownloadSequence-Response.

10.5 InitiateUploadSequence

The abstract syntax of the initiateUploadSequence choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the InitiateUploadSequence-Request and the InitiateUploadSequence-Response respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
InitiateUploadSequence-Request ::= Identifier -- Domain Name

InitiateUploadSequence-Response ::= SEQUENCE {
    ulsmID                  [0] IMPLICIT Integer32,
    listOfCapabilities      [1] IMPLICIT SEQUENCE OF VisibleString
    }
```

10.5.1 InitiateUploadSequence-Request

The abstract syntax of the initiateUploadSequence choice of the ConfirmedServiceRequest shall be the InitiateUploadSe-quence-Request.

10.5.2 InitiateUploadSequence-Response

The abstract syntax of the initiateUploadSequence choice of the ConfirmedServiceResponse shall be the InitiateUpload-Sequence-Response.

10.6 UploadSegment

The abstract syntax of the uploadSegment choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is spec-ified by the UploadSegment-Request and the UploadSegment-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
UploadSegment-Request ::= Integer32 -- ULSM ID

UploadSegment-Response ::= SEQUENCE {
    loadData                CHOICE {
        non-coded               [0] IMPLICIT OCTET STRING,
        coded                   EXTERNAL
        },
    moreFollows             [1] IMPLICIT BOOLEAN DEFAULT TRUE
    }
```

10.6.1 UploadSegment-Request

The abstract syntax of the uploadSegment choice of the ConfirmedServiceRequest shall be the UploadSegment-Request.

10.6.2 UploadSegment-Response

The abstract syntax of the uploadSegment choice of the ConfirmedServiceResponse shall be the UploadSegment-Response.

10.6.2.1 Load Data

The abstract syntax of the Load Data parameter of the UploadSegment service response shall be a choice between an OCTET STRING, indicating that the value of this parameter is not further described and its interpretation is a local matter, and an EXTERNAL, indicating that the abstract syntax referenced by the EXTERNAL contains coding rules for interpreting the value of this parameter.

10.7 TerminateUploadSequence

The abstract syntax of the terminateUploadSequence choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the TerminateUploadSequence-Request and the TerminateUploadSequence-Response respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
TerminateUploadSequence-Request ::= Integer32 -- ULSM ID

TerminateUploadSequence-Response ::= NULL
```

10.7.1 TerminateUploadSequence-Request

The abstract syntax of the terminateUploadSequence choice of the ConfirmedServiceRequest shall be the TerminateUploadSequence-Request.

10.7.2 TerminateUploadSequence-Response

The abstract syntax of the terminateUploadSequence choice of the ConfirmedServiceResponse shall be the TerminateUploadSequence-Response.

10.8 RequestDomainDownload

The abstract syntax of the requestDomainDownload choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the RequestDomainDownload-Request and the RequestDomainDownload-Response respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
RequestDomainDownload-Request ::= SEQUENCE {
    domainName              [0] IMPLICIT Identifier,
    listOfCapabilities      [1] IMPLICIT SEQUENCE OF VisibleString,
    sharable                [2] IMPLICIT BOOLEAN,
    fileName                [4] IMPLICIT FileName
    }

RequestDomainDownload-Response ::= NULL
```

10.8.1 RequestDomainDownload-Request

The abstract syntax of the requestDomainDownload choice of the ConfirmedServiceRequest shall be the RequestDomainDownload-Request.

10.8.2 RequestDomainDownload-Response

The abstract syntax of the requestDomainDownload choice of the ConfirmedServiceResponse shall be the RequestDomainDownload-Response.

10.9 RequestDomainUpload

The abstract syntax of the requestDomainUpload choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the RequestDomainUpload-Request and the RequestDomainUpload-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
RequestDomainUpload-Request ::= SEQUENCE {
    domainName              [0] IMPLICIT Identifier,
    fileName                [1] IMPLICIT FileName
    }

RequestDomainUpload-Response ::= NULL
```

10.9.1 RequestDomainUpload-Request

The abstract syntax of the requestDomainUpload choice of the ConfirmedServiceRequest shall be the RequestDomainUpload-Request.

10.9.2 RequestDomainUpload-Response

The abstract syntax of the requestDomainUpload choice of the ConfirmedServiceResponse shall be the RequestDomainUpload-Response.

10.10 LoadDomainContent

The abstract syntax of the loadDomainContent choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the LoadDomainContent-Request and the LoadDomainContent-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
LoadDomainContent-Request ::= SEQUENCE {
    domainName              [0] IMPLICIT Identifier,
    listOfCapabilities      [1] IMPLICIT SEQUENCE OF VisibleString,
    sharable                [2] IMPLICIT BOOLEAN,
    fileName                [4] IMPLICIT FileName,
    thirdParty              [5] IMPLICIT ApplicationReference OPTIONAL
          -- TPY
    }

LoadDomainContent-Response ::= NULL
```

10.10.1 LoadDomainContent-Request

The abstract syntax of the loadDomainContent choice of the ConfirmedServiceRequest shall be the LoadDomainContent-Request.

10.10.1.1 ApplicationReference

The abstract syntax of the Third Party parameter of the LoadDomainContent service shall be ApplicationReference. This parameter shall not appear unless the TPY parameter conformance building block is supported. If the TPY parameter conformace building block is supported, the use of the thirdParty parameter is optional.

10.10.2 LoadDomainContent-Response

The abstract syntax of the loadDomainContent choice of the ConfirmedServiceResponse shall be the LoadDomainContent-Response.

10.11 StoreDomainContent

The abstract syntax of the storeDomainContent choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the StoreDomainContent-Request and the StoreDomainContent-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
StoreDomainContent-Request ::= SEQUENCE {
    domainName              [0] IMPLICIT Identifier,
    fileName                [1] IMPLICIT FileName,
    thirdParty              [2] IMPLICIT ApplicationReference OPTIONAL
        -- TPY
    }

StoreDomainContent-Response ::= NULL
```

10.11.1 StoreDomainContent-Request

The abstract syntax of the storeDomainContent choice of the ConfirmedServiceRequest shall be the StoreDomainContent-Request.

10.11.1.1 ApplicationReference

The abstract syntax of the Third Party parameter of the LoadDomainContent service shall be ApplicationReference. This parameter shall not appear unless the TPY parameter conformance building block is supported. If the TPY parameter conformance building block is supported, the use of the thirdParty parameter is optional.

10.11.2 StoreDomainContent-Response

The abstract syntax of the storeDomainContent choice of the ConfirmedServiceResponse shall be the StoreDomainContent-Response.

10.12 DeleteDomain

The abstract syntax of the deleteDomain choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the DeleteDomain-Request and the DeleteDomain-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteDomain-Request ::= Identifier -- Domain Name

DeleteDomain-Response ::= NULL
```

10.12.1 DeleteDomain-Request

The abstract syntax of the deleteDomain choice of the ConfirmedServiceRequest shall be the DeleteDomain-Request.

10.12.2 DeleteDomain-Response

The abstract syntax of the deleteDomain choice of the ConfirmedServiceResponse shall be the DeleteDomain-Response.

10.13 GetDomainAttributes

The abstract syntax of the getDomainAttributes choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the GetDomainAttributes-Request and the GetDomainAttributes-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetDomainAttributes-Request ::= Identifier -- Domain Name

GetDomainAttributes-Response ::= SEQUENCE {
    listOfCapabilities        [0] IMPLICIT SEQUENCE OF VisibleString,
    state                     [1] IMPLICIT DomainState,
    mmsDeletable              [2] IMPLICIT BOOLEAN,
    sharable                  [3] IMPLICIT BOOLEAN,
    listOfProgramInvocations  [4] IMPLICIT SEQUENCE OF Identifier,
                              --  Program Invocation Names
    uploadInProgress          [5] IMPLICIT Integer8
    }

DomainState ::=     INTEGER {
            non-existent          (0),    -- NON-EXISTENT
                                          -- shall not be reported
            loading               (1),    -- LOADED
            ready                 (2),    -- READY
            in-use                (3),    -- IN-USE
            complete              (4),    -- COMPLETE
            incomplete            (5),    -- INCOMPLETE
            d1                    (7),    -- D1
            d2                    (8),    -- D2
            d3                    (9),    -- D3
            d4                   (10),    -- D4
            d5                   (11),    -- D5
            d6                   (12),    -- D6
            d7                   (13),    -- D7
            d8                   (14),    -- D8
            d9                   (15)     -- D9
          }
```

10.13.1 GetDomainAttributes-Request

The abstract syntax of the getDomainAttributes choice of the ConfirmedServiceRequest shall be the GetDomainAttributes-Request.

10.13.2 GetDomainAttributes-Response

The abstract syntax of the getDomainAttributes choice of the ConfirmedServiceResponse shall be the GetDomainAttributes-Response.

11 Program Invocation Management Protocol

11.1 Introduction

This clause describes the protocol required for realization of the services which are defined by the Program Invocation Management clause of the MMS service definition. This includes

CreateProgramInvocation
DeleteProgramInvocation
Start
Stop
Resume
Reset
Kill
GetProgramInvocationAttributes

11.2 CreateProgramInvocation

The abstract syntax of the createProgramInvocation choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the CreateProgramInvocation-Request and the CreateProgramInvocation-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
CreateProgramInvocation-Request ::= SEQUENCE {
     programInvocationName      [0] IMPLICIT Identifier,
     listOfDomainNames          [1] IMPLICIT SEQUENCE OF Identifier,
     reusable                   [2] IMPLICIT BOOLEAN DEFAULT TRUE,
     monitorType                [3] IMPLICIT BOOLEAN OPTIONAL
            -- TRUE indicates PERMANENT monitoring,
            -- FALSE indicates CURRENT monitoring
     }

CreateProgramInvocation-Response ::= NULL
```

11.2.1 CreateProgramInvocation-Request

The abstract syntax of the createProgramInvocation choice of the ConfirmedServiceRequest shall be the CreateProgram-Invocation-Request.

11.2.1.1 Monitor

The abstract syntax of the Monitor parameter of the CreateProgramInvocation service shall be inferred from the presence or absence of the BOOLEAN OPTIONAL. If the BOOLEAN element is present, it shall indicate that the Monitor parameter is true. The value of the BOOLEAN shall indicate the value of the MonitorType parameter of the service request and shall indicate whether the derived Event Enrollment is PERMANENT or CURRENT. If the BOOLEAN element is not present, the Monitor parameter is false and no monitoring is required.

11.2.2 CreateProgramInvocation-Response

The abstract syntax of the createProgramInvocation choice of the ConfirmedServiceResponse shall be the CreateProgram-Invocation-Response.

11.3 DeleteProgramInvocation

The abstract syntax of the deleteProgramInvocation choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the DeleteProgramInvocation-Request and the DeleteProgramInvocation-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteProgramInvocation-Request ::= Identifier
      -- Program Invocation Name

DeleteProgramInvocation-Response ::= NULL
```

11.3.1 DeleteProgramInvocation-Request

The abstract syntax of the deleteProgramInvocation choice of the ConfirmedServiceRequest shall be the DeleteProgram-Invocation-Request.

11.3.2 DeleteProgramInvocation-Response

The abstract syntax of the deleteProgramInvocation choice of the ConfirmedServiceResponse shall be the DeleteProgram-Invocation-Response.

11.4 Start

The abstract syntax of the start choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the Start-Request and the Start-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Start-Request ::= SEQUENCE {
    programInvocationName    [0] IMPLICIT Identifier,
    executionArgument        CHOICE {
       simpleString             [1] IMPLICIT VisibleString,
       encodedString             EXTERNAL
       } OPTIONAL
    }

Start-Response ::= NULL

Start-Error    ::= ProgramInvocationState

ProgramInvocationState  ::=  INTEGER {
    non-existent (0),     -- NON-EXISTENT
    unrunnable   (1),     -- UNRUNNABLE
    idle         (2),     -- IDLE
    running      (3),     -- RUNNING
    stopped      (4),     -- STOPPED
    starting     (5),     -- STARTING
    stopping     (6),     -- STOPPING
    resuming     (7),     -- RESUMING
    resetting    (8)      -- RESETTING
    }
```

11.4.1 Start-Request

The abstract syntax of the start choice of the ConfirmedServiceRequest shall be the Start-Request.

11.4.1.1 Execution Argument

The Execution Argument parameter of the Start service request shall be a choice between a VisibleString, indicating that the value of this parameter is not further described and its interpretation is a local matter, or an EXTERNAL, indicating that the abstract syntax referenced by the EXTERNAL contains coding rules for interpreting the value of this parameter.

11.4.2 Start-Response

The abstract syntax of the start choice of the ConfirmedServiceResponse shall be the Start-Response.

11.4.3 Start-Error

The abstract syntax of the start choice of the serviceSpecificInformation choice of the ConfirmedServiceError type shall be Start-Error, which shall be the Program Invocation State sub-parameter of the Result(-) parameter of the Start.response primitive and shall appear as the Program Invocation State sub-parameter of the Result(-) parameter of the Start.confirm primitive, if issued.

11.4.3.1 ProgramInvocationState

The abstract syntax of the ProgramInvocationState shall be INTEGER with the indicated list of distinguished values.

11.5 Stop

The abstract syntax of the stop choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the Stop-Request and the Stop-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Stop-Request ::= SEQUENCE {
    programInvocationName     [0] IMPLICIT Identifier
    }

Stop-Response ::= NULL

Stop-Error ::= ProgramInvocationState
```

11.5.1 Stop-Request

The abstract syntax of the stop choice of the ConfirmedServiceRequest shall be the Stop-Request.

11.5.2 Stop-Response

The abstract syntax of the stop choice of the ConfirmedServiceResponse shall be the Stop-Response.

11.5.3 Stop-Error

The abstract syntax of the stop choice of the serviceSpecificInformation choice of the ConfirmedServiceError type shall be Stop-Error, which shall be the the Program Invocation State sub-parameter of the Result(-) parameter of the Stop.response primitive and shall appear as the Program Invocation State sub-parameter of the Result(-) parameter of the Stop.confirm primitive, if issued.

11.6 Resume

The abstract syntax of the resume choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the Resume-Request and the Resume-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Resume-Request ::= SEQUENCE {
    programInvocationName    [0] IMPLICIT Identifier,
    executionArgument        CHOICE {
       simpleString              [1] IMPLICIT VisibleString,
       encodedString             EXTERNAL
       } OPTIONAL
    }

Resume-Response ::= NULL

Resume-Error    ::= ProgramInvocationState
```

11.6.1 Resume-Request

The abstract syntax of the resume choice of the ConfirmedServiceRequest shall be the Resume-Request.

11.6.1.1 Execution Argument

The Execution Argument parameter of the Resume service request shall be a choice between a VisibleString, indicating that the value of this parameter is not further described and its interpretation is a local matter, or an EXTERNAL, indicating that the abstract syntax referenced by the EXTERNAL contains coding rules for interpreting the value of this parameter.

11.6.2 Resume-Response

The abstract syntax of the resume choice of the ConfirmedServiceResponse shall be the Resume-Response.

11.6.3 Resume-Error

The abstract syntax of the resume choice of the serviceSpecificInformation choice of the ConfirmedServiceError type shall be Resume-Error, which shall be the the Program Invocation State sub-parameter of the Result(-) parameter of the Resume.response primitive and shall appear as the Program Invocation State sub-parameter of the Result(-) parameter of the Resume.confirm primitive, if issued.

11.7 Reset

The abstract syntax of the reset choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the Reset-Request and the Reset-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Reset-Request ::= SEQUENCE {
    programInvocationName  [0] IMPLICIT Identifier
    }

Reset-Response ::= NULL

Reset-Error    ::= ProgramInvocationState
```

11.7.1 Reset-Request

The abstract syntax of the reset choice of the ConfirmedServiceRequest shall be the Reset-Request.

11.7.2 Reset-Response

The abstract syntax of the reset choice of the ConfirmedServiceResponse shall be the Reset-Response.

11.7.3 Reset-Error

The abstract syntax of the reset choice of the serviceSpecificInformation choice of the ConfirmedServiceError type shall be Reset-Error, which shall be the the Program Invocation State sub-parameter of the Result(-) parameter of the Reset.response primitive and shall appear as the Program Invocation State sub-parameter of the Result(-) parameter of the Reset.confirm primitive, if issued.

11.8 Kill

The abstract syntax of the kill choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the Kill-Request and the Kill-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Kill-Request ::= SEQUENCE {
    programInvocationName  [0] IMPLICIT Identifier
    }

Kill-Response ::= NULL
```

11.8.1 Kill-Request

The abstract syntax of the kill choice of the ConfirmedServiceRequest shall be the Kill-Request.

11.8.2 Kill-Response

The abstract syntax of the kill choice of the ConfirmedServiceResponse shall be the Kill-Response.

11.9 GetProgramInvocationAttributes

The abstract syntax of the getProgramInvocationAttributes choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the GetProgramInvocationAttributes-Request and the GetProgramInvocationAttributes-Response, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetProgramInvocationAttributes-Request ::= Identifier
    -- Program Invocation Name

GetProgramInvocationAttributes-Response ::= SEQUENCE {
    state                   [0] IMPLICIT ProgramInvocationState,
    listOfDomainNames       [1] IMPLICIT SEQUENCE OF Identifier,
    mmsDeletable            [2] IMPLICIT BOOLEAN,
    reusable                [3] IMPLICIT BOOLEAN,
    monitor                 [4] IMPLICIT BOOLEAN,
    executionArgument       CHOICE {
       simpleString             [5] IMPLICIT VisibleString,
       encodedString            EXTERNAL
       }
    }
```

11.9.1 GetProgramInvocationAttributes-Request

The abstract syntax of the getProgramInvocationAttributes choice of the ConfirmedServiceRequest shall be the GetProgramInvocationAttributes-Request.

11.9.2 GetProgramInvocationAttributes-Response

The abstract syntax of the getProgramInvocationAttributes choice of the ConfirmedServiceResponse shall be the GetProgramInvocationAttributes-Response.

11.9.2.1 Execution Argument

The Execution Argument parameter of the GetProgramInvocationAttributes service response shall be a choice between a VisibleString, indicating that the value of this parameter is not further described and its interpretation is a local matter, or an EXTERNAL, indicating that the abstract syntax referenced by the EXTERNAL contains coding rules for interpreting the value of this parameter.

12 Variable Access Protocol

This clause describes the service specific protocol elements of the services which are defined by the Variable Access facility of the MMS service definition. This includes the protocol required for realization of the the following services:

Read;
Write;
InformationReport;
GetVariableAccessAttributes;
DefineNamedVariable;
DefineScatteredAccess
GetScatteredAccessAttributes;
DeleteVariableAccess;
DefineNamedVariableList;
GetNamedVariableListAttributes;
DeleteNamedVariableList;
DefineNamedType;
GetNamedTypeAttributes; and
DeleteNamedType.

12.1 Conventions

All protocol elements of this clause follow the conventions of 5.5, unless otherwise noted. Clarification is provided when these conventions do not apply exactly, or when the possibility exists for ambiguous interpretation.

In addition to the conventions of 5.5, in this clause the relationship between an enumerated parameter of the service definition and the protocol shall be as follows.

a) An enumerated parameter of the request (response) primitive which selects between the types of a CHOICE (by selecting among a list of mutually exclusive parameters), shall not appear in the protocol. Instead the selected parameter shall appear as the selection (of the CHOICE) having a derived reference name indicating the selected parameter. This selected parameter shall appear in the indication (confirm) primitive, if issued, as the enumerated parameter, with value as in the request (response) primitive, and the selected parameter.

b) When an enumerated parameter represents the values for an integer type, the correspondence between a value of the parameter and its value in the protocol shall be indicated by an ASN.1 comment.

A given parameter shall be defined in only one of these two ways.

12.2 Protocol For Specifying Types

12.2.1 TypeSpecification

The abstract syntax of the Type Specification parameter is specified below. The derivation of all parameters of this type is provided by 12.1 unless otherwise indicated.

```
TypeSpecification ::= CHOICE {
    typeName          [0] ObjectName,
    array             [1] IMPLICIT SEQUENCE {
        packed            [0] IMPLICIT BOOLEAN DEFAULT FALSE,
        numberOfElements  [1] IMPLICIT Unsigned32,
        elementType       [2] TypeSpecification
        },
    structure         [2] IMPLICIT SEQUENCE {
        packed            [0] IMPLICIT BOOLEAN DEFAULT FALSE,
        components        [1] IMPLICIT SEQUENCE OF SEQUENCE {
            componentName     [0] IMPLICIT Identifier OPTIONAL,
            componentType     [1] TypeSpecification
            }
        },
    -- Simple                          Size              Class
    boolean           [3] IMPLICIT NULL,          -- BOOLEAN
    bit-string        [4] IMPLICIT Integer32,     -- BIT-STRING
    integer           [5] IMPLICIT Unsigned8,     -- INTEGER
    unsigned          [6] IMPLICIT Unsigned8,     -- UNSIGNED
    floating-point    [7] IMPLICIT SEQUENCE {
          format-width      Unsigned8,            -- number of bits of
                                                  -- floating point value
                                                  -- including sign, exponent,
                                                  -- and fraction
          exponent-width    Unsigned8             -- size of exponent in bits
          },
    real              [8] IMPLICIT SEQUENCE {
          base              [0] IMPLICIT INTEGER(2|10),
          exponent          [1] IMPLICIT INTEGER OPTIONAL,
    -- shall appear if and only if base = 2
          mantissa          [2] IMPLICIT INTEGER OPTIONAL
    -- shall appear if and only if base = 2
          },
    octet-string      [9] IMPLICIT Integer32,   -- OCTET-STRING
    visible-string    [10] IMPLICIT Integer32,   -- VISIBLE-STRING
    generalized-time  [11] IMPLICIT NULL,       -- GENERALIZED-TIME
    binary-time       [12] IMPLICIT BOOLEAN,    -- BINARY-TIME
    bcd               [13] IMPLICIT Unsigned8,   -- BCD
    objId             [15] IMPLICIT NULL
}
```

12.3 Protocol For Specifying Alternate Access

12.3.1 AlternateAccess

The abstract syntax of the Alternate Access parameter is specified below. The derivation of all parameters of this type is provided by 12.1 unless otherwise indicated.

```
AlternateAccess ::= SEQUENCE OF CHOICE {
    unnamed AlternateAccessSelection,
    named   [5] IMPLICIT SEQUENCE {
        componentName [0] IMPLICIT Identifier,
        access        AlternateAccessSelection
        }
    }
```

```
AlternateAccessSelection ::= CHOICE {
    selectAlternateAccess [0] IMPLICIT SEQUENCE {
        accessSelection CHOICE {
            component    [0] IMPLICIT Identifier,    --  component
            index        [1] IMPLICIT Unsigned32,    --  1 array element
            indexRange   [2] IMPLICIT SEQUENCE {     --  array elements
                lowIndex          [0] IMPLICIT Unsigned32,
                numberOfElements  [1] IMPLICIT Unsigned32
                },
            allElements  [3] IMPLICIT NULL           --  all array elements
            },
        alternateAccess AlternateAccess
        },
    selectAccess CHOICE {
        component    [1] IMPLICIT Identifier,   --  component
        index        [2] IMPLICIT Unsigned32,    --  1 array element
        indexRange   [3] IMPLICIT SEQUENCE {     --  array elements
            lowIndex          [0] IMPLICIT Unsigned32,
            numberOfElements  [1] IMPLICIT Unsigned32
            },
        allElements  [4] IMPLICIT NULL           --  all array elements
        }
    }
```

The AlternateAccess type shall be the Alternate Access parameter. The elements of the List Of Alternate Access Selection parameter of this parameter shall be contained in corresponding elements of the AlternateAccess sequence-of type. Each element shall select either the named or the unnamed choice, based upon the presence or absence, respectively, of the Component Name parameter for that element of the List Of Alternate Access Selection parameter.

If an element of the List Of Alternate Access Selection parameter specifies the Component Name parameter, then the named selection for the choice representing that element shall be selected. In this case, componentName shall be the Component Name parameter and access shall be the AlternateAccessSelection type which shall specify the specific selection, as explained below.

If an element of the List Of Alternate Access Selection parameter does not specify the Component Name parameter, then the unnamed selection for the choice representing that element shall be selected. It shall be the AlternateAccessSelection type and shall specify the specific selection, as explained below.

The AlternateAccessSelection type shall be derived from the parameters (excluding Component Name) of the corresponding element of the List Of Alternate Access Selection parameter. The derivation of this type is as follows.

a) If Kind Of Selection specifies SELECT-ALTERNATE-ACCESS, then the selectAlternateAccess choice shall be selected. The parameters of this choice shall be derived as follows.

   1) The accessSelection field shall be derived from the Access Selection, Component, Index, and Index Range parameters according to 12.1. If the Access Selection parameter specifies INDEX-RANGE and if Low Index and Number Of Elements are both equal to zero, then the allElements choice for accessSelection may be selected in lieu of the indexRange choice, as a sender option. There is no semantic difference between these choices.

   2) The alternateAccess field shall be derived from the Alternate Access parameter by recursive reference to this procedure.

b) If Kind Of Selection specifies SELECT-ACCESS, then the selectAccess choice shall be selected. The accessSelection field shall be derived from the Access Selection, Component, Index, and Index Range parameters according to 12.1. If the Access Selection parameter specifies INDEX-RANGE and if Low Index and Number Of Elements are both equal to zero, then the allElements choice for accessSelection may be selected in lieu of the indexRange choice, as a sender option. There is no semantic difference between these choices.

12.4 Protocol For Specifying Data Values

12.4.1 AccessResult

The abstract syntax of the Access Result parameter shall be as specified below. The derivation of all parameters of this type is provided by 12.1 unless otherwise indicated.

```
AccessResult ::= CHOICE {
    failure [0] IMPLICIT DataAccessError,
    success Data
    }
```

The success field shall be indicated by the Success parameter of the Access Result parameter in a request (response) primitive having the value TRUE, and shall appear as the Success parameter of the indication (confirm) primitive, with value TRUE, if issued.

The failure field shall be indicated by the Success parameter of the Access Result parameter in a request (response) primitive having the value FALSE. It shall be the Data Access Error parameter of the Access Result and shall appear as the Success parameter, with value FALSE, and the Data Access Error parameter in the indication (confirm) primitive.

12.4.2 Data

The abstract syntax of the Data parameter shall be as specified below. The derivation of all parameters of this type is provided by 12.1 unless otherwise indicated.

```
Data ::= CHOICE {
    -- context tag 0 is reserved for AccessResult
    array                   [1] IMPLICIT SEQUENCE OF Data,
    structure               [2] IMPLICIT SEQUENCE OF Data,
    boolean                 [3] IMPLICIT BOOLEAN,
    bit-string              [4] IMPLICIT BIT STRING,
    integer                 [5] IMPLICIT INTEGER,
    unsigned                [6] IMPLICIT INTEGER,
    floating-point          [7] IMPLICIT FloatingPoint,
    real                    [8] IMPLICIT REAL,
    octet-string            [9] IMPLICIT OCTET STRING,
    visible-string         [10] IMPLICIT VisibleString,
    generalized-time       [11] IMPLICIT GeneralizedTime,
    binary-time            [12] IMPLICIT TimeOfDay,
    bcd                    [13] IMPLICIT INTEGER,
    booleanArray           [14] IMPLICIT BIT STRING,
    objId                  [15] IMPLICIT OBJECT IDENTIFIER
    }
```

12.4.2.1 Derivation

The derivation for the Data parameter is as follows:

a) When Kind Of Data is equal to ARRAY, then the array choice shall be selected and the content of this field shall be the List Of Data parameter of the Array parameter. Elements of List Of Data shall occur in the array field in the order listed in the List Of Data parameter.

The booleanArray choice in the Data production may be used (senders option) instead of the array choice when the data elements of an Array type are of boolean type. In this case, the elements of the List Of Data parameter of the Array, starting with element zero and proceeding to the end of the list, shall be placed into correspondingly numbered bits of the booleanArray. A value of true shall be represented by a one. A value of false shall be represented by a zero.

There is no semantic difference between the booleanArray and an array containing boolean values. The use of the booleanArray is a sender option, which may be chosen in order to increase the efficiency of transfer. All receivers shall be prepared to receive either form of boolean array.

b) When Kind Of Data is equal to STRUCTURE, then the structure choice shall be selected and the content of this field shall be the List Of Data parameter of the Structure parameter. Elements of List Of Data shall occur in the structure field in the order listed in the List Of Data parameter.

c) When Kind Of Data is equal to SIMPLE, then the value of the Class parameter shall select the choice for the Data as specified in 12.1

12.4.2.2 The FloatingPoint Type

```
FloatingPoint ::= OCTET STRING
```

The FloatingPoint type defines a simple type with distinguished values which are the positive and negative real numbers, including zero, and includes a representation for positive and negative infinity, and NaN (Not A Number). The possible values which a FloatingPoint may take are constrained by the representation described below.

The FloatingPoint real value shall be described as consisting of a sign S, a significand M, and exponent E, and an exponent width N, where N is greater than zero (N > 0). The significand is a number in the range

```
For E = 0
        0.0 <= M < 1.0

Otherwise
        1.0 <= M < 2.0
```

When a FloatingPoint value is represented, the FloatingPoint value shall contain four parts, as follows:

a) exponent width part - shall specify the number of bits contained in the exponent part;

b) sign part - shall specify the sign of the FloatingPoint;

c) exponent part - shall specify the value of the exponent;

d) fraction part - shall specify the value of field of the significand that lies to the right of its binary point (in a base 2 representation).

The four parts of the FloatingPoint shall be represented in an OCTET STRING containing two or more octets. The first octet of this OCTET STRING shall contain the exponent width part, represented as a binary integer. The remaining parts of the FloatingPoint shall be represented in the subsequent octets of the OCTET STRING as follows:

a) Number the bits of the subsequent octets from zero to "k", with bit zero as the most significant bit of the first subsequent octet and bit "k" as the least significant bit of the last subsequent octet. Using this numbering assign the bits of the parts of the floating-point value to the bits of the subsequent octets as follows:

   1) The sign part shall be assigned to bit zero. A positive sign shall be represented as a zero. A negative sign shall be represented as a one.

   2) The exponent part shall be assigned, in order of decreasing bit significance, to bits one through "n", and

   3) The fraction part shall be assigned, in order of decreasing bit significance, to bits "n + 1" through "k".

   NOTE – Multiple representations are possible for a single FloatingPoint value due to the possibility of differing values for exponent width. Semantic differences should not be assigned to the different representations of a single FloatingPoint value.

b) The value "NaN" shall be represented by all values having an exponent part containing all bits equal to one, and a fraction part containing at least one bit equal to one. The value of the sign bit shall be irrelevant.

c) Infinity shall be represented by all values having an exponent part containing all bits equal to one, and a fraction part containing all bits equal to zero. The value of the sign bit shall determine the sign of the infinity.

d) The value zero, shall be represented by all values containing all bits of the exponent and fraction parts equal to zero. The value of the sign bit shall be irrelevant.

e) A non-zero, finite FloatingPoint number shall be represented by a FloatingPoint value having an exponent part containing at least one bit equal to zero. The value of the represented FloatingPoint shall be determined by the equation:

$V = -1^{S} * F * 2^{(2-2^{(N-1)})}$ for E equal to zero

$V = -1^{S} * (1 + F) * 2^{(E-2^{(N-1)}+1)}$ otherwise

where the values of the terms of this equation are determined as follows:

1) "S" shall be the value of the sign bit.

2) "E", shall be the value of the exponent part.

3) "N" shall be the number of bits in the exponent part.

4) "F", shall be the sum of the weighted values of the bits of the fraction part.

The most significant bit of the fraction part (bit "N + 1" in the numbering specified in 1. above) shall have a weighted value equal to the value of the bit multiplied by 2^{-1}. The least significant bit of the fraction part (bit "k" in the numbering specified in 1, above) shall have a weighted value equal to the value of the bit multiplied by $2^{(N-K)}$

f) All other values shall be invalid.

The representation of the subsequent octets is compatible with the single precision IEEE 754 floating-point representation when the number of subsequent octets is four and the value of the initial octet is eight. Compatibility with double precision IEEE 754 floating-point representation exists when the number of subsequent octets is eight, and the value of the initial octet is eleven.

Since ISO/IEC 9506-1 and ISO/IEC 9506-2 allow any number of bits in the fraction and exponent parts of a floating-point value, (including, but not limited to, those specified for the formats defined in IEEE 754) and since real systems may not support all of the potential values for these terms, it may not be possible to guarantee that a specific value will be representable in the receiving system. When a floating-point value is received which is not representable in an implementation, the following rules shall apply:

a) If the value of the exponent is representable but the value of the fraction is not representable, the fraction shall be rounded to the nearest representable value if the most significant bit of the unrepresentable part of the fraction contains a one. Otherwise, the fraction shall be truncated.

b) If the value of the exponent is not representable and does not contain all bits equal to one, then

1) if the exponent is negative (the exponent part is less than the exponent bias) the implementation shall round to zero; otherwise

2) the implementation shall round to positive or negative infinity depending on the sign of the floating-point value.

c) If the value of the exponent is not representable and contains all bits equal to one, then

1) if the fraction contains all bits equal to zero, the implementation's representation for positive or negative infinity, as determined by the sign of the floating-point value, shall be used; otherwise

2) the implementation's representation for NaN shall be used.

12.4.2.3 The BCD Type

For data of type BCD, the value of the data shall be transfered using the equivalent integer value. For example, the BCD value 82, '10000010'B, shall be transfered as the integer value 82 or '1010010'B.

12.4.3 DataAccessError

The abstract syntax of the Data Access Error parameter is specified below. The derivation of all parameters of this type is provided by 12.1 unless otherwise indicated.

```
DataAccessError ::= INTEGER {
    object-invalidated            (0),    -- OBJECT-INVALIDATED
    hardware-fault                (1),    -- HARDWARE-FAULT
    temporarily-unavailable       (2),    -- TEMPORARILY-UNAVAILABLE
    object-access-denied          (3),    -- OBJECT-ACCESS-DENIED
    object-undefined              (4),    -- OBJECT-UNDEFINED
    invalid-address               (5),    -- INVALID-ADDRESS
    type-unsupported              (6),    -- TYPE-UNSUPPORTED
    type-inconsistent             (7),    -- TYPE-INCONSISTENT
    object-attribute-inconsistent (8),    -- OBJECT-ATTRIBUTE-INCONSISTENT
    object-access-unsupported     (9),    -- OBJECT-ACCESS-UNSUPPORTED
    object-non-existent           (10)    -- OBJECT-NON-EXISTENT
    }
```

12.5 Protocol for Specifying Access To Variables

12.5.1 VariableAccessSpecification

The abstract syntax of the Variable Access Specification parameter is specified below. The derivation of all parameters of this type is provided by 12.1 unless otherwise indicated.

```
VariableAccessSpecification ::=CHOICE {
    listOfVariable   [0] IMPLICIT SEQUENCE OF SEQUENCE {
        variableSpecification VariableSpecification,
        alternateAccess       [5] IMPLICIT AlternateAccess OPTIONAL
        },
    variableListName [1] ObjectName
    }
```

12.5.2 VariableSpecification

The abstract syntax of the Variable Specification parameter is given below. The derivation of all parameters of this type is provided by 12.1 unless otherwise indicated.

```
VariableSpecification ::= CHOICE {
    name                          [0] ObjectName,
    address                       [1] Address,
    variableDescription           [2] IMPLICIT SEQUENCE {
        address             Address,
        typeSpecification TypeSpecification
        },
    scatteredAccessDescription [3] IMPLICIT ScatteredAccessDescription,
    invalidated                   [4] IMPLICIT NULL
    }
```

The invalidated choice shall result from the Kind Of Variable parameter having a value of INVALIDATED and shall appear as the Kind Of Variable parameter having a value of INVALIDATED.

12.5.3 ScatteredAccessDescription

The abstract syntax of the Shattered Access Description parameter is specified below. The derivation of all parameters of this type is provided by 12.1 unless otherwise indicated.

```
ScatteredAccessDescription ::=SEQUENCE OF SEQUENCE {
    componentName          [0] IMPLICIT Identifier OPTIONAL,
    variableSpecification  [1] VariableSpecification,
    alternateAccess        [2] IMPLICIT AlternateAccess OPTIONAL
    }
```

12.5.4 Address

The abstract syntax of the Address parameter is specified below. The derivation of all parameters of this type is provided by 12.1 unless otherwise indicated.

```
Address ::= CHOICE {
    numericAddress        [0] IMPLICIT Unsigned32,
    symbolicAddress       [1] IMPLICIT VisibleString,
    unconstrainedAddress  [2] IMPLICIT OCTET STRING
    }
```

12.6 Read

The abstract syntax of the read choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Read-Request ::= SEQUENCE {
     specificationWithResult     [0] IMPLICIT BOOLEAN DEFAULT FALSE,
     variableAccessSpecification [1] VariableAccessSpecification
     }

Read-Response ::= SEQUENCE {
    variableAccessSpecification [0] VariableAccessSpecification OPTIONAL,
    listOfAccessResult          [1] IMPLICIT SEQUENCE OF AccessResult
    }
```

12.6.1 Read-Request

The abstract syntax of the read choice of the ConfirmedServiceRequest type shall be the Read-Request type.

12.6.2 Read-Response

The abstract syntax of the read choice of the ConfirmedServiceResponse type shall be the Read-Response type.

12.7 Write

The abstract syntax of the write choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Write-Request ::= SEQUENCE {
    variableAccessSpecification VariableAccessSpecification,
    listOfData                  [0] IMPLICIT SEQUENCE OF Data
    }
```

```
Write-Response ::= SEQUENCE OF CHOICE {
    failure [0] IMPLICIT DataAccessError,
    success [1] IMPLICIT NULL
    }
```

12.7.1 Write-Request

The abstract syntax of the write choice of the ConfirmedServiceRequest type shall be the Write-Request type.

12.7.2 Write-Response

The abstract syntax of the write choice of the ConfirmedServiceResponse type shall be the Write-Response type.

The success field shall be indicated by the Success parameter of the Write.response primitive having the value TRUE, and shall appear as the Success parameter, with value TRUE, in the Write.confirm primitive.

The failure field shall be indicated by the Success Parameter of the Write.response primitive having the value FALSE. It shall be the Data Access Error parameter of the Write.response primitive and shall appear as the Success parameter, with value FALSE, and the Data Access Error parameter of the Write.confirm primitive.

12.8 InformationReport

The abstract syntax of the informationReport choice of the UnconfirmedService type is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
InformationReport ::= SEQUENCE {
    variableAccessSpecification VariableAccessSpecification,
    listOfAccessResult          [0] IMPLICIT SEQUENCE OF AccessResult
    }
```

12.8.1 InformationReport

The abstract syntax of the informationReport choice of the UnconfirmedService type shall be the InformationReport type.

NOTE – The InformationReport service is not confirmed.

12.9 GetVariableAccessAttributes

The abstract syntax of the getVariableAccessAttributes choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetVariableAccessAttributes-Request ::= CHOICE {
    name    [0] ObjectName,
    address [1] Address
    }

GetVariableAccessAttributes-Response ::= SEQUENCE {
    mmsDeletable       [0] IMPLICIT BOOLEAN,
    address            [1] Address OPTIONAL,
    typeSpecification  [2] TypeSpecification
    }
```

12.9.1 GetVariableAccessAttributes-Request

The abstract syntax of the getVariableAccessAttributes choice of the ConfirmedServiceRequest type shall be the GetVariableAccessAttributes-Request type.

12.9.2 GetVariableAccessAttributes-Response

The abstract syntax of the getVariableAccessAttributes choice of the ConfirmedServiceResponse type shall be the GetVariableAccessAttributes-Response type.

12.10 DefineNamedVariable

The abstract syntax of the defineNamedVariable choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DefineNamedVariable-Request ::= SEQUENCE {
    variableName       [0] ObjectName,
    address            [1] Address,
    typeSpecification  [2] TypeSpecification OPTIONAL
    }

DefineNamedVariable-Response ::= NULL
```

12.10.1 DefineNamedVariable-Request

The abstract syntax of the defineNamedVariable choice of the ConfirmedServiceRequest type shall be the DefineNamedVariable-Request type.

12.10.2 DefineNamedVariable-Response

The abstract syntax of the defineNamedVariable choice of the ConfirmedServiceResponse type shall be the DefineNamedVariable-Response type, which shall be a NULL.

12.11 DefineScatteredAccess

The abstract syntax of the defineScatteredAccess choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DefineScatteredAccess-Request ::= SEQUENCE {
    scatteredAccessName        [0] ObjectName,
    scatteredAccessDescription [1] IMPLICIT ScatteredAccessDescription
    }

DefineScatteredAccess-Response ::= NULL
```

12.11.1 DefineScatteredAccess-Request

The abstract syntax of the defineScatteredAccess choice of the ConfirmedServiceRequest type shall be the DefineScatteredAccess-Request type.

12.11.2 DefineScatteredAccess-Response

The abstract syntax of the defineScatteredAccess choice of the ConfirmedServiceResponse type shall be the DefineScattered-Access-Response type, which shall be a NULL.

12.12 GetScatteredAccessAttributes

The abstract syntax of the getScatteredAccessAttributes choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetScatteredAccessAttributes-Request ::= ObjectName
                                        --ScatteredAccessName

GetScatteredAccessAttributes-Response ::= SEQUENCE {
    mmsDeletable                [0] IMPLICIT BOOLEAN,
    scatteredAccessDescription [1] IMPLICIT ScatteredAccessDescription
    }
```

12.12.1 GetScatteredAccessAttributes-Request

The abstract syntax of the getScatteredAccessAttributes choice of the ConfirmedServiceRequest type shall be the GetScatteredAccessAttributes-Request type.

12.12.2 GetScatteredAccessAttributes-Response

The abstract syntax of the getScatteredAccessAttributes choice of the ConfirmedServiceResponse type shall be the GetScatteredAccessAttributes-Response type.

12.13 DeleteVariableAccess

The abstract syntax of the deleteVariableAccess choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteVariableAccess-Request ::= SEQUENCE {
    scopeOfDelete [0] IMPLICIT INTEGER {
        specific     (0), -- SPECIFIC
        aa-specific (1), -- AA-SPECIFIC
        domain       (2), -- DOMAIN
        vmd          (3)  -- VMD
        } DEFAULT specific,
    listOfName    [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
    domainName    [2] IMPLICIT Identifier OPTIONAL
    }

DeleteVariableAccess-Response ::= SEQUENCE {
    numberMatched   [0] IMPLICIT Unsigned32,
    numberDeleted   [1] IMPLICIT Unsigned32
    }

DeleteVariableAccess-Error ::= Unsigned32 -- numberDeleted
```

12.13.1 DeleteVariableAccess-Request

The abstract syntax of the deleteVariableAccess choice of the ConfirmedServiceRequest type shall be the DeleteVariable-Access-Request type.

12.13.2 DeleteVariableAccess-Response

The abstract syntax of the deleteVariableAccess choice of the ConfirmedServiceResponse type shall be the DeleteVariableAccess-Response type.

12.14 DefineNamedVariableList

The abstract syntax of the defineNamedVariableList choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DefineNamedVariableList-Request ::= SEQUENCE {
    variableListName ObjectName,
    listOfVariable   [0] IMPLICIT SEQUENCE OF SEQUENCE {
        variableSpecification VariableSpecification,
        alternateAccess       [5] IMPLICIT AlternateAccess OPTIONAL
        }
    }

DefineNamedVariableList-Response ::= NULL
```

12.14.1 DefineNamedVariableList-Request

The abstract syntax of the defineNamedVariableList choice of the ConfirmedServiceRequest type shall be the DefineNamedVariableList-Request type.

12.14.2 DefineNamedVariableList-Response

The abstract syntax of the defineNamedVariableList choice of the ConfirmedServiceResponse type shall be the DefineNamedVariableList-Response type, which shall be a NULL.

12.15 GetNamedVariableListAttributes

The abstract syntax of the getNamedVariableListAttributes choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetNamedVariableListAttributes-Request ::= ObjectName
                                               -- VariableListName

GetNamedVariableListAttributes-Response ::= SEQUENCE {
    mmsDeletable    [0] IMPLICIT BOOLEAN,
    listOfVariable [1] IMPLICIT SEQUENCE OF SEQUENCE {
        variableSpecification VariableSpecification,
        alternateAccess       [5] IMPLICIT AlternateAccess OPTIONAL
        }
    }
```

12.15.1 GetNamedVariableListAttributes-Request

The abstract syntax of the getNamedVariableListAttributes choice of the ConfirmedServiceRequest type shall be the GetNamedVariableListAttributes-Request type.

12.15.2 GetNamedVariableListAttributes-Response

The abstract syntax of the getNamedVariableListAttributes choice of the ConfirmedServiceResponse type shall be the GetNamedVariableListAttributes-Response type.

12.16 DeleteNamedVariableList

The abstract syntax of the deleteNamedVariableList choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteNamedVariableList-Request ::= SEQUENCE {
    scopeOfDelete           [0] IMPLICIT INTEGER {
        specific    (0), -- SPECIFIC
        aa-specific (1), -- AA-SPECIFIC
        domain      (2), -- DOMAIN
        vmd         (3)  -- VMD
        } DEFAULT specific,
    listOfVariableListName [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
    domainName             [2] IMPLICIT Identifier OPTIONAL
    }

DeleteNamedVariableList-Response ::= SEQUENCE {
    numberMatched   [0] IMPLICIT Unsigned32,
    numberDeleted   [1] IMPLICIT Unsigned32
    }

DeleteNamedVariableList-Error ::= Unsigned32 -- numberDeleted
```

12.16.1 DeleteNamedVariableList-Request

The abstract syntax of the deleteNamedVariableList choice of the ConfirmedServiceRequest type shall be the DeleteNamedVariableList-Request type.

12.16.2 DeleteNamedVariableList-Response

The abstract syntax of the deleteNamedVariableList choice of the ConfirmedServiceResponse type shall be the DeleteNamedVariableList-Response type.

12.17 DefineNamedType

The abstract syntax of the defineNamedType choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DefineNamedType-Request ::= SEQUENCE {
    typeName          ObjectName,
    typeSpecification TypeSpecification
    }

DefineNamedType-Response ::= NULL
```

12.17.1 DefineNamedType-Request

The abstract syntax of the defineNamedType choice of the ConfirmedServiceRequest type shall be the DefineNamedType-Request type.

12.17.2 DefineNamedType-Response

The abstract syntax of the defineNamedType choice of the ConfirmedServiceResponse type shall be the DefineNamedType-Response type, which shall be a NULL.

12.18 GetNamedTypeAttributes

The abstract syntax of the getNamedTypeAttributes choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetNamedTypeAttributes-Request ::= ObjectName --TypeName

GetNamedTypeAttributes-Response ::= SEQUENCE {
    mmsDeletable       [0] IMPLICIT BOOLEAN,
    typeSpecification TypeSpecification
    }
```

12.18.1 GetNamedTypeAttributes-Request

The abstract syntax of the getNamedTypeAttributes choice of the ConfirmedServiceRequest type shall be the GetNamed-TypeAttributes-Request type.

12.18.2 GetNamedTypeAttributes-Response

The abstract syntax of the getNamedTypeAttributes choice of the ConfirmedServiceResponse type shall be the GetNamed-TypeAttributes-Response type.

12.19 DeleteNamedType

The abstract syntax of the deleteNamedType choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. 12.1 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteNamedType-Request ::= SEQUENCE {
    scopeOfDelete  [0] IMPLICIT INTEGER {
        specific     (0), -- SPECIFIC
        aa-specific (1), -- AA-SPECIFIC
        domain       (2), -- DOMAIN
        vmd          (3)  -- VMD
        } DEFAULT specific,
    listOfTypeName [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
    domainName     [2] IMPLICIT Identifier OPTIONAL
    }

DeleteNamedType-Response ::= SEQUENCE {
    numberMatched  [0] IMPLICIT Unsigned32,
    numberDeleted  [1] IMPLICIT Unsigned32
    }

DeleteNamedType-Error ::= Unsigned32 -- numberDeleted
```

12.19.1 DeleteNamedType-Request

The abstract syntax of the deleteNamedType choice of the ConfirmedServiceRequest type shall be the DeleteNamedType-Request type.

12.19.2 DeleteNamedType-Response

The abstract syntax of the deleteNamedType choice of the ConfirmedServiceResponse type shall be the DeleteNamedType-Response type.

13 Semaphore Management Protocol

13.1 Introduction

This clause describes the service-specific protocol elements of the Semaphore Management services:

a) TakeControl

b) RelinquishControl

c) DefineSemaphore

d) DeleteSemaphore

e) ReportSemaphoreStatus

f) ReportPoolSemaphoreStatus

g) ReportSemaphoreEntryStatus

In addition to the above services, this clause describes the specific protocol elements of the AttachToSemaphore modifier.

All protocol elements of this clause follow the conventions of 5.5, unless otherwise noted. Clarification is provided when these conventions do not apply or when the possibility exists for ambiguous interpretation.

13.2 TakeControl

The abstract syntax of the TakeControl choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
TakeControl-Request ::= SEQUENCE {
    semaphoreName              [0] ObjectName,
    namedToken                 [1] IMPLICIT Identifier OPTIONAL,
    priority                   [2] IMPLICIT Priority DEFAULT normalPriority,
    acceptableDelay            [3] IMPLICIT Unsigned32 OPTIONAL,
    controlTimeOut             [4] IMPLICIT Unsigned32 OPTIONAL,
    abortOnTimeOut             [5] IMPLICIT BOOLEAN OPTIONAL,
    relinquishIfConnectionLost [6] IMPLICIT BOOLEAN DEFAULT TRUE,
    applicationToPreempt       [7] IMPLICIT ApplicationReference OPTIONAL
    }

TakeControl-Response ::= CHOICE {
    noResult                   [0] IMPLICIT NULL,
    namedToken                 [1] IMPLICIT Identifier
    }
```

13.2.1 TakeControl-Request

The abstract syntax of the TakeControl choice of the ConfirmedServiceRequest type shall be TakeControl-Request.

The namedToken field shall be the Named Token parameter of the TakeControl.response primitive and shall appear as the Named Token parameter of the TakeControl.confirmation primitive.

13.2.2 TakeControl-Response

The abstract syntax of the TakeControl choice of the ConfirmedServiceResponse shall be TakeControl-Response.

13.3 RelinquishControl

The abstract syntax of the RelinquishControl choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
RelinquishControl-Request ::= SEQUENCE {
    semaphoreName              [0] ObjectName,
    namedToken                 [1] IMPLICIT Identifier OPTIONAL
    }

RelinquishControl-Response ::= NULL
```

13.3.1 RelinquishControl-Request

The abstract syntax for the RelinquishControl choice of the ConfirmedServiceRequest type shall RelinquishControl-Request.

13.3.2 RelinquishControl-Response

The abstract syntax for the RelinquishControl choice of the ConfirmedServiceResponse type shall be RelinquishControl-Response.

13.4 DefineSemaphore

The abstract syntax of the DefineSemaphore choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DefineSemaphore-Request ::= SEQUENCE {
    semaphoreName              [0] ObjectName,
    numberOfTokens             [1] IMPLICIT Unsigned16
    }

DefineSemaphore-Response ::= NULL
```

13.4.1 DefineSemaphore-Request

The abstract syntax of the DefineSemaphore choice of the ConfirmedServiceRequest type shall be DefineSemaphore-Request.

13.4.2 DefineSemaphore-Response

The abstract syntax of the DefineSemaphore choice of the ConfirmedServiceResponse type shall be DefineSemaphore-Response.

13.5 DeleteSemaphore

The abstract syntax of the DeleteSemaphore choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteSemaphore-Request ::= ObjectName  -- Semaphore Name

DeleteSemaphore-Response ::= NULL
```

13.5.1 DeleteSemaphore-Request

The abstract syntax of the DeleteSemaphore choice of the ConfirmedServiceRequest type shall be DeleteSemaphore-Request.

13.5.2 DeleteSemaphore-Response

The abstract syntax of the DeleteSemaphore choice of the ConfirmedServiceResponse type shall be DeleteSemaphore-Response.

13.6 ReportSemaphoreStatus

The abstract syntax of the ReportSemaphoreStatus choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
ReportSemaphoreStatus-Request ::= ObjectName  -- Semaphore Name

ReportSemaphoreStatus-Response ::= SEQUENCE {
    mmsDeletable        [0] IMPLICIT BOOLEAN,
    class               [1] IMPLICIT INTEGER {
        token (0),
        pool  (1)
       },
    numberOfTokens      [2] IMPLICIT Unsigned16,
    numberOfOwnedTokens [3] IMPLICIT Unsigned16,
    numberOfHungTokens  [4] IMPLICIT Unsigned16
    }
```

13.6.1 ReportSemaphoreStatus-Request

The abstract syntax of the ReportSemaphoreStatus choice of the ConfirmedServiceRequest type shall be ReportSemaphoreStatus-Request.

13.6.2 ReportSemaphoreStatus-Response

The abstract syntax of the ReportSemaphoreStatus choice of the ConfirmedServiceResponse type shall be ReportSemaphoreStatus-Response.

13.7 ReportPoolSemaphoreStatus

The abstract syntax of the reportPoolSemaphoreStatus choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
ReportPoolSemaphoreStatus-Request ::= SEQUENCE {
    semaphoreName           [0] ObjectName,
    nameToStartAfter        [1] IMPLICIT Identifier OPTIONAL
    }

ReportPoolSemaphoreStatus-Response ::= SEQUENCE {
    listOfNamedTokens     [0] IMPLICIT SEQUENCE OF
        CHOICE {
            freeNamedToken     [0] IMPLICIT Identifier,
            ownedNamedToken    [1] IMPLICIT Identifier,
            hungNamedToken     [2] IMPLICIT Identifier
           },
    moreFollows           [1] IMPLICIT BOOLEAN DEFAULT TRUE
    }
```

13.7.1 ReportPoolSemaphoreStatus-Request

The abstract syntax of the ReportPoolSemaphoreStatus choice of the ConfirmedServiceRequest type shall be ReportPoolSemaphoreStatus-Request.

13.7.2 ReportPoolSemaphoreStatus-Response

The abstract syntax of the ReportPoolSemaphoreStatus choice of the ConfirmedServiceResponse type shall be ReportPoolSemaphoreStatus-Response.

13.8 ReportSemaphoreEntryStatus

The abstract syntax of the ReportSemaphoreEntryStatus choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
ReportSemaphoreEntryStatus-Request ::= SEQUENCE {
    semaphoreName           [0] ObjectName,
    state                   [1] IMPLICIT INTEGER {
        queued  (0),
        owner   (1),
        hung    (2)
       },
    entryIDToStartAfter     [2] IMPLICIT OCTET STRING OPTIONAL
    }

ReportSemaphoreEntryStatus-Response ::= SEQUENCE {
    listOfSemaphoreEntry    [0] IMPLICIT SEQUENCE OF SemaphoreEntry,
    moreFollows             [1] IMPLICIT BOOLEAN DEFAULT TRUE
    }
```

13.8.1 ReportSemaphoreEntryStatus-Request

The abstract syntax of the ReportSemaphoreEntryStatus choice of the ConfirmedServiceRequest type shall be ReportSemaphoreEntryStatus-Request.

The encoding of the state field shall be 0 for QUEUED, 1 for OWNER and 2 for HUNG.

13.8.2 ReportSemaphoreEntryStatus-Response

The abstract syntax of the ReportSemaphoreEntryStatus choice of the ConfirmedServiceResponse type shall be ReportSemaphoreEntryStatus-Response.

13.9 AttachToSemaphore Modifier

The abstract syntax of the AttachToSemaphore choice of the Modifier type is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
AttachToSemaphore ::= SEQUENCE {
    semaphoreName              [0] ObjectName,
    namedToken                 [1] IMPLICIT Identifier OPTIONAL,
    priority                   [2] IMPLICIT Priority DEFAULT normalPriority,
    acceptableDelay            [3] IMPLICIT Unsigned32 OPTIONAL,
    controlTimeOut             [4] IMPLICIT Unsigned32 OPTIONAL,
    abortOnTimeOut             [5] IMPLICIT BOOLEAN OPTIONAL,
    relinquishIfConnectionLost [6] IMPLICIT BOOLEAN DEFAULT TRUE
    }
```

13.10 Semaphore and Resource Management Supporting Productions

```
SemaphoreEntry ::= SEQUENCE {
    entryID                    [0] IMPLICIT OCTET STRING,
    entryClass                 [1] IMPLICIT INTEGER {
          simple   (0),
          modifier (1)
          },
    applicationReference       [2] ApplicationReference,
    namedToken                 [3] IMPLICIT Identifier OPTIONAL,
    priority                   [4] IMPLICIT Priority DEFAULT normalPriority,
    remainingTimeOut           [5] IMPLICIT Unsigned32 OPTIONAL,
    abortOnTimeOut             [6] IMPLICIT BOOLEAN OPTIONAL,
    relinquishIfConnectionLost [7] IMPLICIT BOOLEAN DEFAULT TRUE
    }
```

14 Operator Communication Protocol

14.1 Introduction

This clause describes the PDUs for the operator communication services. Specifically, this clause specifies the protocol required for realization of the following services:

a) Input;

b) Output.

14.2 Input

The abstract syntax of the input choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Input-Request ::= SEQUENCE {
   operatorStationName [0] IMPLICIT Identifier,
   echo                [1] IMPLICIT BOOLEAN DEFAULT TRUE,
   listOfPromptData    [2] IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
   inputTimeOut        [3] IMPLICIT Unsigned32 OPTIONAL
   }

Input-Response ::= VisibleString -- Input String
```

14.2.1 Input-Request

The abstract syntax of the input choice of the ConfirmedServiceRequest shall be the Input-Request.

14.2.1.1 echo

The echo field shall be the Echo parameter from the Input.request primitive and shall appear as the Echo parameter of the Input.indication, if issued. The default value for the echo parameter is TRUE.

14.2.1.2 inputTimeOut

The inputTimeOut field shall be the Input Time Out parameter from the Input.request primitive and shall appear as the Input Time Out parameter of the Input.indication, if issued. The default value for this parameter when it is absent is an unlimited time out period.

14.2.2 Input-Response

The abstract syntax of the input choice for the ConfirmedServiceResponse shall be the Input-Response, which shall be a VisibleString.

14.3 Output

The abstract syntax of the output choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
Output-Request ::= SEQUENCE {
   operatorStationName [0] IMPLICIT Identifier,
   listOfOutputData    [1] IMPLICIT SEQUENCE OF VisibleString
   }

Output-Response ::= NULL
```

14.3.1 Output-Request

The abstract syntax of the output choice of the ConfirmedServiceRequest shall be the Output-Request.

14.3.2 Output-Response

The abstract syntax of the output choice for the ConfirmedServiceResponse is the Output-Response.

15 Event Management Protocol

15.1 Introduction

This clause describes the service-specific protocol elements of the services and service modifier which are defined by the Event Management Functional Unit of MMS. This includes the

DefineEventCondition,
DeleteEventCondition,
GetEventConditionAttributes,
ReportEventConditionStatus,
AlterEventConditionMonitoring,
TriggerEvent,
DefineEventAction,
DeleteEventAction,

GetEventActionAttributes,
ReportEventActionStatus,
DefineEventEnrollment,
DeleteEventEnrollment,
GetEventEnrollmentAttributes,
ReportEventEnrollmentStatus,
AlterEventEnrollment,
EventNotification,
AcknowledgeEventNotification,
GetAlarmSummary, and
GetAlarmEnrollmentSummary

services and the AttachToEventCondition service modifier.

15.2 DefineEventCondition

The abstract syntax of the defineEventCondition choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DefineEventCondition-Request ::= SEQUENCE {
    eventConditionName   [0] ObjectName,
    class                [1] IMPLICIT EC-Class,
    priority             [2] IMPLICIT Priority DEFAULT normalPriority,
    severity             [3] IMPLICIT Unsigned8 DEFAULT normalSeverity,
    alarmSummaryReports  [4] IMPLICIT BOOLEAN OPTIONAL,
    monitoredVariable    [6] VariableSpecification OPTIONAL,
    evaluationInterval   [7] IMPLICIT Unsigned32 OPTIONAL
    }

DefineEventCondition-Response ::= NULL
```

15.2.1 DefineEventCondition-Request

The abstract syntax of the defineEventCondition choice of the ConfirmedServiceRequest type shall be the DefineEventCondition-Request.

15.2.1.1 alarmSummaryReports

This parameter shall be omitted if the Class parameter of the DefineEventCondition.request service primitive has a value of NETWORK-TRIGGERED. Otherwise, this parameter shall be included.

15.2.2 DefineEventCondition-Response

The abstract syntax of the defineEventCondition choice of the ConfirmedServiceResponse type shall be the DefineEventCondition-Response.

15.3 DeleteEventCondition

The abstract syntax of the deleteEventCondition choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteEventCondition-Request ::= CHOICE {
    specific          [0] IMPLICIT SEQUENCE OF ObjectName,
    aa-specific       [1] IMPLICIT NULL,
    domain            [2] IMPLICIT Identifier,
    vmd               [3] IMPLICIT NULL
    }

DeleteEventCondition-Response ::= Unsigned32
                                  --Candidates Not Deleted
```

15.3.1 DeleteEventCondition-Request

The abstract syntax of the deleteEventCondition choice of the ConfirmedServiceRequest type shall be the DeleteEventCondition-Request. The value of this choice shall be determined as described below.

If the value of the Scope Of Delete parameter of the DeleteEventCondition.Request service primitive is equal to SPECIFIC, the DeleteEventCondition-Request shall contain the specific choice. This choice shall contain the value of the Event Condition Names parameter from the DeleteEventCondition.Request service primitive.

If the value of the Scope Of Delete parameter of the DeleteEventCondition.Request service primitive is equal to AA-SPECIFIC, the DeleteEventCondition-Request shall contain the aa-specific choice.

If the value of the Scope Of Delete parameter of the DeleteEventCondition.Request service primitive is equal to DOMAIN, the DeleteEventCondition-Request shall contain the domain choice. This choice shall contain the value of the Domain Name parameter from the DeleteEventCondition.Request service primitive.

If the value of the Scope Of Delete parameter of the DeleteEventCondition.Request service primitive is equal to VMD, the DeleteEventCondition-Request shall contain the vmd choice.

15.3.2 DeleteEventCondition-Response

The abstract syntax of the deleteEventCondition choice of the ConfirmedServiceResponse type is DeleteEventCondition-Response. This shall be the Candidates Not Deleted parameter from the DeleteEventCondition.response primitive indicating Result(+) and shall appear as the Candidates Not Deleted parameter of the DeleteEventCondition.confirm primitive indicating Result(+).

15.4 GetEventConditionAttributes

The abstract syntax of the getEventConditionAttributes choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetEventConditionAttributes-Request ::= ObjectName
                                        --Event Condition Name

GetEventConditionAttributes-Response ::= SEQUENCE {
    mmsDeletable        [0] IMPLICIT BOOLEAN DEFAULT FALSE,
    class               [1] IMPLICIT EC-Class,
    priority            [2] IMPLICIT Priority DEFAULT normalPriority,
    severity            [3] IMPLICIT Unsigned8 DEFAULT normalSeverity,
    alarmSummaryReports [4] IMPLICIT BOOLEAN DEFAULT FALSE,
    monitoredVariable   [6] CHOICE {
        variableReference   [0] VariableSpecification,
        undefined           [1] IMPLICIT NULL
        } OPTIONAL,
    evaluationInterval  [7] IMPLICIT Unsigned32 OPTIONAL
    }
```

15.4.1 GetEventConditionAttributes-Request

The abstract syntax of the getEventConditionAttributes choice of the ConfirmedServiceRequest type shall be the GetEventConditionAttributes-Request. This shall be the Event Condition Name parameter from the GetEventConditionAttributes.request primitive and shall appear as the Event Condition Name parameter of the GetEventConditionAttributes.indication primitive, if issued.

15.4.2 GetEventConditionAttributes-Response

The abstract syntax of the getEventConditionAttributes choice for the ConfirmedServiceResponse type is GetEventConditionAttributes-Response.

15.4.2.1 alarmSummaryReports

The value FALSE shall be provided if the Class parameter of the GetEventConditionAttributes.Response service primitive does not have the value MONITORED. Otherwise, the value shall be equal to the value of the Alarm Summary Reports parameter of the GetEventConditionAttributes.Response service primitive.

15.4.2.2 monitoredVariable

The undefined choice of the monitoredVariable field shall be selected if the Monitored Variable parameter of the GetEventConditionAttributes.Response service primitive has the value UNDEFINED. Otherwise, the variableReference choice shall be selected.

15.5 ReportEventConditionStatus

The abstract syntax of the reportEventConditionStatus choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
ReportEventConditionStatus-Request ::= ObjectName
                                       --Event Condition Name

ReportEventConditionStatus-Response ::= SEQUENCE {
    currentState                [0] IMPLICIT EC-State,
    numberOfEventEnrollments    [1] IMPLICIT Unsigned32,
    enabled                     [2] IMPLICIT BOOLEAN OPTIONAL,
    timeOfLastTransitionToActive [3] EventTime OPTIONAL,
    timeOfLastTransitionToIdle  [4] EventTime OPTIONAL
}
```

15.5.1 ReportEventConditionStatus-Request

The abstract syntax of the reportEventConditionStatus choice of the ConfirmedServiceRequest type shall be ReportEventConditionStatus-Request. This shall be the Event Condition Name parameter from the ReportEventConditionStatus.request primitive and shall appear as the Event Condition Name parameter of the ReportEventConditionStatus.indication primitive, if issued.

15.5.2 ReportEventConditionStatus-Response

The abstract syntax of the reportEventConditionStatus choice for the ConfirmedServiceResponse type shall be the ReportEventConditionStatus-Response.

15.6 AlterEventConditionMonitoring

The abstract syntax of the alterEventConditionMonitoring choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
AlterEventConditionMonitoring-Request ::= SEQUENCE {
    eventConditionName   [0] ObjectName,
    enabled              [1] IMPLICIT BOOLEAN OPTIONAL,
    priority             [2] IMPLICIT Priority OPTIONAL,
    alarmSummaryReports  [3] IMPLICIT BOOLEAN OPTIONAL,
    evaluationInterval   [4] IMPLICIT Unsigned32 OPTIONAL
    }

AlterEventConditionMonitoring-Response ::= NULL
```

15.6.1 AlterEventConditionMonitoring-Request

The abstract syntax of the alterEventConditionMonitoring choice of the ConfirmedServiceRequest type shall be AlterEventConditionMonitoring-Request.

At least one of enable, priority or alarmSummaryReports must be present.

15.6.2 AlterEventConditionMonitoring-Response

The abstract syntax of the alterEventConditionMonitoring choice for the ConfirmedServiceResponse type shall be the AlterEventConditionMonitoring-Response.

15.7 TriggerEvent

The abstract syntax of the triggerEvent choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
TriggerEvent-Request ::= SEQUENCE {
    eventConditionName   [0] ObjectName,
    priority             [1] IMPLICIT Priority OPTIONAL
    }

TriggerEvent-Response ::= NULL
```

15.7.1 TriggerEvent-Request

The abstract syntax of the triggerEvent choice of the ConfirmedServiceRequest type shall be TriggerEvent-Request.

15.7.2 TriggerEvent-Response

The abstract syntax of the triggerEvent choice for the ConfirmedServiceResponse type shall be the TriggerEvent-Response.

15.8 DefineEventAction

The abstract syntax of the defineEventAction choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DefineEventAction-Request ::= SEQUENCE {
    eventActionName          [0] ObjectName,
    listOfModifier           [1] IMPLICIT SEQUENCE OF Modifier OPTIONAL,
    confirmedServiceRequest  [2] ConfirmedServiceRequest,
    cs-extension             [79] CS-Request-Detail OPTIONAL
                                  -- shall not be transmitted if NULL
    }

DefineEventAction-Response ::= NULL
```

15.8.1 DefineEventAction-Request

The abstract syntax of the defineEventAction choice of the ConfirmedServiceRequest type shall be the DefineEventAction-Request.

15.8.1.1 ConfirmedServiceRequest

The abstract syntax of the Confirmed Service Request parameter of the Define Event Action service shall be the ConfirmedServiceRequest type followed by the choice of the CS-Request-Detail type which corresponds to the choice made of the ConfirmedServiceRequest.

15.8.2 DefineEventAction-Response

The abstract syntax of the defineEventAction choice for the ConfirmedServiceResponse type shall be the DefineEventAction-Response.

15.9 DeleteEventAction

The abstract syntax of the deleteEventAction choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteEventAction-Request ::= CHOICE {
    specific      [0] IMPLICIT SEQUENCE OF ObjectName,
    aa-specific   [1] IMPLICIT NULL,
    domain        [3] IMPLICIT Identifier,
    vmd           [4] IMPLICIT NULL
    }

DeleteEventAction-Response ::= Unsigned32
                                  --Candidates Not Deleted
```

15.9.1 DeleteEventAction-Request

The abstract syntax of the deleteEventAction choice of the ConfirmedServiceRequest type shall be the DeleteEventAction-Request. The value of this choice shall be determined as described below.

If the value of the Scope Of Delete parameter of the DeleteEventAction.Request service primitive is equal to SPECIFIC, the DeleteEventAction-Request shall contain the specific choice. This choice shall contain the value of the Event Action Names parameter of the DeleteEventAction.Request service primitive.

If the value of the Scope Of Delete parameter of the DeleteEventAction.Request service primitive is equal to AA-SPECIFIC, the DeleteEventAction-Request shall contain the aa-specific choice.

If the value of the Scope Of Delete parameter of the DeleteEventAction.Request service primitive is equal to DOMAIN, the DeleteEventAction-Request shall contain the domain choice. This choice shall contain the value of the Domain Name parameter of the DeleteEventAction.Request service primitive.

If the value of the Scope Of Delete parameter of the DeleteEventAction.Request service primitive is equal to VMD, the DeleteEventAction-Request shall contain the vmd choice.

15.9.2 DeleteEventAction-Response

The abstract syntax of the deleteEventAction choice of the ConfirmedServiceResponse type shall be the DeleteEventAction-Response. This shall be the Candidates Not Deleted parameter from the DeleteEventAction.response primitive indicating Result(+) and shall appear as the Candidates Not Deleted parameter of the DeleteEventAction.confirm primitive indicating Result(+).

15.10 GetEventActionAttributes

The abstract syntax of the getEventActionAttributes choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetEventActionAttributes-Request ::= ObjectName
                                     --EventActionName

GetEventActionAttributes-Response ::= SEQUENCE {
    mmsDeletable            [0] IMPLICIT BOOLEAN DEFAULT FALSE,
    listOfModifier          [1] IMPLICIT SEQUENCE OF Modifier,
    confirmedServiceRequest [2] ConfirmedServiceRequest,
    cs-extension            [79] CS-Request-Detail OPTIONAL
                                 -- shall not be transmitted if NULL

    }
```

15.10.1 GetEventActionAttributes-Request

The abstract syntax of the getEventActionAttributes choice of the ConfirmedServiceRequest type shall be the GetEventActionAttributes-Request. This shall be the Event Action Name parameter of the GetEventActionAttributes.request primitive and shall appear as the Event Action Name parameter of the GetEventActionAttributes.indication primitive, if issued.

15.10.2 GetEventActionAttributes-Response

The abstract syntax of the getEventActionAttributes choice for the ConfirmedServiceResponse type shall be the GetEventActionAttributes-Response.

15.10.2.1 ConfirmedServiceRequest

The abstract syntax of the Confirmed Service Request parameter of the Get Event Action Attributes service shall be the ConfirmedServiceRequest type followed by the choice of the CS-Request-Detail type which corresponds to the choice made of the ConfirmedServiceRequest.

15.11 ReportEventActionStatus

The abstract syntax of the reportEventActionStatus choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
ReportEventActionStatus-Request ::= ObjectName
                                    -- Event Action Name
ReportEventActionStatus-Response ::= Unsigned32
                                     -- Number of Event Enrollments
```

15.11.1 ReportEventActionStatus-Request

The abstract syntax of the reportEventActionStatus choice of the ConfirmedServiceRequest type shall be the ReportEventActionStatus-Request. This shall be the Event Action Name parameter of the ReportEventActionStatus.request primitive and shall appear as the Event Action Name parameter of the ReportEventActionStatus.indication primitive, if issued.

15.11.2 ReportEventActionStatus-Response

The abstract syntax of the reportEventActionStatus choice for the ConfirmedServiceResponse type shall be the ReportEventActionStatus-Response. This shall be indicated by a Result(+) containing the Number of Event Enrollments parameter in the ReportEventActionStatus.response service primitive, and shall appear as a Result(+) containing the Number of Event Enrollments parameter in the ReportEventActionStatus.confirm service primitive.

15.12 DefineEventEnrollment

The abstract syntax of the defineEventEnrollment choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DefineEventEnrollment-Request ::= SEQUENCE {
    eventEnrollmentName             [0] ObjectName,
    eventConditionName              [1] ObjectName,
    eventConditionTransitions       [2] IMPLICIT Transitions,
    alarmAcknowledgmentRule         [3] IMPLICIT AlarmAckRule,
    eventActionName                 [4] ObjectName OPTIONAL,
    clientApplication               [5] ApplicationReference OPTIONAL,
    acknowledgementEventCondition   [6] ObjectName OPTIONAL
   }
DefineEventEnrollment-Response ::= NULL
DefineEventEnrollment-Error ::= ObjectName
```

15.12.1 DefineEventEnrollment-Request

The abstract syntax of the defineEventEnrollment choice of the ConfirmedServiceRequest type shall be DefineEventEnrollment-Request.

15.12.2 DefineEventEnrollment-Response

The abstract syntax of the defineEventEnrollment choice for the ConfirmedServiceResponse type shall be the DefineEventEnrollment-Response.

15.12.3 DefineEventEnrollment-Error

The abstract syntax of the defineEventEnrollment choice of the serviceSpecificInformation choice of the ConfirmedServiceError shall be the DefineEventEnrollment-Error, which shall be the Object Not Defined sub-parameter of the Result(-) parameter of the DefineEventEnrollment.response primitive, and shall appear as the Object Not Defined of the Result(-) parameter of the DefineEventEnrollment.confirm primitive, if issued.

15.13 DeleteEventEnrollment

The abstract syntax of the deleteEventEnrollment choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteEventEnrollment-Request ::= CHOICE {
    specific            [0] IMPLICIT SEQUENCE OF ObjectName,
    ec                  [1] ObjectName,
    ea                  [2] ObjectName
    }

DeleteEventEnrollment-Response ::= Unsigned32
                                        --Candidates Not Deleted
```

15.13.1 DeleteEventEnrollment-Request

The abstract syntax of the deleteEventCondition choice of the ConfirmedServiceRequest type shall be the DeleteEventEnrollment-Request. The value of this choice shall be determined as described below.

If the value of the Scope Of Delete parameter of the DeleteEventEnrollment.Request service primitive is equal to EC, the DeleteEventEnrollment-Request shall contain the ec choice. This choice shall contain the value of the Event Condition Name parameter of the DeleteEventEnrollment.Request service primitive.

If the value of the Scope Of Delete parameter of the DeleteEventEnrollment.Request service primitive is equal to EA, the DeleteEventEnrollment-Request shall contain the ea choice. This choice shall contain the value of the Event Action Name parameter from the DeleteEventEnrollment.Request service primitive.

If the value of the Scope Of Delete parameter of the DeleteEventEnrollment.Request service primitive is equal to SPECIFIC, the DeleteEventEnrollment-Request shall contain the specific choice. This choice shall contain the value of the List Of Event Enrollment parameter from the DeleteEventEnrollment.Request service primitive.

15.13.2 DeleteEventEnrollment-Response

The abstract syntax of the deleteEventEnrollment choice of the ConfirmedServiceResponse type shall be the DeleteEventEnrollment-Response. This shall be the Candidates Not Deleted parameter from the DeleteEventEnrollment.response primitive indicating Result(+) and shall appear as the Candidates Not Deleted parameter of the DeleteEventEnrollment.confirm primitive indicating Result(+).

15.14 GetEventEnrollmentAttributes

The abstract syntax of the getEventEnrollmentAttributes choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetEventEnrollmentAttributes-Request ::= SEQUENCE {
    scopeOfRequest          [0] IMPLICIT INTEGER {
        specific (0), --SPECIFIC
        client   (1), -- CLIENT
        ec       (2), -- EC
        ea       (3)  -- EA
        } DEFAULT client,
    eventEnrollmentNames    [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
    clientApplication       [2] ApplicationReference OPTIONAL,
    eventConditionName      [3] ObjectName OPTIONAL,
    eventActionName         [4] ObjectName OPTIONAL,
    continueAfter           [5] ObjectName OPTIONAL
    }
```

```
GetEventEnrollmentAttributes-Response ::= SEQUENCE {
    listOfEventEnrollment  [0] IMPLICIT SEQUENCE OF EventEnrollment,
    moreFollows            [1] IMPLICIT BOOLEAN DEFAULT FALSE
    }

EventEnrollment ::= SEQUENCE {
    eventEnrollmentName           [0] ObjectName,
    eventConditionName            [1] CHOICE {
        eventCondition                [0] ObjectName,
        undefined                     [1] IMPLICIT NULL
        },
    eventActionName               [2] CHOICE {
        eventAction                   [0] ObjectName,
        undefined                     [1] IMPLICIT NULL
        }  OPTIONAL,
    clientApplication             [3] ApplicationReference OPTIONAL,
    mmsDeletable                  [4] IMPLICIT BOOLEAN DEFAULT FALSE,
    enrollmentClass               [5] IMPLICIT EE-Class,
    duration                      [6] IMPLICIT EE-Duration DEFAULT current,
    invokeID                      [7] IMPLICIT Unsigned32 OPTIONAL,
    remainingAcceptableDelay      [8] IMPLICIT Unsigned32 OPTIONAL,
    additionalDetail              [9] IMPLICIT EE-Additional-Detail
                                                          OPTIONAL,
        --additionalDetail shall be omitted if the value is NULL
    acknowledgementEventCondition [10] CHOICE {
        acknowledgementEvent          [0] ObjectName,
        undefined                     [1] IMPLICIT NULL
        } OPTIONAL
    }
```

15.14.1 GetEventEnrollmentAttributes-Request

The abstract syntax of the getEventEnrollmentAttributes choice of the ConfirmedServiceRequest type shall be the GetEventEnrollmentAttributes-Request.

The continueAfter field shall be the Enrollment ID of the Continue After parameter of the GetEventEnrollmentAttributes.request primitive and shall appear as the Enrollment ID of the Continue After parameter of the GetEventEnrollmentAttributes.indication primitive, if issued.

If the Continue After parameter is absent in the request primitive, this field shall be absent from the ConfirmedServiceRequest and the Continue After parameter shall be absent from the indication primitive, if issued.

15.14.2 GetEventEnrollmentAttributes-Response

The abstract syntax of the getEventEnrollmentAttributes choice for the ConfirmedServiceResponse type shall be the GetEventEnrollmentAttributes-Response.

15.14.2.1 listOfEventEnrollment

The listOfEventEnrollment field shall be the List Of Event Enrollment parameter of the GetEventEnrollmentAttributes.response primitive and shall appear as the List Of Event Enrollment parameter of the GetEventEnrollmentAttributes.confirm primitive. This field shall contain zero or more occurrences of the EventEnrollment type, each containing the value of a single Event Enrollment sub-parameter of the List Of Event Enrollment parameter, taken in the order listed. Subclause 5.5, shall be applied to each occurrence of the Event Enrollment sub-parameter of the List Of Event Enrollment parameter in order to derive the corresponding element of the listOfEventEnrollment sequence.

15.14.2.1.1 eventConditionName

The undefined choice of the eventConditionName field shall be selected if the Event Condition Name parameter of the GetEventEnrollmentAttributes.response service primitive has the value UNDEFINED. Otherwise, the eventCondition choice shall be selected.

15.14.2.1.2 eventActionName

If this field is included, the undefined choice of the eventActionName field shall be selected if the EventActionName parameter of the GetEventEnrollmentAttributes.response service primitive has the value UNDEFINED. Otherwise, the eventAction choice shall be selected.

15.14.2.1.3 duration

The value of this field shall have no meaning if the value of the enrollmentClass parameter of the GetEventEnrollmentAttributes.response service primitive has the value MODIFIER. In this case, this value should not be reported.

15.14.2.1.4 additionalDetail

The additionalDetail field of a given Event Enrollment shall be the Additional Detail parameter from the corresponding Event Enrollment of the List Of Event Enrollment parameter of the GetEventEnrollmentAttributes.response primitive and shall appear as the Additional Detail parameter of the corresponding Event Enrollment in the List Of Event Enrollment parameter of the GetEventEnrollmentAttributes.confirm primitive. The abstract syntax of this field shall be as defined in the Companion Standard referenced by the current abstract syntax. If the value of this field is NULL, this field shall be omitted.

15.14.2.1.5 acknowledgementEventCondition

If this field is included, the undefined choice of the acknowledgementEventCondition field shall be selected if the acknowledgementEventCondition parameter of the GetEventEnrollmentAttributes.response service primitive has the value UNDEFINED. Otherwise, the acknowledgementEvent choice shall be selected.

15.15 ReportEventEnrollmentStatus

The abstract syntax of the reportEventEnrollmentStatus choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
ReportEventEnrollmentStatus-Request ::= ObjectName
                                        --Event Enrollment Name
ReportEventEnrollmentStatus-Response ::= SEQUENCE {
    eventConditionTransitions       [0] IMPLICIT Transitions,
    notificationLost                [1] IMPLICIT BOOLEAN DEFAULT FALSE,
    duration                        [2] IMPLICIT EE-Duration,
    alarmAcknowledgmentRule         [3] IMPLICIT AlarmAckRule OPTIONAL,
    currentState                    [4] IMPLICIT EE-State
    }
```

15.15.1 ReportEventEnrollmentStatus-Request

The abstract syntax of the reportEventEnrollmentStatus choice of the ConfirmedServiceRequest type shall be the ReportEventEnrollmentStatus-Request.

15.15.2 ReportEventEnrollmentStatus-Response

The abstract syntax of the reportEventEnrollmentStatus choice for the ConfirmedServiceResponse type shall be the ReportEventEnrollmentStatus-Response.

15.16 AlterEventEnrollment

The abstract syntax of the alterEventEnrollment choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
AlterEventEnrollment-Request ::= SEQUENCE {
    eventEnrollmentName        [0] ObjectName,
    eventConditionTransitions  [1] IMPLICIT Transitions OPTIONAL,
    alarmAcknowledgmentRule    [2] IMPLICIT AlarmAckRule OPTIONAL
    }

AlterEventEnrollment-Response ::= SEQUENCE {
    currentState      [0] CHOICE {
        state             [0] IMPLICIT EE-State,
        undefined         [1] IMPLICIT NULL
        },
    transitionTime    [1] EventTime
    }
```

15.16.1 AlterEventEnrollment-Request

The abstract syntax of the alterEventEnrollment choice of the ConfirmedServiceRequest type shall be AlterEventEnrollment-Request.

15.16.2 AlterEventEnrollment-Response

The abstract syntax of the alterEventEnrollment choice for the ConfirmedServiceResponse type shall be the AlterEvent-Enrollment-Response.

15.16.2.1 Current State

The undefined choice of the Current State parameter shall be selected if the value of the Current State parameter of the AlterEventEnrollment.confirm service primitive is equal to UNDEFINED. Otherwise, the state choice shall be selected.

15.17 EventNotification

The abstract syntax of the eventNotification choice of the UnconfirmedService type is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

NOTE – EventNotification is an unconfirmed service, thus it does not define a response or error type.

```
EventNotification ::= SEQUENCE {
    eventEnrollmentName      [0] ObjectName,
    eventConditionName       [1] ObjectName,
    severity                 [2] IMPLICIT Severity,
    currentState             [3] IMPLICIT EC-State OPTIONAL,
    transitionTime           [4] EventTime,
    notificationLost         [6] IMPLICIT BOOLEAN DEFAULT FALSE,
    alarmAcknowledgmentRule  [7] IMPLICIT AlarmAckRule OPTIONAL,
    actionResult             [8] IMPLICIT SEQUENCE {
        eventActionName       ObjectName,
        eventActionResult      CHOICE {
            success              [0] IMPLICIT SEQUENCE {
                                     ConfirmedServiceResponse,
                                     [79]CS-Response-Detail OPTIONAL
                                 -- shall not be transmitted if value is NULL
                                     },
            failure              [1] IMPLICIT SEQUENCE {
                modifierPosition     [0] IMPLICIT Unsigned32 OPTIONAL,
                serviceError         [1] IMPLICIT ServiceError
                }
            }
        } OPTIONAL
    }
```

15.17.1 EventNotification

The abstract syntax of the eventNotification choice of the UnconfirmedService shall be the EventNotification. The derivation of the fields of this type is provided below.

15.17.1.1 actionResult

When present, the derivation of the actionResult field shall be as specified in 5.5. If the Action Result parameter is present in the EventNotification.request primitive, its eventActionResult field shall be determined as follows.

a) If the SuccessOrFailure sub-parameter of the Action Result parameter of the EventNotification.request primitive is equal to TRUE, the eventActionResult field shall select success and the value of the SuccessOrFailure parameter of the Action Result of the EventNotification.indication primitive, if issued, shall be TRUE. Otherwise, the eventActionResult field shall select failure and the value of the SuccessOrFailure parameter of the Action Result of the EventNotification.indication primitive, if issued, shall be FALSE.

b) If success is selected, the Result(+) parameter of the service requested by the Confirmed Service Request attribute of the Event Action object shall be conveyed using the ConfirmedServiceResponse type of the success selection, and 5.5 shall apply.

c) If failure is selected, the Result(-) parameter of the service requested by the ConfirmedServiceRequest attribute of the Event Action attribute shall be conveyed using the ServiceError type of the failure selection, and 5.5 shall apply.

15.17.1.1.1 ConfirmedServiceResponse

The abstract syntax of the Confirmed Service Response parameter of the Event Notification service shall be the ConfirmedServiceResponse type followed by the choice of the CS-Response-Detail type which corresponds to the choice made of the ConfirmedServiceResponse.

15.18 AcknowledgeEventNotification

The abstract syntax of the acknowledgeEventNotification choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
AcknowledgeEventNotification-Request ::= SEQUENCE {
    eventEnrollmentName           [0] ObjectName,
    acknowledgedState             [2] IMPLICIT EC-State,
    timeOfAcknowledgedTransition  [3] EventTime,
    acknowledgementEventCondition [4] ObjectName OPTIONAL
    }

AcknowledgeEventNotification-Response ::= NULL
```

15.18.1 AcknowledgeEventNotification-Request

The abstract syntax of the acknowledgeEventNotification choice of the ConfirmedServiceRequest type shall be the AcknowledgeEventNotification-Request.

15.18.2 AcknowledgeEventNotification-Response

The abstract syntax of the acknowledgeEventNotification choice for the ConfirmedServiceResponse type shall be the AcknowledgeEventNotification-Response.

15.19 GetAlarmSummary

The abstract syntax of the getAlarmSummary choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetAlarmSummary-Request ::= SEQUENCE {
    enrollmentsOnly             [0] IMPLICIT BOOLEAN DEFAULT TRUE,
    activeAlarmsOnly            [1] IMPLICIT BOOLEAN DEFAULT TRUE,
    acknowledgementFilter       [2] IMPLICIT INTEGER {
        not-acked (0), -- NOT-ACKED
        acked     (1), -- ACKED
        all       (2)  -- ALL
        } DEFAULT not-acked,
    severityFilter              [3] IMPLICIT SEQUENCE {
        mostSevere  [0] IMPLICIT Unsigned8,
        leastSevere [1] IMPLICIT Unsigned8
        } DEFAULT { mostSevere 0, leastSevere 127 },
    continueAfter               [5] ObjectName OPTIONAL
    }

GetAlarmSummary-Response ::= SEQUENCE {
    listOfAlarmSummary [0] IMPLICIT SEQUENCE OF AlarmSummary,
    moreFollows        [1] IMPLICIT BOOLEAN DEFAULT FALSE
    }
```

```
AlarmSummary ::= SEQUENCE {
    eventConditionName              [0] ObjectName,
    severity                        [1] IMPLICIT Unsigned8,
    currentState                    [2] IMPLICIT EC-State,
    unacknowledgedState             [3] IMPLICIT INTEGER {
        none (0),       -- NONE
        active (1),     -- ACTIVE
        idle (2),       -- IDLE
        both (3)        -- BOTH
        },
    additionalDetail                [4] EN-Additional-Detail OPTIONAL,
        -- additionalDetail shall be omitted if the value is NULL
    timeOfLastTransitionToActive [5] EventTime OPTIONAL,
    timeOfLastTransitionToIdle   [6] EventTime OPTIONAL
    }
```

15.19.1 GetAlarmSummary-Request

The abstract syntax of the getAlarmSummary choice of the ConfirmedServiceRequest type shall be the GetAlarmSummary-Request.

15.19.2 GetAlarmSummary-Response

The abstract syntax of the getAlarmSummary choice for the ConfirmedServiceResponse type shall be the GetAlarmSummary-Response. The derivation of the fields of this type is given below.

15.19.2.1 listOfAlarmSummary

The listOfAlarmSummary field shall be the List Of AlarmSummary parameter of the GetAlarmSummary.response primitive and shall appear as the List Of Alarm Summary parameter of the GetAlarmSummary.confirm primitive. This field shall contain zero or more occurrences of the AlarmSummary type, each containing the value of a single Alarm Summary specified in the List Of Alarm Summary parameter, taken in the order provided.

15.19.2.1.1 additionalDetail

The additionalDetail field of a given AlarmSummary shall be the Additional Detail parameter from the corresponding Alarm Summary of the List Of Alarm Summary parameter of the GetAlarmSummary.response primitive and shall appear as the Additional Detail parameter of the corresponding Alarm Summary in the List Of Alarm Summary parameter of the GetAlarmSummary.confirm primitive. The abstract syntax of this field shall be as defined in the Companion Standard referenced by the current abstract syntax. If the value of this field is NULL, this field shall be omitted.

15.20 GetAlarmEnrollmentSummary

The abstract syntax of the getAlarmEnrollmentSummary choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetAlarmEnrollmentSummary-Request ::= SEQUENCE {
    enrollmentsOnly             [0] IMPLICIT BOOLEAN DEFAULT TRUE,
    activeAlarmsOnly            [1] IMPLICIT BOOLEAN DEFAULT TRUE,
    acknowledgementFilter       [2] IMPLICIT INTEGER {
        not-acked (0), -- NOT-ACKED
        acked     (1), -- ACKED
        all       (2)  -- ALL
        } DEFAULT not-acked,
    severityFilter              [3] IMPLICIT SEQUENCE {
        mostSevere  [0] IMPLICIT Unsigned8,
        leastSevere [1] IMPLICIT Unsigned8
        } DEFAULT { mostSevere 0, leastSevere 127 },
    continueAfter               [5] ObjectName OPTIONAL
    }

GetAlarmEnrollmentSummary-Response ::= SEQUENCE {
    listOfAlarmEnrollmentSummary [0] IMPLICIT SEQUENCE OF
        AlarmEnrollmentSummary,
    moreFollows                  [1] IMPLICIT BOOLEAN DEFAULT FALSE
    }

AlarmEnrollmentSummary ::= SEQUENCE {
    eventEnrollmentName          [0] ObjectName,
    clientApplication            [2] ApplicationReference OPTIONAL,
    severity                     [3] IMPLICIT Unsigned8,
    currentState                 [4] IMPLICIT EC-State,
    additionalDetail             [5] EN-Additional-Detail OPTIONAL,
        -- additionalDetail shall be omitted if the value is NULL
    notificationLost             [6] IMPLICIT BOOLEAN DEFAULT FALSE,
    alarmAcknowledgmentRule      [7] IMPLICIT AlarmAckRule,
    enrollmentState              [8] IMPLICIT EE-State OPTIONAL,
    timeOfLastTransitionToActive [9] EventTime OPTIONAL,
    timeActiveAcknowledged      [10] EventTime OPTIONAL,
    timeOfLastTransitionToIdle  [11] EventTime OPTIONAL,
    timeIdleAcknowledged        [12] EventTime OPTIONAL
    }
```

15.20.1 GetAlarmEnrollmentSummary-Request

The abstract syntax of the getAlarmEnrollmentSummary choice of the ConfirmedServiceRequest type shall be the GetAlarmEnrollmentSummary-Request.

15.20.2 GetAlarmEnrollmentSummary-Response

The abstract syntax of the getAlarmEnrollmentSummary choice for the ConfirmedServiceResponse type shall be the GetAlarmEnrollmentSummary-Response. The derivation of the fields of this type is given below.

The listOfAlarmEnrollmentSummary field shall be the List Of Alarm Enrollment Summary parameter of the GetAlarmEnrollmentSummary.response primitive and shall appear as the List Of Alarm Enrollment Summary parameter of the GetAlarmEnrollmentSummary.confirm primitive. This field shall contain zero or more occurrences of the AlarmEnrollmentSummary type, each containing the value of a single Alarm Enrollment Summary specified in the List Of Alarm Enrollment Summary parameter, taken in the order provided.

15.20.2.0.1 additionalDetail

The additionalDetail field of a given AlarmEnrollmentSummary shall be the Additional Detail parameter from the corresponding Alarm Enrollment Summary of the List Of Alarm Enrollment Summary parameter of the GetAlarmEnrollmentSummary.response primitive and shall appear as the Additional Detail parameter of the corresponding Alarm Enrollment Summary in the List Of Alarm Enrollment Summary parameter of the GetAlarmEnrollmentSummary.confirm primitive. The abstract syntax of this field shall be as defined in the Companion Standard referenced by the current abstract syntax. If the value of this field is NULL, this field shall be omitted.

15.21 AttachToEventCondition

The abstract syntax of the attachToEventCondition choice of the Modifier type is specified by the AttachToEventCondition type. This type is specified below. Subclause 5.5 describes the derivation of all parameters of this type.

```
AttachToEventCondition ::= SEQUENCE {
    eventEnrollmentName [0] ObjectName,
    eventConditionName  [1] ObjectName,
    causingTransitions  [2] IMPLICIT Transitions,
    acceptableDelay     [3] IMPLICIT Unsigned32 OPTIONAL
    }
```

15.22 Supporting Productions

The abstract syntax of the various supporting type definitions for the event management protocol is given below. Other supporting productions can be found in clauses 7 and 12.

15.22.1 EC-Class

The EC-Class type conveys the value of the Class attribute of an Event Condition object.

```
EC-Class ::=  INTEGER {
    network-triggered (0),    -- NETWORK-TRIGGERED,
    monitored         (1)     -- MONITORED
    }
```

15.22.2 EC-State

The EC-State type conveys the value of the State attribute of an Event Condition object.

```
EC-State ::= INTEGER {
    disabled (0),    -- DISABLED
    idle     (1),    -- IDLE
    active   (2)     -- ACTIVE
    }
```

15.22.3 EE-State

The EE-State type conveys the value of the State attribute of an Event Enrollment object.

```
EE-State ::= INTEGER {
    disabled        (0),   -- DISABLED
    idle            (1),   -- IDLE
    active          (2),   -- ACTIVE
    activeNoAckA    (3),   -- ACTIVE-NO-ACK-A
    idleNoAckI      (4),   -- IDLE-NO-ACK-I
    idleNoAckA      (5),   -- IDLE-NO-ACK-A
    idleAcked       (6),   -- IDLE-ACKED
    activeAcked     (7)    -- ACTIVE-ACKED
    }
```

15.22.4 Transitions

The Transitions type conveys the value of the Causing Transitions attribute of an Event Enrollment object.

```
Transitions ::= BIT STRING {
    idle-to-disabled    (0), -- IDLE-TO-DISABLED
    active-to-disabled  (1), -- ACTIVE-TO-DISABLED
    disabled-to-idle    (2), -- DISABLED-TO-IDLE
    active-to-idle      (3), -- ACTIVE-TO-IDLE
    disabled-to-active  (4), -- DISABLED-TO-ACTIVE
    idle-to-active      (5), -- IDLE-TO-ACTIVE
    any-to-deleted      (6)  -- ANY-TO-DELETED
    }
```

15.22.5 AlarmAckRule

The AlarmAckRule type conveys the value of the Alarm Acknowledgment Rule attribute of an Event Enrollment object.

```
AlarmAckRule ::= INTEGER {
    none        (0), -- NONE
    simple      (1), -- SIMPLE
    ack-active  (2), -- ACK-ACTIVE
    ack-all     (3)  -- ACK-ALL
    }
```

15.22.6 EE-Class

The EE-Class type conveys the value of the Class attribute of an Event Enrollment object.

```
EE-Class ::=  INTEGER {
    modifier          (0),   -- MODIFIER
    notification      (1)    -- NOTIFICATION
    }
```

15.22.7 EE-Duration

The EE-Duration type conveys the value of the Duration attribute of an Event Enrollment object.

```
EE-Duration ::= INTEGER {
    current   (0), -- CURRENT
    permanent (1)  -- PERMANENT
    }
```

15.22.8 EventTime

The EventTime type conveys the value of time used to record event transitions in the TimeOfLastTransitionToActive and TimeOfLastTransitionToIdle attributes of Event Condition attributes of the Event Condition object, and the Time Active Acknowledged and Time Idle Acknowledged attributes of the Event Enrollment object. This type is also used to convey transition times in the EventNotification-Request, and in the AlterEventEnrollment-Response service primitives.

```
EventTime ::= CHOICE {
    timeOfDay               [0] IMPLICIT TimeOfDay,
    timeSequenceIdentifier  [1] IMPLICIT Unsigned32,
    undefined               [2] IMPLICIT NULL
    }
```

16 Journal Management Protocol

16.1 Introduction

This clause describes the service-specific protocol elements of the services which are defined by the Journal Management clause of the MMS Service Definition. This includes the following services:

ReadJournal,
WriteJournal,
InitializeJournal,
ReportJournalStatus,
CreateJournal, and
DeleteJournal

16.2 ReadJournal

The abstract syntax of the readJournal choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the ReadJournal-Request and ReadJournal-Response types, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
ReadJournal-Request ::= SEQUENCE {
    journalName                 [0] ObjectName,
    rangeStartSpecification     [1] CHOICE {
        startingTime     [0] IMPLICIT TimeOfDay,
        startingEntry    [1] IMPLICIT OCTET STRING
        } OPTIONAL,
    rangeStopSpecification      [2] CHOICE {
        endingTime       [0] IMPLICIT TimeOfDay,
        numberOfEntries  [1] IMPLICIT Integer32
        } OPTIONAL,
    listOfVariables             [4] IMPLICIT SEQUENCE OF VisibleString
                                    OPTIONAL,
    entryToStartAfter           [5] IMPLICIT SEQUENCE {
        timeSpecification   [0] IMPLICIT TimeOfDay,
        entrySpecification  [1] IMPLICIT OCTET STRING
        } OPTIONAL
    }

ReadJournal-Response ::= SEQUENCE {
    listOfJournalEntry      [0] IMPLICIT SEQUENCE OF
       JournalEntry,
    moreFollows             [1] IMPLICIT BOOLEAN DEFAULT FALSE
    }

JournalEntry ::= SEQUENCE {
    entryIdentifier         [0] IMPLICIT OCTET STRING,
    originatingApplication  [1] ApplicationReference,
    entryContent            [2] IMPLICIT EntryContent
    }
```

16.2.1 ReadJournal-Request

The abstract syntax of the readJournal choice of the ConfirmedServiceRequest shall be the ReadJournal-Request.

16.2.2 ReadJournal-Response

The abstract syntax of the readJournal choice of the ConfirmedServiceResponse shall be the ReadJournal-Response.

16.3 WriteJournal

The abstract syntax of the writeJournal choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the WriteJournal-Request and WriteJournal-Response types, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
WriteJournal-Request ::= SEQUENCE {
    journalName           [0] ObjectName,
    listOfJournalEntry    [1] IMPLICIT SEQUENCE OF EntryContent
    }

WriteJournal-Response ::= NULL
```

16.3.1 WriteJournal-Request

The abstract syntax of the writeJournal choice of the ConfirmedServiceRequest shall be the WriteJournal-Request.

16.3.2 WriteJournal-Response

The abstract syntax of the writeJournal choice of the ConfirmedServiceResponse shall be the WriteJournal-Response.

16.4 InitializeJournal

The abstract syntax of the initializeJournal choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the InitializeJournal-Request and InitializeJournal-Response types, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
InitializeJournal-Request ::= SEQUENCE {
    journalName          [0] ObjectName,
    limitSpecification   [1] IMPLICIT SEQUENCE {
        limitingTime   [0] IMPLICIT TimeOfDay,
        limitingEntry  [1] IMPLICIT OCTET STRING OPTIONAL
        } OPTIONAL
    }

InitializeJournal-Response ::= Unsigned32 -- Entries Deleted
```

16.4.1 InitializeJournal-Request

The abstract syntax of the initializeJournal choice of the ConfirmedServiceRequest shall be the InitializeJournal-Request.

16.4.2 InitializeJournal-Response

The abstract syntax of the initializeJournal choice of the ConfirmedServiceResponse shall be the InitializeJournal-Response.

This field shall be the Entries Deleted parameter of the InitializeJournal.response primitive and shall appear as the Entries Deleted parameter of the InitializeJournal.confirm primitive, if issued.

16.5 ReportJournalStatus

The abstract syntax of the reportJournalStatus choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the ReportJournalStatus-Request and ReportJournalStatus-Response types, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
ReportJournalStatus-Request ::= ObjectName    --Journal Name

ReportJournalStatus-Response ::= SEQUENCE {
    currentEntries              [0] IMPLICIT Unsigned32,
    mmsDeletable                [1] IMPLICIT BOOLEAN
    }
```

16.5.1 ReportJournalStatus-Request

The abstract syntax of the reportJournalStatus choice of the ConfirmedService Request shall be the ReportJournalStatus-Request.

This field shall be the Journal Name parameter of the ReportJournalStatus.request primitive and shall appear as the Journal Name parameter of the ReportJournalStatus.indication primitive, if issued.

16.5.2 ReportJournalStatus-Response

The abstract syntax of the reportJournalStatus choice of the ConfirmedService Response shall be the ReportJournalStatus-Response.

16.6 CreateJournal

The abstract syntax of the createJournal choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the CreateJournal-Request and CreateJournal-Response types, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
CreateJournal-Request ::= SEQUENCE {
    journalName       [0] ObjectName
    }

CreateJournal-Response ::= NULL
```

16.6.1 CreateJournal-Request

The abstract syntax of the createJournal choice of the ConfirmedServiceRequest shall be the CreateJournal-Request.

16.6.2 CreateJournal-Response

The abstract syntax of the createJournal choice of the ConfirmedServiceResponse shall be the CreateJournal-Response.

16.7 DeleteJournal

The abstract syntax of the deleteJournal choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the DeleteJournal-Request and DeleteJournal-Response types, respectively. These types are specified below and described in the paragraphs which follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteJournal-Request ::= SEQUENCE {
    journalName       [0] ObjectName
    }

DeleteJournal-Response ::= NULL
```

16.7.1 DeleteJournal-Request

The abstract syntax of the deleteJournal choice of the ConfirmedServiceRequest shall be the DeleteJournal-Request.

16.7.2 DeleteJournal-Response

The abstract syntax of the deleteJournal choice of the ConfirmedServiceResponse shall be the DeleteJournal-Response.

16.8 Supporting Productions

16.8.1 EntryContent

```
EntryContent ::= SEQUENCE {
    occurrenceTime          [0] IMPLICIT TimeOfDay,
    additionalDetail        [1] JOU-Additional-Detail
                                    OPTIONAL,
        -- additionalDetail shall be omitted in the abstract syntax
        -- defined in ISO/IEC 9506-2, clause 19
    entryForm                       CHOICE {
        data                [2] IMPLICIT SEQUENCE {
            event               [0] IMPLICIT SEQUENCE {
                eventConditionName  [0] ObjectName,
                currentState        [1] IMPLICIT EC-State
                } OPTIONAL,
            listOfVariables     [1] IMPLICIT SEQUENCE OF SEQUENCE {
                variableTag         [0] IMPLICIT VisibleString,
                valueSpecification  [1]  Data
                } OPTIONAL
            },
        annotation          [3] IMPLICIT VisibleString
        }
    }
```

16.8.1.1 additionalDetail

The additionalDetail field shall be the Additional Detail parameter of the ReadJournal.response primitive, or the Additional Detail parameter of the WriteJournal.request primitive, and shall appear as the Additional Detail parameter of the ReadJournal.confirm primitive, if issued, or the Additional Detail parameter of the WriteJournal.indication primitive, if issued.

The abstract syntax of this field shall be as defined in the Companion Standard referenced by the abstract syntax associated with the ReadJournal.response primitive, or the WriteJournal.request primitive. If this abstract syntax is the abstract syntax defined in ISO/IEC 9506-2, clause 19, this field shall be omitted.

16.8.1.2 entryForm

The data choice within the WriteJournal-Request shall be chosen if the value of the Entry Form parameter of the WriteJournal.request service primitive is DATAFORM. The annotation choice within the WriteJournal-Request shall be chosen if the value of the Entry Form parameter of the WriteJournal.request service primitive is ANNOTATIONFORM.

The data choice within the ReadJournal-Response shall be chosen if the value of the Entry Form parameter of the ReadJournal.response service primitive is DATAFORM. The annotation choice within the ReadJournal-Response shall be chosen if the value of the Entry Form parameter of the ReadJournal.response service primitive is ANNOTATIONFORM.

The following END statement terminates the module opened in clause 7.

```
END
```

17 Mapping to ACSE and Presentation Services

This clause defines the way in which the Association Control Service Element (ACSE) and the Presentation layer services are used by the Manufacturing Messaging Protocol Machine (MMPM). Any use of the ACSE service or presentation service other than as described in this clause shall constitute a protocol error.

The MMS protocol is positioned within the Open Systems Interconnection Environment within the application layer. As an Application Service Element (ASE), MMS makes use of and maps on to the services and service primitives of ACSE and the Presentation layer. The MMS-user may be elements within the application process or another ASE.

17.1 Mapping of PDUs

All MMS PDUs shall be carried as user data on an ACSE or presentation service primitive. The mapping of PDUs to services shall be as follows (all PDUs are sent on a request or response service primitive, and are received on an indication or confirm service primitive):

MMS PDU	ACSE or Presentation Service Primitive
Confirmed-RequestPDU	P-Data request, indication
Confirmed-ResponsePDU	P-Data request, indication
Confirmed-ErrorPDU	P-Data request, indication
Unconfirmed-RequestPDU	P-Data request, indication
RejectPDU	P-Data request, indication
Cancel-RequestPDU	P-Data request, indication
Cancel-ResponsePDU	P-Data request, indication
Cancel-ErrorPDU	P-Data request, indication
Initiate-RequestPDU	A-Associate request, indication
Initiate-ResponsePDU	A-Associate response, confirm (with Result parameter accepted)
Initiate-ErrorPDU	A-Associate response, confirm (with Result parameter rejected)
Conclude-RequestPDU	P-Data request, indication
Conclude-ResponsePDU	P-Data request, indication
Conclude-ErrorPDU	P-Data request, indication

Any other mappings of MMS PDUs on to ACSE and Presentation services shall constitute a protocol error.

17.2 Directly-Mapped Abort Service

The MMS abort service is directly mapped to the ACSE A-Abort service, and hence MMS does not define an abort PDU.

Upon receiving (from ACSE) an indication service primitive specifying an aborting ACSE service, the entity shall issue an abort indication service primitive to the MMS-user. If the ACSE abort request was generated by the system in which the MMS-user is located, the Locally Generated parameter in the MMS abort indication primitive shall specify the value true. Otherwise, this parameter shall specify the value false.

Upon receiving an MMS abort.request service primitive from the MMS-user, the MMPM shall issue an ACSE A-Abort.request service primitive;

The MMPM may, at any time, issue an abort.indication service primitive to the MMS-user and an ACSE A-Abort.request service primitive as a local matter (due to locally detected conditions).

17.3 Construction of MMS PDUs

Upon receipt of a request or response service primitive for any MMS service other than the abort.request service primitive, the MMPM shall

a) construct the PDU required by clause 7 for the service specified in the primitive, in accordance with the protocol requirements for that service (see clauses 7 to 16 and annexes B and C), and

b) send the constructed PDU as user data on the ACSE or Presentation service primitive specified above in accordance with the requirements of ISO/IEC 9506-1 and ISO/IEC 9506-2.

17.4 Delivery of Service Primitives to an MMS-user

Upon receipt of an indication or confirm service primitive from ACSE or Presentation other than an ACSE abort.indication, the MMPM shall determine if the service primitive received contains as user data a valid MMS PDU. A valid MMS PDU is one that meets the requirements of the MMS abstract syntax for the definition of PDUs, is mapped to the correct ACSE or presentation service primitive (as defined above), and arrives in conformance with all sequencing rules defined in ISO/IEC 9506-1 and ISO/IEC 9506-2.

If the service primitive received contains a valid MMS PDU, the MMPM shall issue the appropriate indication or confirm service primitive, with values in the primitive derived in accordance with the requirements in ISO/IEC 9506-1 and ISO/IEC 9506-2.

If the service primitive received does not contain a valid MMS PDU, the MMPM shall take the following actions:

a) if an Initiate-RequestPDU and an Initiate-ResponsePDU have been successfully exchanged via previous communications over the application association, the MMPM shall issue a reject.indication to the MMS-user, and shall construct a RejectPDU (with parameters based on the error detected) and send this PDU on a P-Data.request service primitive;

b) otherwise, the MMPM shall issue an abort.indication to the MMS-user, and issue an ACSE A-Abort.request service primitive if an application association exists.

17.5 Right to Send Data

MMS requires that the Duplex functional unit of the session layer be selected.

17.6 Reliable Underlying Service

MMS makes no provisions for handling of misordered messages, transmission errors, lost messages, or duplicated messages. MMS assumes that a reliable underlying service for communicating data between two application entities exists.

17.7 Flow Control

There is no peer flow control in MMS. The receiving MMPM may make use of the underlying transport flow control mechanism to effect flow control across the application association, thus limiting the ability of the peer to send data. The decision on when or how to make use of such flow control is a local matter.

17.8 Use of Presentation Contexts

MMS makes use of the multiple presentation contexts capability of the ISO Presentation Service Definition (ISO 8822). The MMS ASE makes use of one or more abstract syntaxes, such that each abstract syntax is either the abstract syntax defined in ISO/IEC 9506-1 and ISO/IEC 9506-2 by the ASN.1 Module ISO-9506-2-MMS-1 or is an abstract syntax defined by a Companion Standard.

The Presentation Context Definition List parameter from the ACSE A-Associate.request service primitive shall be used by the calling AE to propose a series of one or more abstract syntaxes. Similarly, the Presentation Context Result List parameter from the ACSE A-Associate.response service primitive shall be used by the responding AE to accept or reject each of the proposed elements (see ISO 8649 and ISO 8822).

This part of ISO/IEC 9506 defines two ASN.1 modules. The MMS-General-Module-1 is defined in clauses 6 to 16 and annexes B and C, and is intended for reference as a collection of common definitions only by the module in clause 19 and by Companion Standards. This module of general definitions is completed by definition of the fields reserved for use by Companion Standards to create an MMS compatible abstract syntax in accordance with the requirements of annex A of this part of ISO/IEC 9506 and annex A of ISO/IEC 9506-1. Clause 19 defines the MMS Core Abstract Syntax, which is for use when no Companion Standards are in use.

NOTE 1 – Hence, the MMS-General-Module-1 is used by clause 19 and Companion Standards to define MMS abstract syntaxes, and not by itself. The MMS Core Abstract Syntax, defined in clause 19, is for use of MMS without Companion Standards. MMS Companion Standards enhance the MMS-General-Module-1 to describe Companion Standard abstract syntaxes. The MMS-General-Module-1 is not intended for use other than by clause 19 and Companion Standards.

Each instance of communication under the MMS Application Context shall be governed by the MMS Core Abstract Syntax or that defined by an MMS Companion Standard. Over a single application association, it is possible that zero or more MMS abstract syntaxes may be negotiated and in use. Each individual PDU received is interpreted in accordance with the requirements of the Standard that defines the abstract syntax in which it arrives.

At any instant in time after the application association is established, the Presentation Defined Context Set (ISO 8822) shall indicate which abstract syntaxes may be used, and hence identifies which Companion Standard fields and semantics may be placed into effect. When an MMS service request is sent, the presentation context in which it arrives at the responding AE determines the abstract syntax, and thus the Companion Standard (or MMS Core Abstract Syntax) that shall govern the correct syntax and semantics of that service. For confirmed MMS services, each of the RejectPDU, Confirmed-ResponsePDU, Confirmed-ErrorPDU, Cancel-RequestPDU, Cancel-ResponsePDU, and Cancel-ErrorPDU that may issued as a part of a single service request (in accordance with the requirements of this part of ISO/IEC 9506) shall be issued in the same abstract syntax in which the corresponding Confirmed-RequestPDU was received.

NOTE 2 – Hence, the presentation context selection and identification mechanisms defined in ISO 8649 and ISO 8822 are used to operate MMS Companion Standards and negotiate their use.

17.9 Negotiation of MMS Abstract Syntaxes

In this clause, the term "MMS Abstract Syntax" shall represent an abstract syntax drawn from the set containing the abstract syntax defined in clause 19 and abstract syntaxes defined by MMS Companion Standards.

The MMS Initiate service requester shall create and send the Initiate PDU(s) as follows:

a) For each MMS Abstract Syntax whose use is to be proposed for the application association, the Initiate requester shall construct an Initiate-RequestPDU in accordance with the requirements of the Standard defining that MMS Abstract Syntax. The Initiate requester shall not construct an Initiate-RequestPDU for any MMS Abstract Syntax whose requirements it does not meet.

b) Each such Initiate-RequestPDU shall form an element in the sequence of User Data field of the ACSE A-ASSOCIATE.request service primitive such that no MMS abstract syntax appears in more than one element, and that these elements are ordered by the requester from most desirable for use to least desirable for use. Each element shall be a separate entity, and no express relationship need hold true between any pair of elements.

   NOTE 1 – Hence, the parts of MMS Initiate-RequestPDU defined in the MMS-General-Module-1 are repeated for each abstract syntax proposed, while the parts defined in clause 19 or by Companion Standards may differ for each element.

c) The Initiate Requester shall issue the A-ASSOCIATE.Request so constructed in accordance with the requirements of ISO/IEC 9506-1 and ISO/IEC 9506-2.

The MMS Initiate service responder, upon receipt of an A-ASSOCIATE.indication service primitive, shall respond to the Initiate service request as follows:

a) If the User Data field of the A-ASSOCIATE.indication service primitive does not contain any correctly formed MMS PDUs as defined by ISOISO/IEC 9506, the responder shall issue an A-ASSOCIATE.response service primitive with the Result parameter rejected and no MMS PDUs in the User Data field.

NOTE 2 – This may result from inability of the ACSE service provider to successfully negotiate support for the requested abstract syntaxes or from receipt of an incorrectly formed PDU or PDUs.

b) Otherwise, the responder shall, for each valid MMS Initiate-RequestPDU received, accept or reject the associated MMS Abstract Syntax by accepting or rejecting the associated presentation context. The responder shall place the value of accepted to indicate acceptance or user-rejected to indicate rejection of each presentation context in the corresponding element of the Presentation Context Definition Result List parameter of the ACSE A-ASSOCIATE.response primitive. For each valid MMS Initiate-RequestPDU received, whose presentation context is accepted, the responder shall send, in the same order as received, either an Initiate-ResponsePDU or an Initiate-ErrorPDU according to the Initiate service procedure.

The ordered list, constructed of Initiate-ResponsePDUs and/or Initiate-ErrorPDUs or any combination thereof, in the same order as the corresponding Initiate-RequestPDUs received, shall be sent in the User Data field of the A-ASSOCIATE.response service primitive such that each PDU forms an element of this User Data field. The service procedure of the Initiate service shall, for each Initiate-RequestPDU received, govern the issuance of the Initiate.indication service primitive and the handling of the resulting Initiate.response service primitive.

If the User Data field of the A-ASSOCIATE.indication service primitive contains an Initiate-RequestPDU formed in accordance with the requirements of clause 19 (the MMS Core Abstract Syntax), then the responder shall send at least one Initiate-ResponsePDU indicating acceptance of one of the proposed MMS Abstract Syntaxes (chosen by the responding system). Support for the responder role for the Initiate, Conclude, Abort, Reject and Identify services shall be provided, and indicated as required in that Initiate-ResponsePDU. Other MMS Services to be supported shall be a local matter, unless further constrained by an MMS Companion Standard for the MMS Abstract Syntax defined by that Companion Standard.

NOTE 3 – Implementors may find that support of services in the MMS Core Abstract Syntax is straight forward if these services are supported in a Companion Standard Abstract Syntax, since the Companion Standard service is often a selection of valid options from the MMS Core service requirements. ISO/IEC 9506-1, annex A, provides further details.

Receipt of a PDU other than an Initiate-RequestPDU (that meets all of the requirements of ISO/IEC 9506) in the User Data field of the A-ASSOCIATE.indication service primitive shall constitute a protocol error. The responder shall ignore any such PDU received and shall treat the corresponding MMS Abstract Syntax as not accepted.

If the User Data field of the A-ASSOCIATE.response service primitive to be issued in accordance with the requirements of ISO/IEC 9506 contains at least one Initiate-ResponsePDU, the the Result parameter of this primitive shall indicate accepted. Otherwise, the Result parameter of this primitive shall indicate rejected.

The results of this negotiation shall be as follows:

a) Both the requester and the responder shall consider each MMS Abstract Syntax for which an Initiate-ResponsePDU is received by the Initiate requester to be available for use for the duration of the application association. No other MMS Abstract Syntax shall be available.

b) For the Initiate service parameters defined in the MMS-General-Module-1 (and hence repeated in each Initiate-RequestPDU received), both the requester and the responder shall utilize those parameters in the first Initiate-RequestPDU in the ordered list for which an Initiate-ResponsePDU is sent. All others shall be ignored. These values shall be considered to have been negotiated between the requester and responder, and shall remain in effect for the duration of the application association.

c) For any MMS Abstract Syntax for which an Initiate-RequestPDU is sent by the Initiate requester, but no Initiate-ResponsePDU is received by the Initiate requester, this MMS Abstract Syntax shall not be available for use on the application association.

17.10 Termination of Application Association

Upon receipt of a valid Conclude-ResponsePDU, the MMPM shall issue an ACSE A-Release.request service primitive with no user data.

Upon receipt of an ACSE Release.indication service primitive (whose user data shall be ignored), the MMPM shall issue an ACSE A-Release.response service primitive with no user data, and with the result parameter set to indicate successful release of the application association.

Upon receipt of an ACSE Release.confirm service primitive (whose user data shall be ignored), with a result parameter which indicates successful release of the application association, the MMPM shall deliver a conclude.confirm service primitive indicating Result(+) to the MMS-user. If the result parameter indicates that the release attempt was unsuccessful, the MMPM shall issue an ACSE A-Abort.request service primitive and shall deliver a conclude.confirm service primitive indicating Result(+) to the MMS-user.

17.11 Abstract Syntax Definition

This part of ISO/IEC 9506 assigns the ASN.1 object identifier value

```
{iso standard 9506 part(2) mms-abstract-syntax-version1(1)}
```

as an abstract syntax for the set of presentation data values each of which is a value of the ASN.1 type MMS-General-Module-1.MMSpdu. The ASN.1 Module defined in clause 19 shall define this abstract syntax. The corresponding ASN.1 object descriptor value shall be

```
"mms-abstract-syntax-major-version1"
```

This part of ISO/IEC 9506 assigns the ASN.1 object identifier value

```
{iso standard 9506 part(2) mms-general-module-version1(2)}
```

as an ASN.1 module defined in of this part of ISO/IEC 9506, clauses 7 to 16, and annexes B and C. The corresponding ASN.1 object descriptor value shall be

```
"mms-general-module-major-version1"
```

The ASN.1 object identifier and object descriptor values

```
{ joint-iso-ccitt asn1(1) basic-encoding(1) }
```

and

```
"Basic Encoding of a single ASN.1 type"
```

(assigned to an information object in ISO 8825) can be used as a transfer syntax name with this abstract syntax. The major version number of this version of ISO/IEC 9506-1 and ISO/IEC 9506-2 shall be one. The minor version number of this version of ISO/IEC 9506-1 and ISO/IEC 9506-2 shall be one.

17.12 Application Context Name

For the purpose of being able to use an application which only contains the ACSE and MMS as ASEs, the object identifier value

```
{ iso standard 9506 part(2) mms-application-context-version1(3) }
```

and the object descriptor value

```
"ISO MMS"
```

are assigned to an information object of type

```
"ACSE-1.ApplicationContextName"
```

as defined in ISO 8650. Although this object identifier is assigned in this part of ISO/IEC 9506, and therefore includes the arc "part(2)", this application context name shall refer to the requirements placed by ISO/IEC 9506-1 and ISO/IEC 9506-2.

18 Conformance

18.1 Introduction

This clause describes the Protocol Implementation Conformance Statement Proforma (PICS). Every implementor shall complete the entire PICS. There are four parts to the PICS:

a) Part One of the PICS shall indicate information about the implementation and system.

b) Part Two of the PICS shall indicate which service CBBs are implemented.

c) Part Three of the PICS shall indicate which parameter CBBs are implemented, and their values if required by the CBB.

d) Part Four of the PICS shall indicate the local implementation values.

18.2 PICS Part One: Implementation Information

The implementor shall fill out Part One of the PICS with the indicated implementation values. Information for all of the items in Table 1 shall be supplied. For items requiring added information or explanation, the implementor shall put a reference in the table to an attached page, section, or paragraph provided by the implementor.

Table 1 – PICS Implementation Information

PICS Serial Number: Date Issued:

Implementation's Vendor Name	
Implementation's Model Name	
Implementation's Revision Identifier	
Machine Name(s) and Version Number(s)	
Operating System(s)	
MMS abstract syntax (Note 1.)	
MMS Version Number Supported (Note 2.)	
MMS Companion Standard abstract syntaxes (Note 3.)	
MMS Companion Standard Version Number Supported (Note 4.)	
Calling MMS-user (indicate "Yes" or "No")	
Called MMS-user (indicate "Yes" or "No")	
List of Standardized Names (Note 5.)	

Notes for PICS Part One:

1) MMS abstract syntax: The implementor shall state that the implementation supports the MMS abstract syntax as defined in this part of ISO/IEC 9506, clause 19.

2) MMS Version Number Supported: The implementor shall supply the major version number (part of the abstract syntax name) and the minor version number of the version of the abstract syntax defined in this part of ISO/IEC 9506, clause 19.

3) MMS Companion Standard abstract syntax: The implementor shall state all MMS Companion Standard abstract syntaxes which are supported. If there is none, then this item shall be left blank.

4) MMS Companion Standard Version Number Supported: This entry shall only be filled in if one or more MMS Companion Standard abstract syntaxes is supported. The implementor shall supply the abstract syntax name and version number of all such Standards.

5) List of Standardized Objects: This entry shall specify all of the MMS Standardized Objects and MMS Companion Standard Standardized Objects, by name, which are implemented.

18.3 PICS Part Two: Service CBBs

The implementor shall fill out the column in Table 2 which indicates whether the MMS implementation fulfils the server requirements, the client requirements, or both when operating in the abstract syntax defined in this part of ISO/IEC 9506, clause 19. This shall be done as "Server", "Client", or "Both", respectively. If the MMS implementation does not fulfil the server or client requirements for the service or CBB, then the column shall be left blank.

NOTE – The server and client requirements for each CBB are described in ISO/IEC 9506-1, clause 19.

Table 2 – PICS Service CBBs