INTERNATIONAL STANDARD

ISO 9506-2

First edition
2000-08-15

Corrected and reprinted
2000-12-15

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 9506-2:2000(E)

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

Printed in Switzerland

Contents

Page

Figures

Tables

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 commmittee 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.

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

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.

Attention is drawn to the possibility that some of the elements of this part of ISO 9506 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

International Standard ISO 9506-2 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 5, *Architecture, communications and integration frameworks*.

This first edition of ISO 9506-2 cancels and replaces ISO/IEC 9506-2:1990, of which it constitutes a technical revision.

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

— *Part 1: Service definition*
— *Part 2: Protocol specification*

Annexes A to C form a normative part of this part of ISO 9506. Annexes D to F are for information only.

Introduction

This part of ISO 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 that 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. This International Standard provides simple facilities for authentication (passwords) and access control. Systems requiring a higher degree of security will have to consider features beyond the scope of this International Standard. This International Standard does not provides facilities for non-repudiation.

Complexity of Services and Requirements

Some MMS services are quite complex and should be considered 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 9506 is one of a set of 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 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 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 9506-1. It uses services provided by the communication system that it employs for transferring its PDUs.

The MMS protocol is structured so that subsets of protocol can be defined. The variations and options available within this part of ISO 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 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 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 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 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.

Edition

This part of ISO 9506 differs from ISO/IEC 9506-2:1990 in the following ways:

a) The material in ISO/IEC TR 13345 to specify subsets of protocol for MMS has been incorporated into this part of ISO 9506.

b) All the material of Amendments 1 and 2 have been incorporated into the document, as well as the Technical Corrigenda.

c) The formal object model used in ISO 9506-1 provides some type definitions for the protocol specified in this part of ISO 9506. Hence, an IMPORT statement occurs in the ASN.1 module.

d) The services and protocol present in the Companion Standards already published, ISO/IEC 9506-3, ISO/IEC 9506-4, ISO/IEC 9506-5 and ISO/IEC 9506-6, have been incorporated into the base standard.

   As a result of this incorporation, the need for separate abstract syntaxes for each of the Companion Standards has been removed. All Companion Standards can now operate in the single abstract syntax of the base standard, although using other abstract syntaxes remains a possibility for backward compatibility. The separate definition of a module in Clause 19 of ISO/IEC 9506-2:1990 is no longer needed and this clause has been removed.

e) The communication requirements of MMS have been generalized so that MMS is described with respect to an abstract set of services needed for its support. The relation between this abstract set of services and the services provided by the suite of OSI communication protocols is specified in an annex. This opens the possibility of having MMS operating correctly over alternate communication systems (such as reduced stack implementations) as long as the equivalent of these abstract services are provided.

f) The restrictions on the characters that can be used as an Identifier have been relaxed to allow an Identifier to begin with a numeric character and, by extension, to consist solely of numeric characters.

g) Many (but not all) occurrences of VisibleString have been replaced by a new production MMSString that provides the option of using an extended latin alphabet, suitable for western Europe, and an option to use an arbitrary string of characters taken from ISO/IEC 10646 or from elsewhere.

h) A new service, ReconfigureProgramInvocation, has been introduced into the clause on Program Invocation management. This service provides a technique of dynamically changing the constituent Domains of a running Program Invocation.

i) A new field was added to the object model of the Named Variable and the Named Type. This field may be used to describe the semantics associated with the Named Variable or Named Type. The field is either predefined or it has its value established as the name of the Named Type used to construct it in the DefineNamedVariable or DefineNamedType service. This field can be reported with the GetVariableAccessAttributes or GetNamedTypeAttributes service if **sem**, a new parameter CBB, has been negotiated.

j) The material of the document has been reorganized to provide more and shorter clauses.

k) Scattered Access and the Real Data type have been removed from the base document and placed in informative annexes.

l) In accordance with the recommendations in ISO/IEC 8824-1, all occurrences of EXTERNAL in the protocol have been replaced with CHOICE { EXTERNAL, EMBEDDED PDV }.

m) The PICS of the first edition has been replaced by a clause providing configuration and initialization information. This clause provides initialization prescriptions for some fields (relatively few) of the VMD and subordinate objects, and provides a tabular report for initialization values of other fields as supplied by the implementor. A new annex (annex B) has been added that provides an ASN.1 module suitable for communicating the information contained in these tables.

Because of the use of the ASN.1 object modeling technique, the protocol now exists in two separate modules, one that is part of the object model contained in ISO 9506-1, and a second module defined in this part of ISO 9506 that describes the content and structure of all valid PDUs. Despite the fact that the ASN.1 formulation appears different in some cases, nevertheless the PDUs produced through application of ISO/IEC 9506:1990 are identical with those produced by this edition. For this reason, this edition continues to be identified by the major version number one. (The minor version number has been changed to reflect all the new additions to the document.)

There are two exceptions to this statement that should be noted.

a) Syntactic extensions defined by the companion standards are now identified by new parameter CBBs instead of a separate abstract syntax. Therefore, for any use of MMS involving companion standard facilities, there is a change in the Initiate PDU. However, if the companion standard facilities are not used, the Initiate PDU remains the same as that defined by the first edition.

b) Some small changes have been made to the tagging in the ChangeAccessControl service (part of Amendment 2) to bring it into alignment with corresponding protocol in the GetNameList and Rename services.

ASN.1 Modules

The ASN.1 modules defined in ISO 9506 may be obtained from the SC 4 Secretariat in computer readable format. The modules are available in two forms: as published and with the IF - ENDIF brackets removed.

To obtain these files use the Internet location: <http://forums.nema.org:8080/~iso_tc184_sc5>

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 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 abstract services provided by the underlying communication system.

1.3 Applicability

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

1.4 Conformance

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

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 9506. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 9506 are encouraged to investigate the possibility of applying the

most recent editions of the normative documents indicated below. For undated references, the latest eidition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 646:1991, *Information technology - 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/TR 8509:1987, *Information processing systems - Open Systems Interconnection - Service conventions.*

ISO 8571 (all parts), *Information processing systems - Open Systems Interconnection - File Transfer, Access and Management.*

ISO/IEC 8649-1:1996, *Information technolgy - Open Systems Interconnection - Service definition for the Association Control Service Element.*

ISO/IEC 8650:1996, *Information technology - Open Systems Interconnection - Connection-oriented protocol for the Association Control Service Element: Protocol specification.*

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

ISO/IEC 8824-1:1995, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

ISO/IEC 8824-1:1995/Amd. 1:1996, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation - Amendment 1: Rules of extensibility.*

ISO/IEC 8824-2:1995, *Information technology - Abstract Syntax Notation One (ASN.1): Information object specification.*

ISO/IEC 8824-2:1995/Amd. 1:1996, *Information technology - Abstract Abstract Syntax Notation One (ASN.1): Information object specification - Amendment 1: Rules of extensibility.*

ISO/IEC 8825-1:1995, *Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER, and Distinguished Encoding Rules (DER).*

ISO 9506-1:2000, *Industrial automation systems - Manufacturing Message Specification - Part 1: Service definition.*

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

ISO/IEC 10646-1:1993, *Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane*

ANSI/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 9506, the following definitions apply.

3.1 Reference Model definitions

This part of ISO 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;

h) (N) - layer;

i) system;

3.2 Service Convention definitions

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

a) value;

b) type;

c) simple type;

d) structure type;

e) component type;

f) tag;

g) tagging;

h) type (or value) reference name;

i) character string type;

j) boolean type;

k) true;

l) false;

m) integer type;

n) bitstring type;

o) octetstring type;

p) null type;

q) sequence type;

r) sequence-of type;

s) tagged type;

t) choice type;

u) selection type;

v) real type;

w) object identifier type;

x) module;

y) production;

z) ASN.1 encoding rules;

aa) ASN.1 character set;

ab) external type.

3.4 Other definitions

For the purposes of this part of ISO 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 that 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 that 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, event enrollments, and event condition lists.

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 that does not comply with the requirements of this part of ISO 9506 with respect to structure, meaning, or both.

3.4.17 journal:
a set of recorded, time-tagged event transitions, variable data, and/or comments, that 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 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 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 that 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 that 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 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 that most closely corresponds to an execution thread in a multi-tasking environment, 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 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 that behaves as an agent for a VMD for a particular service request instance.

3.4.39 standardized object:
an object instantiation whose definition is provided in ISO 9506-1 or in an MMS Companion Standard.

3.4.40 type:
an abstract description of a set of values that 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 that complies with the requirements of this part of ISO 9506 with respect to structure and 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
CIS	Configuration and Initialization Statement
FRSM	file read state machine
MMPM	Manufacturing Message Protocol Machine
MMS	Manufacturing Message Specification
OSI	Open Systems Interconnection
PDU	protocol data unit
ULSM	upload state machine
VMD	Virtual Manufacturing Device

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 that may convey parameters.

5.2 Base of Numeric Values

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

5.3 Notation

This part of ISO 9506 uses the abstract syntax notation defined in ISO/IEC 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 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 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 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 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, 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, 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 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, 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, 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 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 that 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 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 Calling and Called MMS-user

This part of ISO 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.

5.10 Sending and Receiving MMS-user and MMPM

This part of ISO 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 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.

5.11 Requesting and Responding MMS-user

This part of ISO 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.

5.12 Client and Server of a Service

This part of ISO 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 that behaves as a VMD for a particular service request instance. The Client is the peer communicating entity that 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.

5.13 ASN.1 Definitions

All ASN.1 definitions provided in this part of ISO 9506, clauses 7 to 23, and annexes C and D, 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 ISO 9506.

5.14 Protocol Subset Notation

The notation introduced by this part of ISO 9506 has the form of a preprocessor language in which ASN.1 is embedded. It is very similar in concept to the macro preprocessor for the C language. There are only three commands used in this notation:

- IF (<list of arguments>)

- ELSE

- ENDIF.

The IF command requires an argument list enclosed in parentheses; the arguments are the names of the conformance building blocks, either service or parameter. One or more such arguments must appear. If there is more than one argument, the arguments are separated by one or more spaces. The argument is treated as a boolean variable that has the value true if the corresponding service or parameter building block is supported as a result of the MMS Initiate exchange. If there is one argument, the lines following the IF statement up to the ELSE statement or to the matching ENDIF statement (if no ELSE statement appears) are to be included in the resulting ASN.1 definition if and only if the conformance building block so named in supported. If there is more than one argument, the lines following the IF statement are to be included if any of the conformance building blocks in the argument list is supported. (This can be thought of as a 'logical OR' function of the conformance building blocks.)

IF statements may be nested to any depth; the effect of

IF (x)
IF (y)

is to include the lines following these commands if and only if both x and y are true, that is if conformance block x and conformance block y are both included. (This can be thought of as a 'logical AND' function of the conformance building blocks.)

The ELSE statement may appear to allow ASN.1 statements to be included if a conformance building block is not true. Its use is similar to the normal use of ELSE in programming languages.

The ENDIF statement is used to end the scope of an IF statement or ELSE statement. Each IF statement must have a matching ENDIF statement.

5.15 Determination of the effective protocol

The protocol effective for any specific combination of service and parameter CBBs can be determined by the following procedure:

a) For each service CBB and parameter CBB declared or negotiated by the Initiate exchange, set the corresponding argument equal to true.

b) Process the entire ASN.1 module specified in this part of ISO 9506. For each IF statement, evaluate the argument.

i) If any of the elements in the argument is true, retain the statements following the IF statement up to a matching ENDIF or ELSE statement, if present. Discard the statements following the ELSE statement up to the matching ENDIF.

ii) If all the elements of the argument are false, discard the statements following up to the matching ELSE or ENDIF. If an ELSE statement is present, retain the statements following it to the matching ENDIF.

iii) Discard the IF statement, its matching ENDIF, and the ELSE, if present. The result should be an ASN.1 module devoid of IF, ELSE, and ENDIF statements.

c) In each production replace any occurrence of a comma followed immediately by a right brace with a right brace; i.e., delete such commas.

d) Form an ASN.1 working module of productions containing only the first production (i.e. the production MMSpdu from clause 7).

e) Add to the ASN.1 working module any productions referenced in the working module that are not already contained in that module.

f) Repeat step e) until no new productions are added.

The resulting ASN.1 module is the module that is effective for this combination of CBBs. Receipt of a PDU that is not consistent with this module should result in a reject.

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. 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 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 9506 describes the state diagram for each such service request instance independently.

NOTE Other clauses of this part of ISO 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 1 and Figure 2 are applicable for each of these 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 International 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

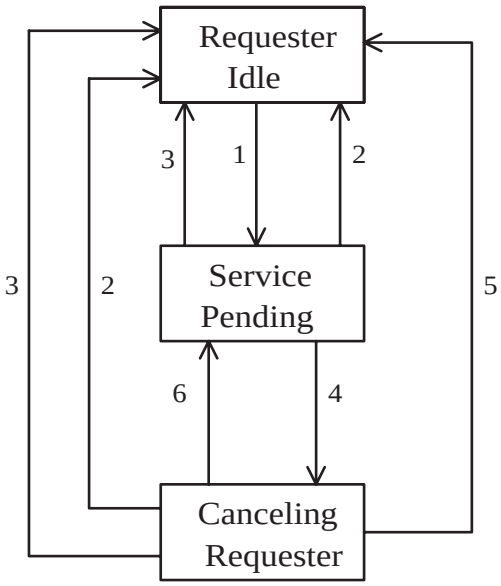

Figure 1 - Confirmed Service Request as seen by the Service Requester

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-

The order of receipt of the Cancel-ResponsePDU and Confirmed-ErrorPDU(x) in transition 5 of 6.3.1.1 shall not be relevant.

This subclause 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 that 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;

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 that 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 1 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 that 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 2 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

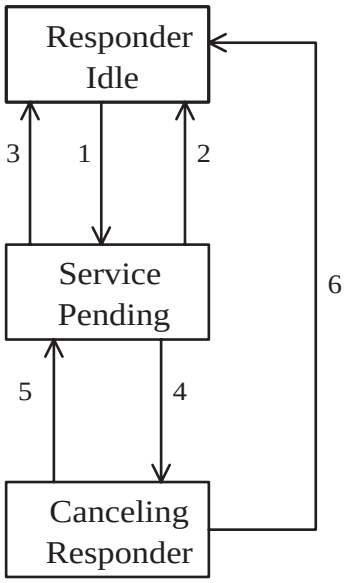

Figure 2 - Confirmed Service Request as seen by the Service Responder

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.reponse-
Cancel-ResponsePDU and Confirmed-ErrorPDU(x)

The order in which the Cancel.response+ and x.response- service primitives are issued in transition 6 of Figure 2 shall not be relevant.

Figure 2 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 1 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-PREEMPT 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-PREEMPT 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 2 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 Figure 3 and Figure 4 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

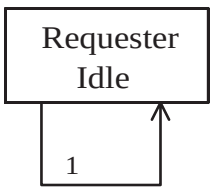

**Figure 3 - Unconfirmed Service
as seen by the Service Requester**

Transition:

1 - y.request
Unconfirmed-PDU(y)

Figure 3 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

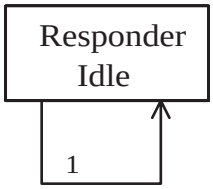

**Figure 4 - Unconfirmed Service
as seen by the Service Responder**

Transition:

1 - Unconfirmed-PDU(y)
y.indication

Figure 4 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. If the invalid PDU is determined to be an invalid RejectPDU, a RejectPDU shall not be sent.

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 9506-1, clause 7.

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 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 24. The mapping of MMS services to these PDUs is described in clauses 8 to 23.

```
 ISO-9506-MMS-1 { iso standard 9506 part(2) mms-abstract-syntax-version1(1) }
DEFINITIONS ::= BEGIN

    EXPORTS AlternateAccess,
    ConfirmedServiceRequest,
    Data,
    EE-State
    EventCondition,
    Identifier,
    Integer32,
    MMSString,
    ObjectName,
    TimeOfDay,
    TypeSpecification,
    Unsigned32,
    Unsigned8;

    IMPORTS AccessCondition,
    AdditionalCBBOptions,
    AdditionSupportOptions,
    Address,
    AlarmAckRule,
    Control-State,
    DomainState,
```

```
        EC-State,
        EC-Class,
        EE-Duration,
        EE-Class,
        EventTime,
        Journal-Variable,
        LogicalStatus,
        Modifier,
        normalPriority,
        normalSeverity,
        ParameterSupportOptions,
        PhysicalStatus,
        Priority,
        ProgramInvocationState,
        Running-Mode,
        ServiceSupportOptions,
        Severity,
        Transitions,
        TypeDescription,
        ULState,
        VMDState
        FROM MMS-Object-Module-1 { iso standard 9506 part(1) mms-object-model-version1(2) };

 MMSpdu ::= CHOICE {
        confirmed-RequestPDU    [0] IMPLICIT Confirmed-RequestPDU,
        confirmed-ResponsePDU   [1] IMPLICIT Confirmed-ResponsePDU,
        confirmed-ErrorPDU      [2] IMPLICIT Confirmed-ErrorPDU,
IF ( unsolicitedStatus informationReport eventNotification )
        unconfirmed-PDU         [3] IMPLICIT Unconfirmed-PDU,
ENDIF
        rejectPDU               [4] IMPLICIT RejectPDU,
IF (cancel)
        cancel-RequestPDU       [5] IMPLICIT Cancel-RequestPDU,
        cancel-ResponsePDU      [6] IMPLICIT Cancel-ResponsePDU,
        cancel-ErrorPDU         [7] IMPLICIT Cancel-ErrorPDU,
ENDIF
        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,
IF (attachToEventCondition attachToSemaphore )
        listOfModifiers         SEQUENCE OF Modifier OPTIONAL,
ENDIF
        service                 ConfirmedServiceRequest,
        ...,
IF ( csr cspi )
        service-ext             [79] Request-Detail OPTIONAL
                                -- shall not be transmitted if value is the value
                                -- of a tagged type derived from NULL
ENDIF
        }
```


The Confirmed-RequestPDU is a sequence containing four elements, an unsigned integer, an optional list of modifiers, a ConfirmedServiceRequest, and a 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 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 **listOfModifiers** shall be used to prescribe modifiers to the execution of service requests. A modifier specified in a **listOfModifiers** shall be successfully executed before the next modifier in the **listOfModifiers** or before execution of the service request. The order of the modifiers in the list is therefore important. If the **listOfModifiers** is not present, 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 7.1.1.

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 9506.

7.1.1 ConfirmedServiceRequest

```
ConfirmedServiceRequest ::= CHOICE {
IF ( status )
    status                              [0] IMPLICIT Status-Request,
ENDIF
IF ( getNameList )
    getNameList                         [1] IMPLICIT GetNameList-Request,
ENDIF
IF ( identify )
    identify                            [2] IMPLICIT Identify-Request,
ENDIF
IF ( rename )
    rename                              [3] IMPLICIT Rename-Request,
ENDIF
IF ( read )
    read                                [4] IMPLICIT Read-Request,
ENDIF
IF ( write )
    write                               [5] IMPLICIT Write-Request,
ENDIF
IF ( vnam vadr )
IF ( getVariableAccessAttributes )
    getVariableAccessAttributes         [6] GetVariableAccessAttributes-Request,
ENDIF
ENDIF
IF ( vnam )
IF ( vadr )
IF ( defineNamedVariable )
    defineNamedVariable                 [7] IMPLICIT DefineNamedVariable-Request,
ENDIF
ENDIF
ENDIF
IF ( vsca )
    --  [8] is reserved for a service defined in Annex E
IF ( defineScatteredAccess )
    defineScatteredAccess               [8] IMPLICIT DefineScatteredAccess-Request
ENDIF
    --  [9] is reserved for a service defined in Annex E
IF ( getScatteredAccessAttributes )
```

```
    getScatteredAccessAttributes          [9] IMPLICIT GetScatteredAccessAttributes-Request
ENDIF
ENDIF
IF ( vnam )
IF ( deleteVariableAccess )
    deleteVariableAccess                  [10] IMPLICIT DeleteVariableAccess-Request,
ENDIF
ENDIF
IF ( vlis )
IF ( vnam )
IF ( defineNamedVariableList )
    defineNamedVariableList               [11] IMPLICIT DefineNamedVariableList-Request,
ENDIF
IF ( getNamedVariableListAttributes )
    getNamedVariableListAttributes        [12] GetNamedVariableListAttributes-Request,
ENDIF
IF ( deleteNamedVariableList )
    deleteNamedVariableList               [13] IMPLICIT DeleteNamedVariableList-Request,
ENDIF
ENDIF
ENDIF
IF ( vnam )
IF ( defineNamedType )
    defineNamedType                       [14] IMPLICIT DefineNamedType-Request,
ENDIF
IF ( getNamedTypeAttributes )
    getNamedTypeAttributes                [15] GetNamedTypeAttributes-Request,
ENDIF
IF ( deleteNamedType )
    deleteNamedType                       [16] IMPLICIT DeleteNamedType-Request,
ENDIF
ENDIF
IF ( input )
    input                                 [17] IMPLICIT Input-Request,
ENDIF
IF ( output )
    output                                [18] IMPLICIT Output-Request,
ENDIF
IF ( takeControl )
    takeControl                           [19] IMPLICIT TakeControl-Request,
ENDIF
IF ( relinquishControl )
    relinquishControl                     [20] IMPLICIT RelinquishControl-Request,
ENDIF
IF ( defineSemaphore )
    defineSemaphore                       [21] IMPLICIT DefineSemaphore-Request,
ENDIF
IF ( deleteSemaphore )
    deleteSemaphore                       [22] DeleteSemaphore-Request,
ENDIF
IF ( reportSemaphoreStatus )
    reportSemaphoreStatus                 [23] ReportSemaphoreStatus-Request,
ENDIF
IF ( reportPoolSemaphoreStatus )
    reportPoolSemaphoreStatus             [24] IMPLICIT ReportPoolSemaphoreStatus-Request,
ENDIF
IF ( reportSemaphoreEntryStatus )
    reportSemaphoreEntryStatus            [25] IMPLICIT ReportSemaphoreEntryStatus-Request,
ENDIF
IF ( initiateDownloadSequence )
    initiateDownloadSequence              [26] IMPLICIT InitiateDownloadSequence-Request,
    downloadSegment                       [27] IMPLICIT DownloadSegment-Request,
    terminateDownloadSequence             [28] IMPLICIT TerminateDownloadSequence-Request,
```



```
ENDIF
IF ( initiateUploadSequence )
    initiateUploadSequence          [29] IMPLICIT InitiateUploadSequence-Request,
    uploadSegment                   [30] IMPLICIT UploadSegment-Request,
    terminateUploadSequence         [31] IMPLICIT TerminateUploadSequence-Request,
ENDIF
IF ( requestDomainDownload )
    requestDomainDownload           [32] IMPLICIT RequestDomainDownload-Request,
ENDIF
IF ( requestDomainUpload )
    requestDomainUpload             [33] IMPLICIT RequestDomainUpload-Request,
ENDIF
IF ( loadDomainContent )
    loadDomainContent               [34] IMPLICIT LoadDomainContent-Request,
ENDIF
IF ( storeDomainContent )
    storeDomainContent              [35] IMPLICIT StoreDomainContent-Request,
ENDIF
IF ( deleteDomain )
    deleteDomain                    [36] IMPLICIT DeleteDomain-Request,
ENDIF
IF ( getDomainAttributes )
    getDomainAttributes             [37] IMPLICIT GetDomainAttributes-Request,
ENDIF
IF ( createProgramInvocation )
    createProgramInvocation         [38] IMPLICIT CreateProgramInvocation-Request,
ENDIF
IF ( deleteProgramInvocation )
    deleteProgramInvocation         [39] IMPLICIT DeleteProgramInvocation-Request,
ENDIF
IF ( start )
    start                           [40] IMPLICIT Start-Request,
ENDIF
IF ( stop )
    stop                            [41] IMPLICIT Stop-Request,
ENDIF
IF ( resume )
    resume                          [42] IMPLICIT Resume-Request,
ENDIF
IF ( reset )
    reset                           [43] IMPLICIT Reset-Request,
ENDIF
IF ( kill )
    kill                            [44] IMPLICIT Kill-Request,
ENDIF
IF ( getProgramInvocationAttributes )
    getProgramInvocationAttributes  [45] IMPLICIT GetProgramInvocationAttributes-Request,
ENDIF
IF ( obtainFile )
    obtainFile                      [46] IMPLICIT ObtainFile-Request,
ENDIF
IF ( defineEventCondition )
    defineEventCondition            [47] IMPLICIT DefineEventCondition-Request,
ENDIF
IF ( deleteEventCondition )
    deleteEventCondition            [48] DeleteEventCondition-Request,
ENDIF
IF ( getEventConditionAttributes )
    getEventConditionAttributes     [49] GetEventConditionAttributes-Request,
ENDIF
IF ( reportEventConditionStatus )
    reportEventConditionStatus      [50] ReportEventConditionStatus-Request,
ENDIF
```

```
IF ( alterEventConditionMonitoring )
    alterEventConditionMonitoring          [51] IMPLICIT AlterEventConditionMonitoring-Request,
ENDIF
IF ( triggerEvent )
    triggerEvent                           [52] IMPLICIT TriggerEvent-Request,
ENDIF
IF ( defineEventAction )
    defineEventAction                      [53] IMPLICIT DefineEventAction-Request,
ENDIF
IF ( deleteEventAction )
    deleteEventAction                      [54] DeleteEventAction-Request,
ENDIF
IF ( getEventActionAttributes )
    getEventActionAttributes               [55] GetEventActionAttributes-Request,
ENDIF
IF ( reportEventActionStatus )
    reportEventActionStatus                [56] ReportEventActionStatus-Request,
ENDIF
IF ( defineEventEnrollment )
    defineEventEnrollment                  [57] IMPLICIT DefineEventEnrollment-Request,
ENDIF
IF ( deleteEventEnrollment )
    deleteEventEnrollment                  [58] DeleteEventEnrollment-Request,
ENDIF
IF ( alterEventEnrollment )
    alterEventEnrollment                   [59] IMPLICIT AlterEventEnrollment-Request,
ENDIF
IF ( reportEventEnrollmentStatus )
    reportEventEnrollmentStatus            [60] ReportEventEnrollmentStatus-Request,
ENDIF
IF ( getEventEnrollmentAttributes )
    getEventEnrollmentAttributes           [61] IMPLICIT GetEventEnrollmentAttributes-Request,
ENDIF
IF ( acknowledgeEventNotification )
    acknowledgeEventNotification           [62] IMPLICIT AcknowledgeEventNotification-Request,
ENDIF
IF ( getAlarmSummary )
    getAlarmSummary                        [63] IMPLICIT GetAlarmSummary-Request,
ENDIF
IF ( getAlarmEnrollmentSummary )
    getAlarmEnrollmentSummary              [64] IMPLICIT GetAlarmEnrollmentSummary-Request,
ENDIF
IF ( readJournal )
    readJournal                            [65] IMPLICIT ReadJournal-Request,
ENDIF
IF ( writeJournal )
    writeJournal                           [66] IMPLICIT WriteJournal-Request,
ENDIF
IF ( initializeJournal )
    initializeJournal                      [67] IMPLICIT InitializeJournal-Request,
ENDIF
IF ( reportJournalStatus )
    reportJournalStatus                    [68] ReportJournalStatus-Request,
ENDIF
IF ( createJournal )
    createJournal                          [69] IMPLICIT CreateJournal-Request,
ENDIF
IF ( deleteJournal )
    deleteJournal                          [70] IMPLICIT DeleteJournal-Request,
ENDIF
IF ( getCapabilityList )
    getCapabilityList                      [71] IMPLICIT GetCapabilityList-Request,
ENDIF
```

```
            -- choices [72] through [77] are reserved for use by services
            -- defined in annex D
IF ( fileOpen )
    fileOpen                        [72] IMPLICIT FileOpen-Request,
ENDIF
IF ( fileRead )
    fileRead                        [73] IMPLICIT FileRead-Request,
ENDIF
IF ( fileClose )
    fileClose                       [74] IMPLICIT FileClose-Request,
ENDIF
IF ( fileRename )
    fileRename                      [75] IMPLICIT FileRename-Request,
ENDIF
IF ( fileDelete )
    fileDelete                      [76] IMPLICIT FileDelete-Request,
ENDIF
IF ( fileDirectory )
    fileDirectory                   [77] IMPLICIT FileDirectory-Request,
ENDIF
IF ( csr cspi )
    additionalService               [78] AdditionalService-Request,
ENDIF
            -- choice [79] is reserved
IF ( getDataExchangeAttributes )
    getDataExchangeAttributes       [80] GetDataExchangeAttributes-Request,
                    -- Shall not appear in minor version 1
ENDIF
IF ( exchangeData )
    exchangeData                    [81] IMPLICIT ExchangeData-Request,
                    -- Shall not appear in minor version 1
ENDIF
IF ( defineAccessControlList )
    defineAccessControlList         [82] IMPLICIT DefineAccessControlList-Request,
                    -- Shall not appear in minor version 1 or 2
ENDIF
IF ( getAccessControlListAttributes )
    getAccessControlListAttributes  [83] IMPLICIT GetAccessControlListAttributes-Request,
                    -- Shall not appear in minor version 1 or 2
ENDIF
IF ( reportAccessControlledObjects )
    reportAccessControlledObjects   [84] IMPLICIT ReportAccessControlledObjects-Request,
                    -- Shall not appear in minor version 1 or 2
ENDIF
IF ( deleteAccessControlList )
    deleteAccessControlList         [85] IMPLICIT DeleteAccessControlList-Request,
                    -- Shall not appear in minor version 1 or 2
ENDIF
IF ( changeAccessControl )
    changeAccessControl             [86] IMPLICIT ChangeAccessControl-Request,
                    -- Shall not appear in minor version 1 or 2
ENDIF
    ...
    }
```

The ConfirmedServiceRequest type 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 that is referenced by the ConfirmedServiceRequest production. Each of the services in the ConfirmedServiceRequest choice is a confirmed service.

7.1.2 AdditionalService-Request

```
AdditionalService-Request ::= CHOICE {
IF ( csr )
IF ( vMDStop )
    vMDStop                                 [0] IMPLICIT VMDStop-Request,
ENDIF
IF ( vMDReset )
    vMDReset                                [1] IMPLICIT VMDReset-Request,
ENDIF
IF ( select )
    select                                  [2] IMPLICIT Select-Request,
ENDIF
IF ( alterProgramInvocationAttributes )
    alterPI                                 [3] IMPLICIT AlterProgramInvocationAttributes-Request,
ENDIF
ENDIF
IF ( cspi )
IF ( initiateUnitControlLoad )
    initiateUCLoad                          [4] IMPLICIT InitiateUnitControlLoad-Request,
ENDIF
IF ( unitControlLoadSegment )
    uCLoad                                  [5] IMPLICIT UnitControlLoadSegment-Request,
ENDIF
IF ( unitControlUpload )
    uCUpload                                [6] IMPLICIT UnitControlUpload-Request,
ENDIF
IF ( startUnitControl )
    startUC                                 [7] IMPLICIT StartUnitControl-Request,
ENDIF
IF ( stopUnitControl )
    stopUC                                  [8] IMPLICIT StopUnitControl-Request,
ENDIF
IF ( createUnitControl )
    createUC                                [9] IMPLICIT CreateUnitControl-Request,
ENDIF
IF ( addToUnitControl )
    addToUC                                 [10] IMPLICIT AddToUnitControl-Request,
ENDIF
IF ( removeFromUnitControl )
    removeFromUC                            [11] IMPLICIT RemoveFromUnitControl-Request,
ENDIF
IF ( getUnitControlAttributes )
    getUCAttributes                         [12] IMPLICIT GetUnitControlAttributes-Request,
ENDIF
IF ( loadUnitControlFromFile )
    loadUCFromFile                          [13] IMPLICIT LoadUnitControlFromFile-Request,
ENDIF
IF ( storeUnitControlToFile )
    storeUCToFile                           [14] IMPLICIT StoreUnitControlToFile-Request,
ENDIF
IF ( deleteUnitControl )
    deleteUC                                [15] IMPLICIT DeleteUnitControl-Request,
ENDIF
IF ( defineEventConditionList )
    defineECL                               [16] IMPLICIT DefineEventConditionList-Request,
ENDIF
IF ( deleteEventConditionList )
    deleteECL                               [17] IMPLICIT DeleteEventConditionList-Request,
ENDIF
IF ( addEventConditionListReference )
    addECLReference                         [18] IMPLICIT AddEventConditionListReference-Request,
```



```
ENDIF
IF ( removeEventConditionListReference )
    removeECLReference              [19] IMPLICIT RemoveEventConditionListReference-Request,
ENDIF
IF ( getEventConditionListAttributes )
    getECLAttributes                [20] IMPLICIT GetEventConditionListAttributes-Request,
ENDIF
IF ( reportEventConditionListStatus )
    reportECLStatus                 [21] IMPLICIT ReportEventConditionListStatus-Request,
ENDIF
IF ( alterEventConditionListMonitoring )
    alterECLMonitoring              [22] IMPLICIT AlterEventConditionListMonitoring-Request
ENDIF
ENDIF
    }
```

7.1.3 Request-Detail

```
Request-Detail ::= CHOICE {
                        -- this choice shall be selected if the tag value of the
                        -- ConfirmedServiceRequest does not match any of the tags below
    otherRequests               NULL,
IF ( createProgramInvocation )
    createProgramInvocation     [38] IMPLICIT CS-CreateProgramInvocation-Request,
ENDIF
IF ( start )
    start                       [40] IMPLICIT CS-Start-Request,
ENDIF
IF ( resume )
    resume                      [42] IMPLICIT CS-Resume-Request,
ENDIF
IF ( defineEventCondition )
    defineEventCondition        [47] IMPLICIT CS-DefineEventCondition-Request,
ENDIF
IF ( alterEventConditionMonitoring )
    alterEventConditionMonitoring   [51] IMPLICIT CS-AlterEventConditionMonitoring-Request,
ENDIF
IF ( defineEventEnrollment )
    defineEventEnrollment       [57] IMPLICIT CS-DefineEventEnrollment-Request,
ENDIF
IF ( alterEventEnrollment )
    alterEventEnrollment        [59] IMPLICIT CS-AlterEventEnrollment-Request
ENDIF
    }
```

7.2 The Unconfirmed-PDU

```
Unconfirmed-PDU ::= SEQUENCE {
    service         UnconfirmedService,
    ...,
IF ( cspi )
    service-ext     [79] Unconfirmed-Detail OPTIONAL
                        -- shall not be transmitted if value is the value
                        -- of a tagged type derived from NULL
ENDIF
    }
```

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

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

7.2.1 UnconfirmedService

```
 UnconfirmedService ::= CHOICE {
IF (informationReport )
      informationReport                  [0] IMPLICIT InformationReport,
ENDIF
IF ( unsolicitedStatus )
      unsolicitedStatus                  [1] IMPLICIT UnsolicitedStatus,
ENDIF
IF ( eventNotification )
      eventNotification                  [2] IMPLICIT EventNotification
ENDIF
      }
```

The UnconfirmedService type 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.

7.2.2 Unconfirmed-Detail

```
 Unconfirmed-Detail ::= CHOICE {
            -- this choice shall be selected if the tag value of the
            -- UnconfirmedService does not match any of the tags below
      otherRequests                      NULL,
IF ( cspi )
      eventNotification                  [2] IMPLICIT CS-EventNotification
ENDIF
      }
```

7.3 The Confirmed-ResponsePDU

```
 Confirmed-ResponsePDU ::= SEQUENCE {
      invokeID                 Unsigned32,
      service                  ConfirmedServiceResponse,
      ...,
IF ( csr cspi )
      service-ext              [79] Response-Detail OPTIONAL
                               -- shall not be transmitted if value is the value
                               -- of a tagged type derived from NULL
ENDIF
      }
```

The Confirmed-ResponsePDU shall be a sequence containing three elements, an unsigned integer, a ConfirmedServiceResponse and a 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 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 7.3.1.

7.3.1 ConfirmedServiceResponse

```
ConfirmedServiceResponse ::= CHOICE {
IF ( status )
    status                                  [0] IMPLICIT Status-Response,
ENDIF
IF ( getNameList )
    getNameList                             [1] IMPLICIT GetNameList-Response,
ENDIF
IF ( identify )
    identify                                [2] IMPLICIT Identify-Response,
ENDIF
IF ( rename )
    rename                                  [3] IMPLICIT Rename-Response,
ENDIF
IF ( read )
    read                                    [4] IMPLICIT Read-Response,
ENDIF
IF ( write )
    write                                   [5] IMPLICIT Write-Response,
ENDIF
IF ( vnam vadr )
IF ( getVariableAccessAttributes )
    getVariableAccessAttributes             [6] IMPLICIT GetVariableAccessAttributes-Response,
ENDIF
ENDIF
IF ( vnam )
IF ( vadr )
IF ( defineNamedVariable )
    defineNamedVariable                     [7] IMPLICIT DefineNamedVariable-Response,
ENDIF
ENDIF
ENDIF
IF ( vsca )
    --  choice [8] is reserved for a service defined in Annex E
IF ( defineScatteredAccess )
    defineScatteredAccess                   [8] IMPLICIT DefineScatteredAccess-Response
ENDIF
    --  choice [9] is reserved for a service defined in Annex E
IF ( getScatteredAccessAttributes )
    getScatteredAccessAttributes            [9] IMPLICIT GetScatteredAccessAttributes-Response
ENDIF
ENDIF
IF ( vnam )
IF ( deleteVariableAccess )
    deleteVariableAccess                    [10] IMPLICIT DeleteVariableAccess-Response,
ENDIF
ENDIF
IF ( vlis )
IF ( vnam )
IF ( defineNamedVariableList )
    defineNamedVariableList                 [11] IMPLICIT DefineNamedVariableList-Response,
ENDIF
IF ( getNamedVariableListAttributes )
    getNamedVariableListAttributes          [12] IMPLICIT GetNamedVariableListAttributes-Response,
ENDIF
IF ( deleteNamedVariableList )
    deleteNamedVariableList                 [13] IMPLICIT DeleteNamedVariableList-Response,
ENDIF
ENDIF
ENDIF
IF ( vnam )
```

```
IF ( defineNamedType )
    defineNamedType                 [14] IMPLICIT DefineNamedType-Response,
ENDIF
IF ( getNamedTypeAttributes )
    getNamedTypeAttributes          [15] IMPLICIT GetNamedTypeAttributes-Response,
ENDIF
IF ( deleteNamedType )
    deleteNamedType                 [16] IMPLICIT DeleteNamedType-Response,
ENDIF
ENDIF
IF ( input )
    input                           [17] IMPLICIT Input-Response,
ENDIF
IF ( output )
    output                          [18] IMPLICIT Output-Response,
ENDIF
IF ( takeControl )
    takeControl                     [19] TakeControl-Response,
ENDIF
IF ( relinquishControl )
    relinquishControl               [20] IMPLICIT RelinquishControl-Response,
ENDIF
IF ( defineSemaphore )
    defineSemaphore                 [21] IMPLICIT DefineSemaphore-Response,
ENDIF
IF ( deleteSemaphore )
    deleteSemaphore                 [22] IMPLICIT DeleteSemaphore-Response,
ENDIF
IF ( reportSemaphoreStatus )
    reportSemaphoreStatus           [23] IMPLICIT ReportSemaphoreStatus-Response,
ENDIF
IF ( reportPoolSemaphoreStatus )
    reportPoolSemaphoreStatus       [24] IMPLICIT ReportPoolSemaphoreStatus-Response,
ENDIF
IF ( reportSemaphoreEntryStatus )
    reportSemaphoreEntryStatus      [25] IMPLICIT ReportSemaphoreEntryStatus-Response,
ENDIF
IF ( initiateDownloadSequence )
    initiateDownloadSequence        [26] IMPLICIT InitiateDownloadSequence-Response,
    downloadSegment                 [27] IMPLICIT DownloadSegment-Response,
    terminateDownloadSequence       [28] IMPLICIT TerminateDownloadSequence-Response,
ENDIF
IF ( initiateUploadSequence )
    initiateUploadSequence          [29] IMPLICIT InitiateUploadSequence-Response,
    uploadSegment                   [30] IMPLICIT UploadSegment-Response,
    terminateUploadSequence         [31] IMPLICIT TerminateUploadSequence-Response,
ENDIF
IF ( requestDomainDownload )
    requestDomainDownload           [32] IMPLICIT RequestDomainDownload-Response,
ENDIF
IF ( requestDomainUpload )
    requestDomainUpload             [33] IMPLICIT RequestDomainUpload-Response,
ENDIF
IF ( loadDomainContent )
    loadDomainContent               [34] IMPLICIT LoadDomainContent-Response,
ENDIF
IF ( storeDomainContent )
    storeDomainContent              [35] IMPLICIT StoreDomainContent-Response,
ENDIF
IF ( deleteDomain )
    deleteDomain                    [36] IMPLICIT DeleteDomain-Response,
ENDIF
IF ( getDomainAttributes )
```



```
    getDomainAttributes                 [37] IMPLICIT GetDomainAttributes-Response,
ENDIF
IF ( createProgramInvocation )
    createProgramInvocation             [38] IMPLICIT CreateProgramInvocation-Response,
ENDIF
IF ( deleteProgramInvocation )
    deleteProgramInvocation             [39] IMPLICIT DeleteProgramInvocation-Response,
ENDIF
IF ( start )
    start                               [40] IMPLICIT Start-Response,
ENDIF
IF ( stop )
    stop                                [41] IMPLICIT Stop-Response,
ENDIF
IF ( resume )
    resume                              [42] IMPLICIT Resume-Response,
ENDIF
IF ( reset )
    reset                               [43] IMPLICIT Reset-Response,
ENDIF
IF ( kill )
    kill                                [44] IMPLICIT Kill-Response,
ENDIF
IF ( getProgramInvocationAttributes )
    getProgramInvocationAttributes      [45] IMPLICIT  GetProgramInvocationAttributes-Response,
ENDIF
IF ( obtainFile )
    obtainFile                          [46] IMPLICIT ObtainFile-Response,
ENDIF
IF ( defineEventCondition )
    defineEventCondition                [47] IMPLICIT DefineEventCondition-Response,
ENDIF
IF ( deleteEventCondition )
    deleteEventCondition                [48] IMPLICIT DeleteEventCondition-Response,
ENDIF
IF ( getEventConditionAttributes )
    getEventConditionAttributes         [49] IMPLICIT GetEventConditionAttributes-Response,
ENDIF
IF ( reportEventConditionStatus )
    reportEventConditionStatus          [50] IMPLICIT ReportEventConditionStatus-Response,
ENDIF
IF ( alterEventConditionMonitoring )
    alterEventConditionMonitoring       [51] IMPLICIT AlterEventConditionMonitoring-Response,
ENDIF
IF ( triggerEvent )
    triggerEvent                        [52] IMPLICIT TriggerEvent-Response,
ENDIF
IF ( defineEventAction )
    defineEventAction                   [53] IMPLICIT DefineEventAction-Response,
ENDIF
IF ( deleteEventAction )
    deleteEventAction                   [54] IMPLICIT DeleteEventAction-Response,
ENDIF
IF ( getEventActionAttributes )
    getEventActionAttributes            [55] IMPLICIT GetEventActionAttributes-Response,
ENDIF
IF ( reportEventActionStatus )
    reportEventActionStatus             [56] IMPLICIT ReportEventActionStatus-Response,
ENDIF
IF ( defineEventEnrollment )
    defineEventEnrollment               [57] IMPLICIT DefineEventEnrollment-Response,
ENDIF
IF ( deleteEventEnrollment )
```

```
    deleteEventEnrollment                [58] IMPLICIT DeleteEventEnrollment-Response,
ENDIF
IF ( alterEventEnrollment )
    alterEventEnrollment                 [59] IMPLICIT AlterEventEnrollment-Response,
ENDIF
IF ( reportEventEnrollmentStatus )
    reportEventEnrollmentStatus          [60] IMPLICIT ReportEventEnrollmentStatus-Response,
ENDIF
IF ( getEventEnrollmentAttributes )
    getEventEnrollmentAttributes         [61] IMPLICIT GetEventEnrollmentAttributes-Response,
ENDIF
IF ( acknowledgeEventNotification )
    acknowledgeEventNotification         [62] IMPLICIT AcknowledgeEventNotification-Response,
ENDIF
IF ( getAlarmSummary )
    getAlarmSummary                      [63] IMPLICIT GetAlarmSummary-Response,
ENDIF
IF ( getAlarmEnrollmentSummary )
    getAlarmEnrollmentSummary            [64] IMPLICIT GetAlarmEnrollmentSummary-Response,
ENDIF
IF ( readJournal )
    readJournal                          [65] IMPLICIT ReadJournal-Response,
ENDIF
IF ( writeJournal )
    writeJournal                         [66] IMPLICIT WriteJournal-Response,
ENDIF
IF ( initializeJournal )
    initializeJournal                    [67] IMPLICIT InitializeJournal-Response,
ENDIF
IF ( reportJournalStatus )
    reportJournalStatus                  [68] IMPLICIT ReportJournalStatus-Response,
ENDIF
IF ( createJournal )
    createJournal                        [69] IMPLICIT CreateJournal-Response,
ENDIF
IF ( deleteJournal )
    deleteJournal                        [70] IMPLICIT DeleteJournal-Response,
ENDIF
IF ( getCapabilityList )
    getCapabilityList                    [71] IMPLICIT GetCapabilityList-Response,
ENDIF
        -- choices [72] through [77] are reserved for use by services
        -- defined in annex D
IF ( fileOpen )
    fileOpen                             [72] IMPLICIT FileOpen-Response,
ENDIF
IF ( fileRead )
    fileRead                             [73] IMPLICIT FileRead-Response,
ENDIF
IF ( fileClose )
    fileClose                            [74] IMPLICIT FileClose-Response,
ENDIF
IF ( fileRename )
    fileRename                           [75] IMPLICIT FileRename-Response,
ENDIF
IF ( fileDelete )
    fileDelete                           [76] IMPLICIT FileDelete-Response,
ENDIF
IF ( fileDirectory )
    fileDirectory                        [77] IMPLICIT FileDirectory-Response,
ENDIF
IF ( csr cspi )
    additionalService                    [78] AdditionalService-Response,
```



```
ENDIF
        -- choice [79] is reserved
IF ( getDataExchangeAttributes )
    getDataExchangeAttributes          [80] GetDataExchangeAttributes-Response,
                    -- Shall not appear in minor version 1
ENDIF
IF ( exchangeData )
    exchangeData                       [81] IMPLICIT ExchangeData-Response,
                    -- Shall not appear in minor version 1
ENDIF
IF ( defineAccessControlList )
    defineAccessControlList            [82] IMPLICIT DefineAccessControlList-Response,
                    -- Shall not appear in minor version 1 or 2
ENDIF
IF ( getAccessControlListAttributes )
    getAccessControlListAttributes     [83] IMPLICIT GetAccessControlListAttributes-Response,
                    -- Shall not appear in minor version 1 or 2
ENDIF
IF ( reportAccessControlledObjects )
    reportAccessControlledObjects      [84] IMPLICIT ReportAccessControlledObjects-Response,
                    -- Shall not appear in minor version 1 or 2
ENDIF
IF ( deleteAccessControlList )
    deleteAccessControlList            [85] IMPLICIT DeleteAccessControlList-Response,
                    -- Shall not appear in minor version 1 or 2
ENDIF
IF ( changeAccessControl )
    changeAccessControl                [86] IMPLICIT ChangeAccessControl-Response,
                    -- Shall not appear in minor version 1 or 2
ENDIF
    ...
    }
```

The ConfirmedServiceResponse type 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.

7.3.2 AdditionalService-Response

```
 AdditionalService-Response ::= CHOICE {
IF ( csr )
IF ( vMDStop )
    vMDStop                            [0] IMPLICIT VMDStop-Response,
ENDIF
IF ( vMDReset )
    vMDReset                           [1] IMPLICIT VMDReset-Response,
ENDIF
IF ( select )
    select                             [2] IMPLICIT Select-Response,
ENDIF
IF ( alterProgramInvocationAttributes )
    alterPI                            [3] IMPLICIT AlterProgramInvocationAttributes-Response,
ENDIF
ENDIF
IF ( cspi )
IF ( initiateUnitControlLoad )
    initiateUCLoad                     [4] IMPLICIT InitiateUnitControlLoad-Response,
ENDIF
IF ( unitControlLoadSegment )
    uCLoad                             [5] IMPLICIT UnitControlLoadSegment-Response,
ENDIF
IF ( unitControlUpload )
    uCUpload                           [6] IMPLICIT UnitControlUpload-Response,
```

```
ENDIF
IF ( startUnitControl )
    startUC                         [7] IMPLICIT StartUnitControl-Response,
ENDIF
IF ( stopUnitControl )
    stopUC                          [8] IMPLICIT StopUnitControl-Response,
ENDIF
IF ( createUnitControl )
    createUC                        [9] IMPLICIT CreateUnitControl-Response,
ENDIF
IF ( addToUnitControl )
    addToUC                         [10] IMPLICIT AddToUnitControl-Response,
ENDIF
IF ( removeFromUnitControl )
    removeFromUC                    [11] IMPLICIT RemoveFromUnitControl-Response,
ENDIF
IF ( getUnitControlAttributes )
    getUCAttributes                 [12] IMPLICIT GetUnitControlAttributes-Response,
ENDIF
IF ( loadUnitControlFromFile )
    loadUCFromFile                  [13] IMPLICIT LoadUnitControlFromFile-Response,
ENDIF
IF ( storeUnitControlToFile )
    storeUCToFile                   [14] IMPLICIT StoreUnitControlToFile-Response,
ENDIF
IF ( deleteUnitControl )
    deleteUC                        [15] IMPLICIT DeleteUnitControl-Response,
ENDIF
IF ( defineEventConditionList )
    defineECL                       [16] IMPLICIT DefineEventConditionList-Response,
ENDIF
IF ( deleteEventConditionList )
    deleteECL                       [17] IMPLICIT DeleteEventConditionList-Response,
ENDIF
IF ( addEventConditionListReference )
    addECLReference                 [18] IMPLICIT AddEventConditionListReference-Response,
ENDIF
IF ( removeEventConditionListReference )
    removeECLReference              [19] IMPLICIT RemoveEventConditionListReference-Response,
ENDIF
IF ( getEventConditionListAttributes )
    getECLAttributes                [20] IMPLICIT GetEventConditionListAttributes-Response,
ENDIF
IF ( reportEventConditionListStatus )
    reportECLStatus                 [21] IMPLICIT ReportEventConditionListStatus-Response,
ENDIF
IF ( alterEventConditionListMonitoring )
    alterECLMonitoring              [22] IMPLICIT AlterEventConditionListMonitoring-Response
ENDIF
ENDIF
    }
```

7.3.3 Response-Detail

```
Response-Detail ::= CHOICE {
                        -- this choice shall be selected if the tag value of the
                        -- ConfirmedServiceResponse does not match any of the tags below
    otherRequests                   NULL,
IF ( status )
    status                          [0] IMPLICIT CS-Status-Response,
ENDIF
IF ( getProgramInvocationAttributes )
    getProgramInvocationAttributes  [45] IMPLICIT CS-GetProgramInvocationAttributes-Response,
```



```
ENDIF
IF ( defineEventCondition )
    defineEventCondition                [47] IMPLICIT CS-DefineEventCondition-Response,
ENDIF
IF ( getEventConditionAttributes )
    getEventConditionAttributes         [49] IMPLICIT CS-GetEventConditionAttributes-Response,
ENDIF
IF ( defineEventEnrollment )
    defineEventEnrollment               [57] IMPLICIT CS-DefineEventEnrollment-Response
ENDIF
    }
```

7.4 The Confirmed-ErrorPDU

```
Confirmed-ErrorPDU ::= SEQUENCE {
    invokeID                    [0] IMPLICIT Unsigned32,
IF ( attachToEventCondition attachToSemaphore )
    modifierPosition            [1] IMPLICIT Unsigned32 OPTIONAL,
ENDIF
    serviceError                [2] IMPLICIT ServiceError
    }
```

The Confirmed-ErrorPDU shall be a sequence containing three elements, an unsigned integer, an optional unsigned integer, 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 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 that were specified in the **listOfModifiers** 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 9506-1, clause 24).

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 7.4.1.

7.4.1 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
        },
IF ( cancel )
        cancel                              [10] IMPLICIT INTEGER {     -- CANCEL
            other                               (0),  -- OTHER
            invoke-id-unknown                   (1),  -- INVOKE-ID-UNKNOWN
            cancel-not-possible                 (2)   -- CANCEL-NOT-POSSIBLE
        },
ENDIF
IF ( fileOpen fileClose fileRead fileRename fileDelete fileDirectory obtainFile )
        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
        },
ENDIF
        others                                  [12] IMPLICIT INTEGER -- OTHERS
    } ,
        additionalCode              [1] IMPLICIT INTEGER OPTIONAL,
        additionalDescription       [2] IMPLICIT VisibleString OPTIONAL,
IF ( obtainFile start stop resume reset deleteVariableAccess deleteNamedVariableList deleteNamedType
defineEventEnrollment fileRename )
        serviceSpecificInfo         [3] CHOICE {
IF ( obtainFile )
            obtainFile                          [0] IMPLICIT ObtainFile-Error,
ENDIF
IF ( start )
            start                               [1] IMPLICIT Start-Error,
ENDIF
IF ( stop )
            stop                                [2] IMPLICIT Stop-Error,
ENDIF
IF ( resume )
            resume                              [3] IMPLICIT Resume-Error,
ENDIF
IF ( reset )
            reset                               [4] IMPLICIT Reset-Error,
ENDIF
IF ( deleteVariableAccess )
            deleteVariableAccess                [5] IMPLICIT DeleteVariableAccess-Error,
ENDIF
IF ( deleteNamedVariableList )
            deleteNamedVariableList             [6] IMPLICIT DeleteNamedVariableList-Error,
ENDIF
IF ( deleteNamedType )
            deleteNamedType                     [7] IMPLICIT DeleteNamedType-Error,
ENDIF
IF ( defineEventEnrollment )
            defineEventEnrollment-Error         [8] DefineEventEnrollment-Error,
ENDIF
            -- [9] Reserved for use by annex D
IF ( fileRename )
        fileRename                              [9] IMPLICIT FileRename-Error,
ENDIF
IF ( csr cspi )
            additionalService                   [10] AdditionalService-Error,
ENDIF
            changeAccessControl                 [11] IMPLICIT ChangeAccessControlError
            } OPTIONAL
ENDIF
        }
```

The ServiceError type 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 that 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 of subclause 5.5 in this part of ISO 9506, and the definitions in clause 24 of ISO 9506-1.

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, 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 if 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.

7.4.2 AdditionalService-Error

```
AdditionalService-Error ::= CHOICE {
IF ( defineEventConditionList )
    defineEcl                          [0] DefineEventConditionList-Error,
ENDIF
IF ( addEventConditionListReference )
    addECLReference                    [1] AddEventConditionListReference-Error,
ENDIF
IF ( removeEventConditionListReference )
    removeECLReference                 [2] RemoveEventConditionListReference-Error,
ENDIF
IF ( initiateUnitControlLoad )
    initiateUC                         [3] InitiateUnitControl-Error,
ENDIF
IF ( startUnitControl )
    startUC                            [4] IMPLICIT StartUnitControl-Error,
ENDIF
IF ( stopUnitControl )
    stopUC                             [5] IMPLICIT StopUnitControl-Error,
ENDIF
IF ( deleteUnitControl )
    deleteUC                           [6] IMPLICIT DeleteUnitControl-Error,
ENDIF
IF ( loadUnitControlFromFile )
    loadUCFromFile                     [7] IMPLICIT LoadUnitControlFromFile-Error
ENDIF
    }
```

7.5 Common MMS Types

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

7.5.1 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:

'0000ttttttttttttttttttttttttttttttddddddddddddddddd'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).

7.5.2 Identifiers and Integer Types

The types "Identifier", "Integer8", "Integer16", "Integer32", "Unsigned8", "Unsigned16", and "Unsigned32" are used throughout this part of ISO 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))

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 of characters defined by the VisibleString type as specified in ASN.1 (ISO/IEC 8824-1). Identifiers shall be case sensitive.

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

7.5.3 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 5.5 of this part of ISO 9506, using the definition of the ObjectName service parameter in clause 7 of ISO 9506-1.

7.5.4 ObjectClass

```
ObjectClass ::=   CHOICE {
    basicObjectClass        [0] IMPLICIT INTEGER {
IF ( vnam )
        namedVariable               (0),
ENDIF
                -- value 1 is reserved for definition in Annex E
If ( vsca )
        scatteredAccess             (1),
ENDIF
IF ( vlis )
        namedVariableList           (2),
ENDIF
IF ( vnam )
        namedType                   (3),
ENDIF
        semaphore                   (4),
        eventCondition              (5),
        eventAction                 (6),
        eventEnrollment             (7),
        journal                     (8),
        domain                      (9),
        programInvocation           (10),
        operatorStation             (11),
        dataExchange                (12),     -- Shall not appear in minor version 1
        accessControlList           (13)      -- Shall not appear in minor version 1 or 2
        },
    ...,
IF ( cspi )
    csObjectClass               [1] IMPLICIT INTEGER {
        eventConditionList          (0),
        unitControl                 (1) }
ENDIF
    }
```

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

7.5.5 ApplicationReference

The form of the ApplicationReference type depends on the communication system employed. For a definition appropriate to OSI communications, see annex A.

7.5.6 MMSString

The MMSString is used to store the user defined strings in character sets appropriate to their use. Its type is defined as follows:

```
MMSString ::= CHOICE {
    english         VisibleString,
    european        ISO10646String,
    general         CHARACTER STRING
    }

ISO10646String ::=
    BMPString (FROM Level1 INTERSECTION(BasicLatin UNION Latin-1Supplement))
```


The **english** choice provides the familiar 94 character set used in English. The **european** choice uses the BMP of ISO 10646 (usually a 16 bit encoding) and covers most of the character sets of western Europe. The **general** choice allows the full specification of any character set, and may be negotiated or agreed to before communication.

7.5.7 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 9506-1 and ISO 9506-2 on lengths and legal characters in a FileName shall be specified in the System Configuraion and Initialisation Statement (see clause 25). 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 9506-1 and ISO 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.

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:

Initiate	Cancel
Conclude	Reject
Abort	

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.

```
Initiate-RequestPDU ::= SEQUENCE {
    localDetailCalling                  [0] IMPLICIT Integer32 OPTIONAL,
    proposedMaxServOutstandingCalling   [1] IMPLICIT Integer16,
    proposedMaxServOutstandingCalled    [2] IMPLICIT Integer16,
    proposedDataStructureNestingLevel   [3] IMPLICIT Integer8 OPTIONAL,
    initRequestDetail                   [4] IMPLICIT SEQUENCE {
        proposedVersionNumber               [0] IMPLICIT Integer16,
        proposedParameterCBB                [1] IMPLICIT ParameterSupportOptions,
        servicesSupportedCalling            [2] IMPLICIT ServiceSupportOptions ,
        ...,
IF (csr cspi)
        additionalSupportedCalling          [3] IMPLICIT AdditionalSupportOptions,
ENDIF
IF (cspi)
        additionalCbbSupportedCalling       [4] IMPLICIT AdditionalCbbOptions,
        privilegeClassIdentityCalling       [5] IMPLICIT VisibleString
ENDIF
```

```
        }
    }

Initiate-ResponsePDU ::= SEQUENCE {
    localDetailCalled                       [0] IMPLICIT Integer32 OPTIONAL,
    negotiatedMaxServOutstandingCalling     [1] IMPLICIT Integer16,
    negotiatedMaxServOutstandingCalled      [2] IMPLICIT Integer16,
    negotiatedDataStructureNestingLevel     [3] IMPLICIT Integer8 OPTIONAL,
    initResponseDetail                      [4] IMPLICIT SEQUENCE {
        negotiatedVersionNumber                 [0] IMPLICIT Integer16,
        negotiatedParameterCBB                  [1] IMPLICIT ParameterSupportOptions,
        servicesSupportedCalled                 [2] IMPLICIT ServiceSupportOptions,
        ...,
IF (csr cspi)
        additionalSupportedCalled               [3] IMPLICIT AdditionalSupportOptions,
ENDIF
IF (cspi )
        additionalCbbSupportedCalled            [4] IMPLICIT AdditionalCbbOptions,
        privilegeClassIdentityCalled            [5] IMPLICIT VisibleString
ENDIF
        }
    }

Initiate-ErrorPDU ::= ServiceError
```

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 M-U-ABORT service (see clause 24).

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
            },
IF ( cancel )
        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
            },
ENDIF
        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 **rejectReason** field 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 Conditioned Service Response Protocol

9.1 Introduction

This clause describes the protocol required for the realization of the services that are defined in Clause 9 of ISO 9506-1. This includes:

DefineAccessControlList
GetAccessControlListAttributes
ReportAccessControlledObjects
DeleteAccessControlList
ChangeAccessControl

9.2 Access Condition

The abstract syntax of the Access Condition parameter is the AccessCondition type. The AccessCondition type is defined in 9.1.2 of ISO 9506-1.

9.3 DefineAccessControlList

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

```
DefineAccessControlList-Request ::= SEQUENCE {
    accessControlListName           [0] IMPLICIT Identifier,
    accessControlListElements       [1] IMPLICIT SEQUENCE {
        readAccessCondition             [0] AccessCondition OPTIONAL,
        storeAccessCondition            [1] AccessCondition OPTIONAL,
        writeAccessCondition            [2] AccessCondition OPTIONAL,
        loadAccessCondition             [3] AccessCondition OPTIONAL,
        executeAccessCondition          [4] AccessCondition OPTIONAL,
        deleteAccessCondition           [5] AccessCondition OPTIONAL,
        editAccessCondition             [6] AccessCondition OPTIONAL
        }
    }

DefineAccessControlList-Response ::= NULL
```

9.3.1 DefineAccessControlList-Request

The abstract syntax of the **defineAccessControlList-Request** choice of the ConfirmedServiceRequest type shall be the DefineAccessControlList-Request.

9.3.1.1 accessControlListElements

The **accessControlListElements** field shall be the List of Access Control Element parameter of the DefineAccessControlList.request primitive and shall appear as the List of Access Control Element parameter of the DefineAccessControlList.indication. This field shall contain zero or one occurrence of the AccessCondition type for each of th e seven possible values of the Service Class parameter. The AccessCondition associated with the Service Class equal to Read shall appear in the **readAccessCondition** field of the accessControlListElement; similarly for the other values of the Service Class parameter.

9.3.2 DefineAccessControlList-Response

The abstract syntax of the **defineAccessControlList-Response** choice of the ConfirmedServiceResponse type shall be the DefineAccessControlList-Response.

9.4 GetAccessControlListAttributes

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

```
GetAccessControlListAttributes-Request ::= CHOICE {
    accessControlListName                [0] IMPLICIT Identifier,
    vMD                                  [1] IMPLICIT NULL,
    namedObject                          [2] IMPLICIT SEQUENCE {
        objectClass                          [0] ObjectClass,
        objectName                           [1] ObjectName
        }
    }

GetAccessControlListAttributes-Response ::= SEQUENCE {
    name                                 [0] Identifier, -- Access Control List name
    accessControlListElements            [1] IMPLICIT SEQUENCE {
        readAccessCondition                  [0] AccessCondition OPTIONAL,
        storeAccessCondition                 [1] AccessCondition OPTIONAL,
        writeAccessCondition                 [2] AccessCondition OPTIONAL,
        loadAccessCondition                  [3] AccessCondition OPTIONAL,
        executeAccessCondition               [4] AccessCondition OPTIONAL,
        deleteAccessCondition                [5] AccessCondition OPTIONAL,
        editAccessCondition                  [6] AccessCondition OPTIONAL
        },
    vMDuse                               [2] IMPLICIT BOOLEAN,
    references                           [3] IMPLICIT SEQUENCE OF SEQUENCE {
        objectClass                          [0] ObjectClass,
        objectCount                          [1] IMPLICIT INTEGER
        },
    accessControlList                    [4] IMPLICIT Identifier OPTIONAL
                                 -- shall be included if and only if
                                 -- aco has been negotiated
    }
```

9.4.1 GetAccessControlListAttributes-Request

The abstract syntax of the **getAccessControlListAttributes-Request** choice of the ConfirmedServiceRequest type shall be the GetAccessControlListAttributes-Request.

9.4.2 GetAccessControlListAttributes-Response

The abstract syntax of the **getAccessControlListAttributes-Response** choice of the ConfirmedServiceResponse type shall be the GetAccessControlListAttributes-Response.

9.4.2.1 accessControlListElements

The **accessControlListElements** field shall be the List of Access Control Element parameter of the DefineAccessControlList.response primitive and shall appear as the List of Access Control Element parameter of the DefineAccessControlList.confirm. This field shall contain zero or one occurrence of the AccessCondition type for each of the seven possible values of the Service Class parameter. The AccessCondition associated with the Service Class equal to Read shall appear in the **readAccessCondition** field of the accessControlListElement; similarly for the other values of the Service Class parameter.

9.4.2.2 Counts of Controlled Objects

The abstract syntax of the Counts of Controlled Objects parameter shall be the **references** field of the GetAccessControlListAttributes-Response type. For each object class that contains one or more objects that reference this Access Control List object, a sequence of objectClass and objectCount shall be included. If there are no objects in a class that reference this Access Control List object (that is, the count is zero), this sequence shall not be included. If **aco** has not been negotiated, this field shall not appear.

9.4.2.3 Access Control List

The accessControlList parameter shall appear if and only if the **aco** CBB has been negotiated.

9.5 ReportAccessControlledObjects

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

```
ReportAccessControlledObjects-Request ::= SEQUENCE {
    accessControlList       [0] IMPLICIT Identifier,
    objectClass             [1] ObjectClass,
    continueAfter           [2] ObjectName OPTIONAL
    }

ReportAccessControlledObjects-Response ::= SEQUENCE {
    listOfNames             [0] IMPLICIT SEQUENCE OF ObjectName,
    moreFollows             [1] IMPLICIT BOOLEAN DEFAULT FALSE
    }
```

9.5.1 ReportAccessControlledObjects-Request

The abstract syntax of the **reportAccessControlledObjects-Request** choice of the ConfirmedServiceRequest type shall be the ReportAccessControlledObjects-Request.

9.5.2 ReportAccessControlledObjects-Response

The abstract syntax of the **reportAccessControlledObjects-Response** choice of the ConfirmedServiceResponse type shall be the ReportAccessControlledObjects-Response.

9.6 DeleteAccessControlList

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

```
DeleteAccessControlList-Request ::= Identifier  -- Name of Access Control List Object

DeleteAccessControlList-Response ::= NULL
```

9.6.1 DeleteAccessControlList-Request

The abstract syntax of the **deleteAccessControlList-Request** choice of the ConfirmedServiceRequest type shall be the DeleteAccessControlList-Request.

9.6.2 DeleteAccessControlList-Response

The abstract syntax of the **deleteAccessControlList-Response** choice of the ConfirmedServiceResponse type shall be the DeleteAccessControlList-Response.

9.7 ChangeAccessControl

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

```
ChangeAccessControl-Request ::= SEQUENCE {
    scopeOfChange               CHOICE {
        vMDOnly                     [0] IMPLICIT NULL,
        listOfObjects               [1] IMPLICIT SEQUENCE {
            objectClass                 [0] ObjectClass,
            objectScope                 [1] CHOICE {
                specific                    [0] IMPLICIT SEQUENCE OF ObjectName, -- SPECIFIC
                                -- Names of the objects (of class objectClass)
                                -- whose access is to be changed
                aa-specific                 [1] IMPLICIT NULL,          -- AA-SPECIFIC
                domain                      [2] IMPLICIT Identifier,    -- DOMAIN
                                -- Name of the Domain whose elements
                                -- are to be changed
                vmd                         [3] IMPLICIT NULL           -- VMD
                }
            }
        },
    accessControlListName           [2] IMPLICIT Identifier
                -- name of the AccessControlList Object that contains
                -- the conditions for access control
    }

ChangeAccessControl-Response ::= SEQUENCE {
    numberMatched           [0] IMPLICIT Unsigned32,
    numberChanged           [1] IMPLICIT Unsigned32
    }

ChangeAccessControl-Error ::= Unsigned32 -- numberChanged
```

9.7.1 ChangeAccessControl-Request

The abstract syntax of the **changeAccessControl-Request** choice of the ConfirmedServiceRequest type shall be the ChangeAccessControl-Request.

9.7.2 ChangeAccessControl-Response

The abstract syntax of the **changeAccessControl-Response** choice of the ConfirmedServiceResponse type shall be the ChangeAccessControl-Response.

9.7.3 ChangeAccessControl-Error

The abstract syntax of the **changeAccessControl** choice of the ServiceError type shall be the ChangeAccessControl-Error.

10 VMD Support Protocol

10.1 Introduction

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

Status	Rename
UnsolicitedStatus	GetCapabilityList
GetNameList	VMDStop
Identify	VMDReset

10.2 Status Response Parameter

The abstract syntax of the Status Response parameter is the StatusResponse type.

```
StatusResponse ::= 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
    }
```

10.2.1 CS-Status-Response

```
CS-Status-Response ::= CHOICE {
IF ( csr )
    SEQUENCE {
        operation State           [0] IMPLICIT OperationState,
        extended Status           [1] IMPLICIT ExtendedStatus,
        extendedStatusMask        [2] IMPLICIT ExtendedStatus DEFAULT '1111'B,
        selectedProgramInvocation CHOICE {
            programInvocation         [3] IMPLICIT Identifier,
            noneSelected              [4] IMPLICIT NULL       } },
ENDIF
IF ( cspi )
    NULL
ENDIF
    }
```

10.2.2 OperationState

```
OperationState ::= INTEGER {
    idle                      (0),
    loaded                    (1),
    ready                     (2),
    executing                 (3),
    motion-paused             (4),
    manualInterventionRequired (5) }
```

10.2.3 ExtendedStatus

```
ExtendedStatus ::= BITSTRING {
    safetyInterlocksViolated          (0),
    anyPhysicalResourcePowerOn        (1),
    allPhysicalResourcesCalibrated    (2),
    localControl                      (3) }
```

10.3 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. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

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

Status-Response ::= StatusResponse
```

10.3.1 Status-Request

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

10.3.2 Status-Response

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

10.4 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 ::= StatusResponse
```

10.4.1 UnsolicitedStatus

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

10.5 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 {
    objectClass             [0] ObjectClass,
    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 }
```


10.5.1 GetNameList-Request

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

10.5.1.1 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.

10.5.2 GetNameList-Response

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

10.6 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              CHOICE {
        english                 [0] IMPLICIT VisibleString,
        european                [4] IMPLICIT ISO10646String,
        general                 [5] IMPLICIT CHARACTER STRING },
        modelName           CHOICE {
        english                 [1] IMPLICIT VisibleString,
        european                [6] IMPLICIT ISO10646String,
        general                 [7] IMPLICIT CHARACTER STRING },
    revision                CHOICE {
        english                 [2] IMPLICIT VisibleString,
        european                [8] IMPLICIT ISO10646String,
        general                 [9] IMPLICIT CHARACTER STRING },
    listOfAbstractSyntaxes  [3] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL
    }
```

10.6.1 Identify-Request

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

10.6.2 Identify-Response

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

10.7 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 {
    objectClass                 [0] ObjectClass,
    currentName                 [1] ObjectName,
    newIdentifier               [2] IMPLICIT Identifier }

Rename-Response ::= NULL
```

10.7.1 Rename-Request

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

10.7.2 Rename-Response

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

10.8 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               MMSString OPTIONAL
    }

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

10.8.1 GetCapabilityList-Request

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

10.8.2 GetCapabilityList-Response

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

10.9 VMDStop

The abstract syntax of the **vMDStop** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the VMDStop-Request and VMDStop-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.

```
VMDStop-Request ::= NULL

VMDStop-Response ::= NULL
```

10.9.1 VMDStop-Request

The abstract syntax of the **vMDStop** choice of the ConfirmedServiceRequest shall be the VMDStop-Request.

10.9.2 VMDStop-Response

The abstract syntax of the **vMDStop** choice of the ConfirmedServiceResponse shall be the VMDStop-Response.

10.10 VMDReset

The abstract syntax of the **vMDReset** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the VMDReset-Request and VMDReset-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.

VMDReset-Request ::= BOOLEAN -- Extended Derivation

VMDReset-Response ::= StatusResponse

10.10.1 VMDReset-Request

The abstract syntax of the **vMDReset** choice of the ConfirmedServiceRequest shall be the VMDReset-Request.

10.10.2 VMDStop-Response

The abstract syntax of the **vMDReset** choice of the ConfirmedServiceResponse shall be the VMDReset-Response.

11 Domain Management Protocol

11.1 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

11.2 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. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

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

InitiateDownloadSequence-Response ::= NULL
```

11.2.1 InitiateDownloadSequence-Request

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

11.2.2 InitiateDownloadSequence-Response

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

11.3 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            LoadData,
    moreFollows         [1] IMPLICIT BOOLEAN DEFAULT TRUE }

LoadData ::= CHOICE {
    non-coded           [0] IMPLICIT OCTET STRING,
    coded               EXTERNAL,
    embedded            EMBEDDED PDV }
```

11.3.1 DownloadSegment-Request

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

11.3.2 DownloadSegment-Response

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

11.3.2.1 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, or an EXTERNAL or EMBEDDED PDV, indicating that the abstract syntax referenced by the EXTERNAL or EMBEDDED PDV contains coding rules for interpreting the value of this parameter.

11.4 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
```

11.4.1 TerminateDownloadSequence-Request

The abstract syntax of the **terminateDownloadSequence** choice of the ConfirmedServiceRequest shall be the TerminateDownloadSequence-Request.

11.4.1.1 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.

11.4.2 TerminateDownloadSequence-Response

The abstract syntax of the **terminateDownloadSequence** choice of the ConfirmedServiceResponse shall be the TerminateDownloadSequence-Response.

11.5 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 MMSString }
```

11.5.1 InitiateUploadSequence-Request

The abstract syntax of the **initiateUploadSequence** choice of the ConfirmedServiceRequest shall be the InitiateUploadSequence-Request.

11.5.2 InitiateUploadSequence-Response

The abstract syntax of the **initiateUploadSequence** choice of the ConfirmedServiceResponse shall be the InitiateUploadSequence-Response.

11.6 UploadSegment

The abstract syntax of the **uploadSegment** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified 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                LoadData,
    moreFollows             [1] IMPLICIT BOOLEAN DEFAULT TRUE }
```

11.6.1 UploadSegment-Request

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

11.6.2 UploadSegment-Response

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

11.6.2.1 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, or an EXTERNAL or EMBEDDED PDV, indicating that the abstract syntax referenced by the EXTERNAL or EMBEDDED PDV contains coding rules for interpreting the value of this parameter.

11.7 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
```

11.7.1 TerminateUploadSequence-Request

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

11.7.2 TerminateUploadSequence-Response

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

11.8 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 MMSString OPTIONAL,
    sharable                [2] IMPLICIT BOOLEAN,
    fileName                [4] IMPLICIT FileName }

RequestDomainDownload-Response ::= NULL
```

11.8.1 RequestDomainDownload-Request

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

If the List Of Capabilities parameter is present in the service request, and the parameter specifies an empty list, a SEQUENCE OF with zero elements shall be transmitted; if the parameter is not present in the service request, this field shall not be transmitted.

11.8.2 RequestDomainDownload-Response

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

11.9 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
```

11.9.1 RequestDomainUpload-Request

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

11.9.2 RequestDomainUpload-Response

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

11.10 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 MMSString OPTIONAL,
    sharable                [2] IMPLICIT BOOLEAN,
    fileName                [4] IMPLICIT FileName,
    thirdParty              [5] IMPLICIT ApplicationReference OPTIONAL       -- TPY
    }

LoadDomainContent-Response ::= NULL
```

11.10.1 LoadDomainContent-Request

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

If the List Of Capabilities parameter is present in the service request, and the parameter specifies an empty list, a SEQUENCE OF with zero elements shall be transmitted; if the parameter is not present in the service request, this field shall not be transmitted.

11.10.1.1 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.

11.10.2 LoadDomainContent-Response

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

11.11 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
```

11.11.1 StoreDomainContent-Request

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

11.11.1.1 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.

11.11.2 StoreDomainContent-Response

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

11.12 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
```

11.12.1 DeleteDomain-Request

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

11.12.2 DeleteDomain-Response

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

11.13 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 MMSString,
    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,
    accessControlList           [6] IMPLICIT Identifier OPTIONAL
                            -- Shall not appear in minor version one or two
    }
```

11.13.1 GetDomainAttributes-Request

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

11.13.2 GetDomainAttributes-Response

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

11.13.2.1 state

The DomainState type is defined in clause 11 of ISO 9506-1.

11.13.2.2 accessControlList

The accessControlList field shall appear if and only if the **aco** CBB has been negotiated.

12 Program Invocation Management Protocol

12.1 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
Select
AlterProgramInvocationAttributes
ReconfigureProgramInvocation

12.2 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

CS-CreateProgramInvocation-Request ::= INTEGER {
    normal          (0),
    controlling     (1),
    controlled      (2)
    }
```

12.2.1 CreateProgramInvocation-Request

The abstract syntax of the **createProgramInvocation** choice of the ConfirmedServiceRequest shall be the CreateProgramInvocation-Request.

12.2.1.1 Monitor

The abstract syntax of the Monitor parameter of the CreateProgramInvocation service shall be inferred from the presence or absence of the monitorType field. If the monitorType field is present, it shall indicate that the Monitor parameter is true. The value of the monitorType field 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 monitorType field is not present, the Monitor parameter is false and no monitoring is required.

12.2.2 CreateProgramInvocation-Response

The abstract syntax of the **createProgramInvocation** choice of the ConfirmedServiceResponse shall be the CreateProgramInvocation-Response.

12.2.3 CS-CreateProgramInvocation-Request

The abstract syntax of the **createProgramInvocation** choice of the Request-Detail shall be the CS-CreateProgramInvocation-Request.

12.3 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
```

12.3.1 DeleteProgramInvocation-Request

The abstract syntax of the **deleteProgramInvocation** choice of the ConfirmedServiceRequest shall be the DeleteProgramInvocation-Request.

12.3.2 DeleteProgramInvocation-Response

The abstract syntax of the **deleteProgramInvocation** choice of the ConfirmedServiceResponse shall be the DeleteProgramInvocation-Response.

12.4 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 MMSString,
        encodedString               EXTERNAL,
        embeddedString              EMBEDDED PDV  } OPTIONAL }

Start-Response ::= NULL

Start-Error ::= ProgramInvocationState

CS-Start-Request ::= [0] CHOICE {
    normal          NULL,
    controlling     SEQUENCE {
        startLocation       [0] IMPLICIT VisibleString OPTIONAL,
        startCount          [1] StartCount DEFAULT cycleCount 1
        }     }

StartCount ::= CHOICE {
    noLimit         [0] IMPLICIT NULL,
    cycleCount      [1] IMPLICIT INTEGER,
    stepCount       [2] IMPLICIT INTEGER   }
```

12.4.1 Start-Request

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

12.4.1.1 Execution Argument

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

12.4.2 Start-Response

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

12.4.3 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.

12.4.3.1 programInvocationState

The abstract syntax of the **programInvocationState** field is defined in clause 12 of ISO 9506-1.

12.4.4 CS-Start-Request

The abstract syntax of the **start** choice of the Request-Detail shall be the CS-Start-Request.

12.5 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
```

12.5.1 Stop-Request

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

12.5.2 Stop-Response

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

12.5.3 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.

12.6 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 MMSString,
        encodedString               EXTERNAL,
        enmbeddedString             EMBEDDED PDV  } OPTIONAL
    }

Resume-Response ::= NULL

Resume-Error ::= ProgramInvocationState

CS-Resume-Request ::= [0] CHOICE {
    normal              NULL,
    controlling         SEQUENCE {
                            CHOICE {
            continueMode        [0] IMPLICIT NULL,
```



```
changeMode              [1] StartCount
}    }    }
```

12.6.1 Resume-Request

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

12.6.1.1 Execution Argument

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

12.6.2 Resume-Response

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

12.6.3 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.

12.6.4 CS-Resume-Request

The abstract syntax of the **resume** choice of the Request-Detail shall be the CS-Resume-Request.

12.7 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
```

12.7.1 Reset-Request

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

12.7.2 Reset-Response

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

12.7.3 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.

12.8 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
```

12.8.1 Kill-Request

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

12.8.2 Kill-Response

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

12.9 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 MMSString,
        encodedString           EXTERNAL,
        enmbeddedString         EMBEDDED PDV     },
    accessControlList       [6] IMPLICIT Identifier OPTIONAL
                -- Shall not appear in minor version one or two
    }

CS-GetProgramInvocationAttributes-Response ::= SEQUENCE {
    errorCode               [0] IMPLICIT INTEGER,
    control                 [1] CHOICE {
        controlling             [0] IMPLICIT SEQUENCE {
            controlledPI            [0] IMPLICIT SEQUENCE OF Identifier,
            programLocation         [1] IMPLICIT VisibleString OPTIONAL,
            runningMode             [2] CHOICE {
                freeRunning             [0] IMPLICIT NULL,
                cycleLimited            [1] IMPLICIT INTEGER,
                stepLimited             [2] IMPLICIT INTEGER  }
            },
        controlled              [1] CHOICE {
            controllingPI           [0] IMPLICIT Identifier, -- Reference to Controlling
            none                    [1] IMPLICIT NULL -- uncontrolled
        },
        normal                  [2] IMPLICIT NULL  }     }
```


12.9.1 GetProgramInvocationAttributes-Request

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

12.9.2 GetProgramInvocationAttributes-Response

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

12.9.2.1 Execution Argument

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

12.9.2.2 Access Control List

The accessControlList parameter shall appear if and only if the **aco** CBB has been negotiated.

12.9.3 CS-GetProgramInvocationAttributes-Response

The abstract syntax of the **getProgramInvocationAttributes** choice of the Response-Detail shall be the CS-GetProgramInvocationAttributes-Response.

12.10 Select

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

```
Select-Request ::= SEQUENCE {
    controlling         [0] IMPLICIT Identifier OPTIONAL,
    controlled          [1] IMPLICIT SEQUENCE OF Identifier OPTIONAL
        -- this field shall appear if and only if the controlling field is included
    }

Select-Response ::= NULL
```

12.10.1 Select-Request

The abstract syntax of the **select** choice of the ConfirmedServiceRequest shall be the Select-Request.

12.10.2 Select-Response

The abstract syntax of the **select** choice of the ConfirmedServiceResponse shall be the Select-Response.

12.11 AlterProgramInvocationAttributes

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

```
AlterProgramInvocationAttributes-Request ::= SEQUENCE {
    programInvocation       [0] IMPLICIT Identifier,
    startCount              [1] IMPLICIT StartCount DEFAULT cycleCount 1 }

AlterProgramInvocationAttributes-Response ::= NULL
```

12.11.1 AlterProgramInvocationAttributes-Request

The abstract syntax of the **alterProgramInvocationAttributes** choice of the ConfirmedServiceRequest shall be the AlterProgramInvocationAttributes-Request.

12.11.2 AlterProgramInvocationAttributes-Response

The abstract syntax of the **alterProgramInvocationAttributes** choice of the ConfirmedServiceResponse shall be the AlterProgramInvocationAttributes-Response.

12.12 ReconfigureProgramInvocation

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

```
ReconfigureProgramInvocation-Request ::= SEQUENCE {
    oldProgramInvocationName    [0] IMPLICIT Identifier,
    newProgramInvocationName    [1] IMPLICIT Identifier OPTIONAL,
    domainsToAdd                [2] IMPLICIT SEQUENCE OF Identifier,
    domainsToRemove             [3] IMPLICIT SEQUENCE OF Identifier }

ReconfigureProgramInvocation-Response ::= NULL
```

12.12.1 ReconfigureProgramInvocation-Request

The abstract syntax of the **reconfigureProgramInvocation** choice of the ConfirmedServiceRequest shall be the ReconfigureProgramInvocation-Request.

12.12.2 ReconfigureProgramInvocation-Response

The abstract syntax of the **reconfigureProgramInvocation** choice of the ConfirmedServiceResponse shall be the ReconfigureProgramInvocation-Response.

13 Unit Control Protocol

13.1 Introduction

This clause describes the protocol required for realization of the services that are defined in clause 13 of ISO 9506-1. This includes:

InitiateUnitControlLoad
UnitControlLoadSegment
UnitControlUpload
StartUnitControl
StopUnitControl
CreateUnitControl
AddToUnitControl
RemoveFromUnitControl
GetUnitControlAttributes
LoadUnitControlFromFile
StoreUnitControlToFile
DeleteUnitControl

13.2 Control Element

The Control Element is a complex parameter used in several services to describe a single element of a Unit Control object.

```
ControlElement ::= CHOICE {
    beginDomainDef          [0] SEQUENCE {
        domainName              [1] IMPLICIT Identifier,
        capabilities            [2] IMPLICIT SEQUENCE OF MMSString,
        sharable                [3] IMPLICIT BOOLEAN,
        loadData                [4] LoadData OPTIONAL
        },
    continueDomainDef       [1] SEQUENCE {
        domainName              [1] IMPLICIT Identifier,
        loadData                [3] LoadData
        },
    endDomainDef            [2] IMPLICIT Identifier,
    piDefinition            [3] IMPLICIT SEQUENCE {
        piName                  [0] IMPLICIT Identifier,
        listOfDomains           [1] IMPLICIT SEQUENCE OF Identifier,
        reusable                [2] IMPLICIT BOOLEAN DEFAULT TRUE,
        monitorType             [3] IMPLICIT BOOLEAN OPTIONAL,
        pIState                 [4] IMPLICIT ProgramInvocationState OPTIONAL
        }   }
```

13.2.1 Monitor

The abstract syntax of the Monitor parameter of the control element parameter shall be inferred from the presence or absence of the monitorType field. If the monitorType field is present, it shall indicate that the Monitor parameter is true. The value of the monitorType field shall indicate the value of the Monitor Type parameter of the service request as specified in 11.2.1.1.4 of ISO 9506-1.

13.3 InitiateUnitControlLoad service

The abstract syntax of the **initiateUCLoad** choice of the AdditionalService-Request and the AdditionalService-Response is specified by the InitiateUnitControlLoad-Request and the InitiateUnitControlLoad-Response respectively. The abstract syntax of the **initiateUCLoad** choice of the AdditionalService-Error is specified by the InitiateUnitControl-Error. These types are specified below and described in the paragraphs that follow. Subclause 5.5 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
InitiateUnitControlLoad-Request ::= Identifier -- Unit Control Name

InitiateUnitControlLoad-Response ::= NULL

InitiateUnitControl-Error ::= CHOICE {
    domain                  [0] IMPLICIT Identifier,
    programInvocation       [1] IMPLICIT Identifier
    }
```

13.3.1 InitiateUnitControlLoad-Request

The abstract syntax of the **initiateUCLoad** choice of the AdditionalService-Request type shall be the InitiateUnitControlLoad-Request.

13.3.2 InitiateUnitControlLoad-Response

The abstract syntax of the **initiateUCLoad** choice of the AdditionalService-Response type shall be the InitiateUnitControlLoad-Response.

13.3.3 InitiateUnitControlLoad-Error

The abstract syntax of the **initiateUCLoad** choice of the AdditionalService-Error type shall be the InitiateUnitControlLoad-Error.

13.4 UnitControlLoadSegment service

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

```
UnitControlLoadSegment-Request ::= Identifier -- Unit Control Name

UnitControlLoadSegment-Response ::= SEQUENCE {
    controlElements     [0] IMPLICIT SEQUENCE OF ControlElement,
    moreFollows         [1] IMPLICIT BOOLEAN DEFAULT TRUE       }
```

13.4.1 UnitControlLoadSegment-Request

The abstract syntax of the **uCLoad** choice of the AdditionalService-Request type shall be the UnitControlLoadSegment-Request.

13.4.2 UnitControlLoadSegment-Response

The abstract syntax of the **uCLoad** choice of the AdditionalService-Response type shall be the UnitControlLoadSegment-Response.

13.5 UnitControlUpload service

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

```
UnitControlUpload-Request ::= SEQUENCE {
    unitControlName     [0] IMPLICIT Identifier, -- Unit Control Name
    continueAfter       CHOICE {
        domain              [1] IMPLICIT Identifier,
        ulsmID              [2] IMPLICIT INTEGER,
        programInvocation   [3] IMPLICIT Identifier    } OPTIONAL
    }

UnitControlUpload-Response ::= SEQUENCE {
    controlElements     [0] IMPLICIT SEQUENCE OF ControlElement,
    nextElement         CHOICE {
        domain              [1] IMPLICIT Identifier,
        ulsmID              [2] IMPLICIT INTEGER,
        programInvocation   [3] IMPLICIT Identifier    } OPTIONAL
    }
```

13.5.1 UnitControlUpload-Request

The abstract syntax of the **uCUpload** choice of the AdditionalService-Request type shall be the UnitControlUpload-Request.

13.5.2 UnitControlUpload-Response

The abstract syntax of the **uCUpload** choice of the AdditionalService-Response type shall be the UnitControlUpload-Response.

13.6 StartUnitControl service

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

```
StartUnitControl-Request ::= SEQUENCE {
    unitControlName          [0] IMPLICIT Identifier, -- Unit Control Name
    executionArgument        CHOICE {
        simpleString             [1] IMPLICIT MMSString,
        encodedString            EXTERNAL,
        enmbeddedString          EMBEDDED PDV } OPTIONAL
    }

StartUnitControl-Response ::= NULL

StartUnitControl-Error ::= SEQUENCE {
    programInvocationName    [0] IMPLICIT Identifier,
    programInvocationState   [1] IMPLICIT ProgramInvocationState   }
```

13.6.1 StartUnitControl-Request

The abstract syntax of the **startUC** choice of the AdditionalService-Request type shall be the StartUnitControl-Request.

13.6.2 StartUnitControl-Response

The abstract syntax of the **startUC** choice of the AdditionalService-Response type shall be the StartUnitControl-Response.

13.6.3 StartUnitControl-Error

The abstract syntax of the **startUC** choice of the AdditionalService-Error type shall be the StartUnitControl-Error, which shall be Program Invocation Name sub-parameter and the Program Invocation State sub-parameter, respectively, of the Result(-) parameter of the StartUnitControl.response primitive and shall appear as the Program Invocation Name sub-parameter and the Program Invocation State sub-parameter, respectively, of the StartUnitControl.confirm primitive, if issued.

13.7 StopUnitControl service

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

```
StopUnitControl-Request ::= Identifier -- Unit Control Name

StopUnitControl-Response ::= NULL

StopUnitControl-Error ::= SEQUENCE {
    programInvocationName    [0] IMPLICIT Identifier,
    programInvocationState   [1] IMPLICIT ProgramInvocationState   }
```

13.7.1 StopUnitControl-Request

The abstract syntax of the **stopUC** choice of the AdditionalService-Request type shall be the StopUnitControl-Request.

13.7.2 StopUnitControl-Response

The abstract syntax of the **stopUC** choice of the AdditionalService-Response type shall be the StopUnitControl-Response.

13.7.3 StopUnitControl-Error

The abstract syntax of the **stopUC** choice of the AdditionalService-Error shall be the StopUnitControl-Error, which shall be Program Invocation Name sub-parameter and the Program Invocation State sub-parameter, respectively, of the Result(-) parameter of the StopControUnit.response primitive and shall appear as the Program Invocation Name sub-parameter and the Program Invocation State sub-parameter, respectively, of the StopUnitControl.confirm primitive, if issued.

13.8 CreateUnitControl service

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

```
CreateUnitControl-Request ::= SEQUENCE {
    unitControl             [0] IMPLICIT Identifier, -- Unit Control Name
    domains                 [1] IMPLICIT SEQUENCE OF Identifier,
    programInvocations      [2] IMPLICIT SEQUENCE OF Identifier     }

CreateUnitControl-Response ::= NULL
```

13.8.1 CreateUnitControl-Request

The abstract syntax of the **createUC** choice of the AdditionalService-Request type shall be the CreateUnitControl-Request.

13.8.2 CreateUnitControl-Response

The abstract syntax of the **createUC** choice of the AdditionalService-Response type shall be the CreateUnitControl-Response.

13.9 AddToUnitControl service

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

```
AddToUnitControl-Request ::= SEQUENCE {
    unitControl             [0] IMPLICIT Identifier, -- Unit Control Name
    domains                 [1] IMPLICIT SEQUENCE OF Identifier,
    programInvocations      [2] IMPLICIT SEQUENCE OF Identifier }

AddToUnitControl-Response ::= NULL
```

13.9.1 AddToUnitControl-Request

The abstract syntax of the **addToUC** choice of the AdditionalService-Request type shall be the AddToUnitControl-Request.

13.9.2 AddToUnitControl-Response

The abstract syntax of the **addToUC** choice of the AdditionalService-Response type shall be the AddToUnitControl-Response.

13.10 RemoveFromUnitControl service

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


```
RemoveFromUnitControl-Request ::= SEQUENCE {
    unitControl             [0] IMPLICIT Identifier, -- Unit Control Name
    domains                 [1] IMPLICIT SEQUENCE OF Identifier,
    programInvocations      [2] IMPLICIT SEQUENCE OF Identifier }

RemoveFromUnitControl-Response ::= NULL
```

13.10.1 RemoveFromUnitControl-Request

The abstract syntax of the **removeFromUC** choice of the AdditionalService-Request type shall be the RemoveFromUnitControl-Request.

13.10.2 RemoveFromUnitControl-Response

The abstract syntax of the **removeFromUC** choice of the AdditionalService-Response type shall be the RemoveFromUnitControl-Response.

13.11 GetUnitControlAttributes service

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

```
GetUnitControlAttributes-Request ::= Identifier -- Unit Control Name

GetUnitControlAttributes-Response ::= SEQUENCE {
    domains                 [0] IMPLICIT SEQUENCE OF Identifier,
    programInvocations      [1] IMPLICIT SEQUENCE OF Identifier }
```

13.11.1 GetUnitControlAttributes-Request

The abstract syntax of the **getUCAttributes** choice of the AdditionalService-Request type shall be the GetUnitControlAttributes-Request.

13.11.2 GetUnitControlAttributes-Response

The abstract syntax of the **getUCAttributes** choice of the AdditionalService-Response type shall be the GetUnitControlAttributes-Response.

13.12 LoadUnitControlFromFile service

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

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

LoadUnitControlFromFile-Response ::= NULL

LoadUnitControlFromFile-Error ::= CHOICE {
    none                    [0] IMPLICIT NULL,
    domain                  [1] IMPLICIT Identifier,
    programInvocation       [2] IMPLICIT Identifier
```

}

13.12.1 LoadUnitControlFromFile-Request

The abstract syntax of the **loadUCFromFile** choice of the AdditionalService-Request type shall be the LoadUnitControlFromFile-Request.

13.12.2 LoadUnitControlFromFile-Response

The abstract syntax of the **loadUCFromFile** choice of the AdditionalService-Response type shall be the LoadUnitControlFromFile-Response.

13.12.3 LoadUnitControlFromFile-Error

The abstract syntax of the **loadUCFromFile** choice of the AdditionalService-Error type shall be the LoadUnitControlFromFile-Error.

13.13 StoreUnitControlToFile service

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

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

StoreUnitControlToFile-Response ::= NULL
```

13.13.1 StoreUnitControlToFile-Request

The abstract syntax of the **storeUCToFile** choice of the AdditionalService-Request type shall be the StoreUnitControlToFile-Request.

13.13.2 StoreUnitControlToFile-Response

The abstract syntax of the **storeUCToFile** choice of the AdditionalService-Response type shall be the StoreUnitControlToFile-Response.

13.14 DeleteUnitControl service

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

```
DeleteUnitControl-Request ::= Identifier -- Unit Control Name

DeleteUnitControl-Response ::= NULL

DeleteUnitControl-Error ::= CHOICE {
    domain                   [0] IMPLICIT Identifier,
    programInvocation        [1] IMPLICIT Identifier    }
```


13.14.1 DeleteUnitControl-Request

The abstract syntax of the **deleteUC** choice of the AdditionalService-Request type shall be the DeleteUnitControl-Request.

13.14.2 DeleteUnitControl-Response

The abstract syntax of the **deleteUC** choice of the AdditionalService-Response type shall be the DeleteUnitControl-Response.

13.14.3 DeleteUnitControl-Error

The abstract syntax of the **deleteUC** choice of the AdditionalService-Error type shall be the DeleteUnitControl-Error.

14 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
DeleteVariableAccess
DefineNamedVariableList
GetNamedVariableListAttributes
DeleteNamedVariableList
DefineNamedType
GetNamedTypeAttributes
DeleteNamedType

14.1 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.

14.2 Protocol For Specifying Types

14.2.1 TypeSpecification

The abstract syntax of the Type Specification parameter is specified below. Subclause 14.1 describes the derivations of all parameters for which explicit derivations are not provided in this clause.

```
TypeSpecification ::= CHOICE {
    typeName            [0] ObjectName,
    typeDescription     TypeDescription   }
```

14.3 Protocol For Specifying Alternate Access

14.3.1 AlternateAccess

The abstract syntax of the Alternate Access parameter is specified below. Subclause 14.1 describes the derivations of all parameters for which explicit derivations are not provided in this clause.

```
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, 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, 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, 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 14.1. If the Access Selection parameter specifies INDEX-RANGE and if Low Index and Number Of Elements are both equal to zero, 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, the **selectAccess** choice shall be selected. The **accessSelection** field shall be derived from the Access Selection, Component, Index, and Index Range parameters according to 14.1. If the Access Selection parameter specifies INDEX-RANGE and if Low Index and Number Of Elements are both equal to zero, 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.

14.4 Protocol For Specifying Data Values

14.4.1 AccessResult

The abstract syntax of the Access Result parameter shall be as specified below. Subclause 14.1 describes the derivations of all parameters for which explicit derivations are not provided in this clause.

```
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.

14.4.2 Data

The abstract syntax of the Data parameter shall be as specified below. Subclause 14.1 describes the derivations of all parameters for which explicit derivations are not provided in this clause.

```
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, -- shall not be negative
    floating-point          [7] IMPLICIT FloatingPoint,
IF ( real )
    --    [8] is reserved for the type defined in Annex F
    real                    [8] IMPLICIT REAL,
ENDIF
    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, -- shall not be negative
    booleanArray            [14] IMPLICIT BIT STRING,
    objId                   [15] IMPLICIT OBJECT IDENTIFIER,
    ...,
    mMSString               [16] MMSString
    }
```

14.4.2.1 Derivation

The derivation for the Data parameter is as follows:

a) If Kind Of Data is equal to ARRAY, 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) IF Kind Of Data is equal to STRUCTURE, 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) IF Kind Of Data is equal to SIMPLE, the value of the Class parameter shall select the choice for the Data as specified in 14.1.

14.4.2.2 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 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.

3) 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 9506-1 and ISO 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.

14.4.2.3 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.

14.4.3 DataAccessError

The abstract syntax of the Data Access Error parameter is specified below. Subclause 14.1 describes the derivations of all parameters for which explicit derivations are not provided in this clause.

```
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
    object-value-invalid            (11)    -- OBJECT-VALUE-INVALID
    }
```

14.5 Protocol for Specifying Access To Variables

14.5.1 VariableAccessSpecification

The abstract syntax of the Variable Access Specification parameter is specified below. Subclause 14.1 describes the derivations of all parameters for which explicit derivations are not provided in this clause.

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


14.5.2 VariableSpecification

The abstract syntax of the Variable Specification parameter is given below. Subclause 14.1 describes the derivations of all parameters for which explicit derivations are not provided in this clause.

```
VariableSpecification ::= CHOICE {
    name                        [0] ObjectName,
    address                     [1] Address,
    variableDescription         [2] IMPLICIT SEQUENCE {
        address                     Address,
        typeSpecification           TypeSpecification
        },
    -- the following element is only present to support the services
    -- defined in annex E
IF ( vsca )
    scatteredAccessDescription  [3] IMPLICIT ScatteredAccessDescription,
ENDIF
    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.

14.6 Read

The abstract syntax of the **read** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations 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 }
```

14.6.1 Read-Request

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

14.6.2 Read-Response

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

14.7 Write

The abstract syntax of the **write** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations 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  }
```

14.7.1 Write-Request

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

14.7.2 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.

14.8 InformationReport

The abstract syntax of the **informationReport** choice of the UnconfirmedService type is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations of all parameters for which explicit derivations are not provided in this clause.

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

14.8.1 InformationReport

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

NOTE The InformationReport service is unconfirmed.

14.9 GetVariableAccessAttributes

The abstract syntax of the **getVariableAccessAttributes** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations 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,
    typeDescription         [2] TypeDescription,
    accessControlList       [3] IMPLICIT Identifier OPTIONAL,
                -- Shall not appear in minor version one or two
IF ( sem )
    meaning                 [4] IMPLICIT VisibleString OPTIONAL
ENDIF
    }
```


14.9.1 GetVariableAccessAttributes-Request

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

14.9.2 GetVariableAccessAttributes-Response

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

14.9.2.1 Access Control List

The accessControlList parameter shall appear if and only if the **aco** CBB has been negotiated.

14.10 DefineNamedVariable

The abstract syntax of the defineNamedVariable choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations 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
```

14.10.1 DefineNamedVariable-Request

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

14.10.2 DefineNamedVariable-Response

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

14.11 DeleteVariableAccess

The abstract syntax of the **deleteVariableAccess** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations 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
```

14.11.1 DeleteVariableAccess-Request

The abstract syntax of the **deleteVariableAccess** choice of the ConfirmedServiceRequest type shall be the DeleteVariableAccess-Request type.

14.11.2 DeleteVariableAccess-Response

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

14.12 DefineNamedVariableList

The abstract syntax of the **defineNamedVariableList** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations 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,
IF (valt)
        alternateAccess             [5] IMPLICIT AlternateAccess OPTIONAL
ENDIF
        }   }

DefineNamedVariableList-Response ::= NULL
```

14.12.1 DefineNamedVariableList-Request

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

14.12.2 DefineNamedVariableList-Response

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

14.13 GetNamedVariableListAttributes

The abstract syntax of the **getNamedVariableListAttributes** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations 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    },
    accessControlList           [2] IMPLICIT Identifier OPTIONAL
            -- Shall not appear in minor version one or two
    }
```

14.13.1 GetNamedVariableListAttributes-Request

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

14.13.2 GetNamedVariableListAttributes-Response

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

14.13.2.1 Access Control List

The accessControlList parameter shall appear if and only if the **aco** CBB has been negotiated.

14.14 DeleteNamedVariableList

The abstract syntax of the **deleteNamedVariableList** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations 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
```

14.14.1 DeleteNamedVariableList-Request

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

14.14.2 DeleteNamedVariableList-Response

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

14.15 DefineNamedType

The abstract syntax of the **defineNamedType** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations of all parameters for which explicit derivations are not provided in this clause.

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

DefineNamedType-Response ::= NULL
```

14.15.1 DefineNamedType-Request

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

14.15.2 DefineNamedType-Response

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

14.16 GetNamedTypeAttributes

The abstract syntax of the **getNamedTypeAttributes** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations of all parame ters for which explicit derivations are not provided in this clause.

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

GetNamedTypeAttributes-Response ::= SEQUENCE {
    mmsDeletable            [0] IMPLICIT BOOLEAN,
    typeSpecification       TypeSpecification,
    accessControlList       [1] IMPLICIT Identifier OPTIONAL,
                    -- Shall not appear in minor version one or two
IF ( sem )
    meaning                 [4] IMPLICIT VisibleString OPTIONAL
ENDIF
    }
```

14.16.1 GetNamedTypeAttributes-Request

The abstract syntax of the **getNamedTypeAttributes** choice of the ConfirmedServiceRequest type shall be the GetNamedTypeAttributes-Request type.

14.16.2 GetNamedTypeAttributes-Response

The abstract syntax of the **getNamedTypeAttributes** choice of the ConfirmedServiceResponse type shall be the GetNamedTypeAttributes-Response type.

14.16.2.1 Access Control List

The accessControlList parameter shall appear if and only if the **aco** CBB has been negotiated.

14.17 DeleteNamedType

The abstract syntax of the **deleteNamedType** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse types is specified below and described in the paragraphs which follow. Subclause 14.1 describes the derivations 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
```


14.17.1 DeleteNamedType-Request

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

14.17.2 DeleteNamedType-Response

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

15 Data Exchange Protocol

15.1 Introduction

This clause describes the protocol required for realization of the services which are defined in Clause15 of ISO 9506-1. This includes:

GetDataExchangeAttributes
ExchangeData

15.2 ExchangeData

The abstract syntax of the **exchangeData** 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.

```
ExchangeData-Request ::= SEQUENCE {
    dataExchangeName        [0] ObjectName,
    listOfRequestData       [1] IMPLICIT SEQUENCE OF Data     }

ExchangeData-Response ::= SEQUENCE {
    listOfResponseData      [0] IMPLICIT SEQUENCE OF Data     }
```

15.2.1 ExchangeData-Request

The abstract syntax of the **exchangeData** choice of the ConfirmedServiceRequest type shall be the ExchangeData-Request.

15.2.2 ExchangeData-Response

The abstract syntax of the **exchangeData** choice of the ConfirmedServiceResponse type shall be the ExchangeData-Response.

15.3 GetDataExchangeAttributes

The abstract syntax of the **getDataExchangeAttributes** choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified 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.

```
GetDataExchangeAttributes-Request ::= ObjectName -- Data Exchange Type Name

GetDataExchangeAttributes-Response ::= SEQUENCE {
    inUse                           [0] IMPLICIT BOOLEAN,
    listOfRequestTypeDescriptions   [1] IMPLICIT SEQUENCE OF TypeDescription,
    listOfResponseTypeDescriptions  [2] IMPLICIT SEQUENCE OF TypeDescription,
    programInvocation               [3] IMPLICIT Identifier OPTIONAL,
IF (aco)
    accessControlList               [4] IMPLICIT Identifier OPTIONAL
```

```
                    -- Shall not appear in minor version one or two
ENDIF
    }
```

15.3.1 GetDataExchangeAttributes-Request

The abstract syntax of the **getDataExchangeAttributes** choice of the ConfirmedServiceRequest type shall be the GetDataExchangeAttributes-Request.

15.3.2 GetDataExchangeAttributes-Response

The abstract syntax of the **getDataExchangeAttributes** choice of the ConfirmedServiceResponse type shall be the GetDataExchangeAttributes-Response.

15.3.2.1 Program Invocation

The presence of the **programInvocation** element of the GetDataExchangeAttributes-Response shall indicate that the value of the Linked attribute of the Data Exchange Object is true. If present, the value of this parameter shall convey the name of the Program Invocation referenced by the Program Invocation Reference attribute of the Data Exchange Object.

15.3.2.2 Access Control List

The **accessControlList** parameter shall appear if and only if the **aco** CBB has been negotiated.

16 Semaphore Management Protocol

16.1 Introduction

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

TakeControl
RelinquishControl
DefineSemaphore
DeleteSemaphore
ReportSemaphoreStatus
ReportPoolSemaphoreStatus
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.

16.2 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 OPTIONAL     }
```

16.2.1 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.

If the value of the Acceptable Delay parameter in the confirmation primitive is an integer, that value shall appear as the value of the **acceptableDelay** field. If the value of the Acceptable Delay parameter is FOREVER, the **acceptableDelay** field shall not be present.

If the value of the Control Time Out parameter in the confirmation primitive is an integer, that value shall appear as the value of the **controlTimeOut** field. If the value of the Control Time Out parameter is FOREVER, the **controlTimeOut** field shall not be present.

The **abortOnTimeOut** field shall be present if and only if the Abort On Time Out field is present in the confirmation primitive.

16.2.2 TakeControl-Response

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

16.3 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
```

16.3.1 RelinquishControl-Request

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

16.3.2 RelinquishControl-Response

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

16.4 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
```

16.4.1 DefineSemaphore-Request

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

16.4.2 DefineSemaphore-Response

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

16.5 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
```

16.5.1 DeleteSemaphore-Request

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

16.5.2 DeleteSemaphore-Response

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

16.6 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,
IF (aco)
    accessControlList           [5] IMPLICIT Identifier OPTIONAL
                    -- Shall not appear in minor version one or two
ENDIF
    }
```

16.6.1 ReportSemaphoreStatus-Request

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

16.6.2 ReportSemaphoreStatus-Response

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

16.6.2.1 Access Control List

The **accessControlList** parameter shall appear if and only if the **aco** CBB has been negotiated.

16.7 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
    }
```

16.7.1 ReportPoolSemaphoreStatus-Request

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

16.7.2 ReportPoolSemaphoreStatus-Response

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

16.8 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      }
```

16.8.1 ReportSemaphoreEntryStatus-Request

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

16.8.2 ReportSemaphoreEntryStatus-Response

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

16.8.3 SemaphoreEntry

```
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 }
```

16.9 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      }
```

17 Operator Communication Protocol

17.1 Introduction

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

Input
Output

17.2 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,
IF (output)
    listOfPromptData        [2] IMPLICIT SEQUENCE OF MMSString OPTIONAL,
ENDIF
    inputTimeOut            [3] IMPLICIT Unsigned32 OPTIONAL   }

Input-Response ::= MMSString -- Input String
```


17.2.1 Input-Request

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

17.2.2 Input-Response

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

17.3 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 MMSString   }

Output-Response ::= NULL
```

17.3.1 Output-Request

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

17.3.2 Output-Response

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

18 Event Management Protocol

18.1 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

TriggerEvent
EventNotification
AcknowledgeEventNotification
GetAlarmSummary
GetAlarmEnrollmentSummary

18.2 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
```

18.2.1 TriggerEvent-Request

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

18.2.2 TriggerEvent-Response

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

18.3 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,
        successOrFailure            CHOICE {
            success                     [0] IMPLICIT SEQUENCE {
                confirmedServiceResponse    ConfirmedServiceResponse,
                cs-Response-Detail          [79] CS-Response-Detail OPTIONAL
                                        -- shall not be transmitted if value is the
                                        -- value of a tagged type derived from NULL
                },
            failure                     [1] IMPLICIT SEQUENCE {
                modifierPosition            [0] IMPLICIT Unsigned32 OPTIONAL,
                serviceError                [1] IMPLICIT ServiceError }
            }
        } OPTIONAL
    }

CS-EventNotification ::= [0] Choice {
    string              [0] IMPLICIT VisibleString,
    index               [1] IMPLICIT INTEGER,
    noEnhancement       NULL    }
```

18.3.1 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.

18.3.1.1 actionResult

If 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 **successOrFailure** field shall be determined as follows.

a) If the Success or Failure sub-parameter of the Action Result parameter of the EventNotification.request primitive is equal to true, the **successOrFailure** field shall select **success** and the value of the Success or Failure 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 Success or Failure 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 &confirmedServiceRequest field 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, and failure occurred in execution of one of the modifiers specified in the &Modifiers field of the Event Action object, the **modifierPosition** choice of the **failure** selection shall be conveyed, indication the modifier causing the failure.

c) If **failure** is selected, and failure occurred in the execution of the requested Confirmed Service, the Result(-) parameter of the service requested by the &confirmedServiceRequest field of the Event Action object shall be conveyed using the **serviceError** choice of the **failure** selection, and 5.5 shall apply.

18.3.1.1.1 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 that corresponds to the choice made of the ConfirmedServiceResponse.

18.3.1.2 Display Enhancement

The abstract syntax of the **eventNotification** choice of the Unconfirmed-Detail shall be the CS-EventNotification and this field shall convey the Display Enhancement parameter.

18.4 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 }

AcknowledgeEventNotification-Response ::= NULL
```

18.4.1 AcknowledgeEventNotification-Request

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

18.4.2 AcknowledgeEventNotification-Response

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

18.5 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 },
    displayEnhancement              [4] EN-Additional-Detail OPTIONAL,
            -- shall not be transmitted if the value is NULL
    timeOfLastTransitionToActive    [5] EventTime OPTIONAL,
    timeOfLastTransitionToIdle      [6] EventTime OPTIONAL      }

EN-Additional-Detail ::= [0] Choice {
    string                          [0] IMPLICIT VisibleString,
    index                           [1] IMPLICIT INTEGER,
    noEnhancement                   NULL    }
```

18.5.1 GetAlarmSummary-Request

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

18.5.2 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.

18.5.2.1 listOfAlarmSummary

The **listOfAlarmSummary** field shall be the List Of Alarm Summary 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.

18.5.2.1.1 displayEnhancement

The **displayEnhancement** field of a given AlarmSummary shall be the Display Enhancement parameter from the corresponding Alarm Summary of the List Of Alarm Summary parameter of the GetAlarmSummary.response primitive and shall appear as the Display Enhancement 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 the EN-Additional-Detail type and 5.5 shall apply.

18.6 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,
    displayEnhancement              [5] EN-Additional-Detail OPTIONAL,
                -- shall not be transmitted 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     }
```

18.6.1 GetAlarmEnrollmentSummary-Request

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

18.6.2 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.

18.6.2.1 listOfAlarmEnrollmentSummary

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.

18.6.2.1.1 displayEnhancement

The **displayEnhancement** field of a given AlarmEnrollmentSummary shall be the Display Enhancement 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 Display Enhancement 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 the EN-Additional-Detail type and 5.5 shall apply.

18.7 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 for which explicit derivations are not provided in this clause.

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

19 Event Condition Protocol

19.1 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

19.2 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

CS-DefineEventCondition-Request ::= [0] Choice {
    string                  [0] IMPLICIT VisibleString,
    index                   [1] IMPLICIT INTEGER,
    noEnhancement           NULL    }
```

19.2.1 DefineEventCondition-Request

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

19.2.2 DefineEventCondition-Response

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

19.2.3 CS-DefineEventCondition-Request

The abstract syntax of the **defineEventCondition** choice of the Request-Detail type shall be the CS-DefineEventCondition-Request and this field shall convey the value of the Display Enhancement parameter, if present.

19.3 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
```

19.3.1 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.

19.3.2 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(+).

19.4 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,
IF (aco)
    accessControlList           [8] IMPLICIT Identifier OPTIONAL
                        -- Shall not appear in minor version one or two
ENDIF
    }

CS-GetEventConditionAttributes-Response ::= SEQUENCE {
    groupPriorityOverride       [0] CHOICE {
        priority                    [0] IMPLICIT Priority,
        undefined                   [1] IMPLICIT NULL  } OPTIONAL,
    listOfReferencingECL        [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
    displayEnhancement          [2] CHOICE {
        string                      [0] IMPLICIT VisibleString,
        index                       [1] IMPLICIT INTEGER,
        noEnhancement               [2] IMPLICIT NULL  }
    }
```

19.4.1 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.

19.4.2 GetEventConditionAttributes-Response

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

19.4.2.1 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.

19.4.2.2 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.

19.4.2.3 accessControlList

The **accessControlList** parameter shall appear if and only if the **aco** CBB has been negotiated.

19.4.3 CS-GetEventConditionAttributes-Response

The abstract syntax of the **getEventConditionAttributes** choice of the Response-Detail shall be the CS-GetEventConditionAttributes-Response type.

19.5 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 }
```

19.5.1 ReportEventConditionStatus-Request

The abstract syntax of the **reportEventConditionStatus** choice of the ConfirmedServiceRequest type shall be ReportEventConditionStatus-Request.

19.5.2 ReportEventConditionStatus-Response

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

19.6 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
            -- At least one of enabled, priority, alarmSummaryReports, or
            -- evaluationInterval shall be present.
    }

AlterEventConditionMonitoring-Response ::= NULL

CS-AlterEventConditionMonitoring-Request ::= SEQUENCE {
    changeDisplay           CHOICE {
        string                  [0] IMPLICIT VisibleString,
        index                   [1] IMPLICIT INTEGER,
        noEnhancement           [2] NULL    } OPTIONAL
    }
```

19.6.1 AlterEventConditionMonitoring-Request

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

19.6.2 AlterEventConditionMonitoring-Response

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

19.6.3 CS-AlterEventConditionMonitoring-Request

The abstract syntax of the **alterEventConditionMonitoring** choice of the Request-Detail shall be the CS-AlterEventConditionMonitoring-Request and this field shall convey the Display Enhancement parameter. This field shall be included if and only if the Display Enhancement parameter is present in the AlterEventConditionMonitoring.indication service primitive.

20 Event Action Protocol

20.1 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

DefineEventAction
DeleteEventAction
GetEventActionAttributes
ReportEventActionStatus

20.2 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 value is the value
                        -- of a tagged type derived from NULL
    }

DefineEventAction-Response ::= NULL
```

20.2.1 DefineEventAction-Request

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

20.2.1.1 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.

20.2.2 DefineEventAction-Response

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

20.3 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
```

20.3.1 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.

20.3.2 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(+).

20.4 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 value is the value
                    -- of a tagged type derived from NULL
IF (aco)
    accessControlList           [3] IMPLICIT Identifier OPTIONAL
                -- Shall not appear in minor version one or two
ENDIF
```

}

20.4.1 GetEventActionAttributes-Request

The abstract syntax of the **getEventActionAttributes** choice of the ConfirmedServiceRequest type shall be the GetEventActionAttributes-Request.

20.4.2 GetEventActionAttributes-Response

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

20.4.2.1 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 Request-Detail type that corresponds to the choice made of the ConfirmedServiceRequest.

20.4.2.2 Access Control List

The **accessControlList** parameter shall appear if and only if the **aco** CBB has been negotiated.

20.5 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

20.5.1 ReportEventActionStatus-Request

The abstract syntax of the **reportEventActionStatus** choice of the ConfirmedServiceRequest type shall be the ReportEventActionStatus-Request.

20.5.2 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.

21 Event Enrollment Protocol

21.1 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

DefineEventEnrollment
DeleteEventEnrollment
GetEventEnrollmentAttributes
ReportEventEnrollmentStatus
AlterEventEnrollment

services and the AttachToEventCondition service modifier.

21.2 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    }

DefineEventEnrollment-Response ::= NULL

DefineEventEnrollment-Error ::= ObjectName

CS-DefineEventEnrollment-Request ::= [0] Choice {
    string                      [0] IMPLICIT VisibleString,
    index                       [1] IMPLICIT INTEGER,
    noEnhancement               NULL    }
```

21.2.1 DefineEventEnrollment-Request

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

21.2.2 DefineEventEnrollment-Response

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

21.2.3 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 parameter of the Result(-) parameter of the DefineEventEnrollment.response primitive, and shall appear as the Object Not Defined parameter of the Result(-) parameter of the DefineEventEnrollment.confirm primitive, if issued.

21.2.4 CS-DefineEventEnrollment-Request

The abstract syntax of the **defineEventEnrollment** choice of the Request-Detail shall be the CS-DefineEventEnrollment-Request and shall convey the value of the Display Enhancement parameter.

21.3 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
```

21.3.1 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 Scope Of Delete parameter indicates the List of Event Enrollment Names parameter, the DeleteEventEnrollment-Request shall select the **specific** choice. This choice shall contain the value of the List Of Event Enrollment Names parameter from the DeleteEventEnrollment.request service primitive.

If the Scope Of Delete parameter indicates the Event Condition Name parameter, the DeleteEventEnrollment-Request shall select the **ec** choice. This choice shall contain the value of the Event Condition Name parameter of the DeleteEventEnrollment.request service primitive.

If the Scope Of Delete parameter indicates the Event Action Name parameter, the DeleteEventEnrollment-Request shall select the **ea** choice. This choice shall contain the value of the Event Action Name parameter from the DeleteEventEnrollment.request service primitive.

21.3.2 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(+).

21.4 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),
        client                      (1),
        ec                          (2),
        ea                          (3)     } 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 {
    listOfEEAttributes          [0] IMPLICIT SEQUENCE OF EEAttributes,
    moreFollows                 [1] IMPLICIT BOOLEAN DEFAULT FALSE      }

EEAttributes ::= 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,
    displayEnhancement              [9] Choice {
        string                          [0] IMPLICIT VisibleString,
        index                           [1] IMPLICIT INTEGER,
        noEnhancement                   NULL    },
                -- shall not be transmitted if the value is NULL
IF ( aco )
    accessControlList               [11] IMPLICIT Identifier OPTIONAL
                -- shall not appear in minor version one or two
ENDIF
    }
```

21.4.1 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.

21.4.1.1 scopeOfRequest

The **scopeOfRequest** field shall indicate the selection made in the Scope Of Request parameter of the request primitive. If the List of Event Enrollment Names was selected in the request primitive, the **specific** choice of the **scopeOfRequest** field shall be selected. If the Client Application was selected in the request primitive, the **client** choice of the **scopeOfRequest** field shall be selected. If the Event Condition Name was selected in the request primitive, the **ec** choice of the **scopeOfRequest** field shall be selected. If the Event Action Name was selected in the request primitive, the **ea** choice of the **scopeOfRequest** field shall be selected.

21.4.1.2 eventEnrollmentNames

If the List of Event Enrollment Names was selected for the Scope of Request parameter, this field shall contain the value of this parameter. Otherwise this field shall not appear.

21.4.1.3 clientApplication

If the Client Application was selected for the Scope of Request parameter, and this parameter does not specify the MMS client of this service request, this field shall contain the value of that parameter. Otherwise this field shall not appear.

If the Event Condition Name or Event Action Name was selected for the Scope of Request parameter, and a Client Application was specified as an option for this parameter, this field shall contain the value of that parameter. Otherwise this field shall not appear.

21.4.1.4 eventConditionName

If the Event Condition Name was selected for the Scope of Request parameter, this field shall contain the value of this parameter. Otherwise this field shall not appear.

21.4.1.5 eventActionName

If the Event Action Name was selected for the Scope of Request parameter, this field shall contain the value of this parameter. Otherwise this field shall not appear.

21.4.2 GetEventEnrollmentAttributes-Response

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

21.4.2.1 listOfEEAttributes

The **listOfEEAttributes** field shall be the List Of EE Attributes parameter of the GetEventEnrollmentAttributes.response primitive and shall appear as the List Of EE Attributes parameter of the GetEventEnrollmentAttributes.confirm primitive. This field shall contain zero or more occurrences of the EEAttributes type, each containing the value of a single EEAttributes parameter of the List Of EE Attributes parameter, taken in the order listed. Subclause 5.5 shall be applied to each occurrence of the EE Attributes parameter of the List Of EE Attributes parameter in order to derive the corresponding element of the List Of EE Attributes parameter.

21.4.2.1.1 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.

21.4.2.1.2 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.

21.4.2.1.3 Access Control List

The **accessControlList** parameter shall appear if and only if the **aco** CBB has been negotiated.

21.5 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     }
```

21.5.1 ReportEventEnrollmentStatus-Request

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

21.5.2 ReportEventEnrollmentStatus-Response

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

21.6 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      }

CS-AlterEventEnrollment-Request ::= SEQUENCE {
    changeDisplay                CHOICE {
        string                       [0] IMPLICIT VisibleString,
        index                        [1] IMPLICIT INTEGER,
        noEnhancement                [2] NULL }     OPTIONAL }
```

21.6.1 AlterEventEnrollment-Request

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

21.6.2 AlterEventEnrollment-Response

The abstract syntax of the **alterEventEnrollment** choice for the ConfirmedServiceResponse type shall be the AlterEventEnrollment-Response.

21.6.2.1 Current State

The **undefined** choice of the **currentState** field 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.

21.6.3 CS-AlterEventEnrollment-Request

The abstract syntax of the **alterEventEnrollment** choice of the Request-Detail shall be CS-AlterEventEnrollment-Request and shall convey the value of Display Enhancement parameter. If the Display Enhancement parameter is present in the request primitive, the **changeDisplay** field shall appear in the CS-AlterEventEnrollment-Request field with the appropriate type selected. If the Display Enhancement parameter is not present in the request primitive, the **changeDisplay** field shall not appear in the CS-AlterEventEnrollment-Request, and this field shall consist of an empty SEQUENCE.

21.7 Supporting Productions

The abstract syntax of the various supporting type definitions for the event management protocol is given below.

21.7.1 EE-State

The EE-State type is used for certain parameters to convey combined state information of the Event Condition and the Event Enrollment objects.

```
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
undefined       (8)   -- UNDEFINED    }
```

22 Event Condition List Protocol

22.1 Introduction

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

DefineEventConditionList
DeleteEventConditionList
AddEventConditionListReference
RemoveEventConditionListReference
GetEventConditionListAttributes
ReportEventConditionListStatus
AlterEventConditionListMonitoring

22.2 DefineEventConditionList protocol

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

```
DefineEventConditionList-Request ::= SEQUENCE {
    eventConditionListName          [0] ObjectName,
    listOfEventConditionName        [1] IMPLICIT SEQUENCE OF ObjectName,
IF ( recl )
    listOfEventConditionListName    [2] IMPLICIT SEQUENCE OF ObjectName OPTIONAL
        -- shall appear if an only if recl has been negotiated.
ENDIF
    }

DefineEventConditionList-Response ::= NULL

DefineEventConditionList-Error ::= ObjectName
```

22.2.1 DefineEventConditionList-Request

The abstract syntax of the **defineECL** choice of the ConfirmedServiceRequest type shall be the DefineEventConditionList-Request.

22.2.2 DefineEventConditionList-Response

The abstract syntax of the **defineECL** choice of the ConfirmedServiceResponse type shall be the DefineEventConditionList-Response.

22.2.3 DefineEventConditionList-Error

The abstract syntax of the **defineECL** choice of the AdditionalService-Error shall be the DefineEventConditionList-Error, which shall be the Object in error parameter of the Result(-) parameter of the DefineEventConditionList.response primitive, and shall appear as the Object in error parameter of the DefineEventConditionList.confirm primitive, if issued.

22.3 DeleteEventConditionList protocol

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

```
DeleteEventConditionList-Request ::= ObjectName  -- EventConditionListName

DeleteEventConditionList-Response ::= NULL
```

22.3.1 DeleteEventConditionList-Request

The abstract syntax of the **deleteECL** choice of the ConfirmedServiceRequest type shall be the DeleteEventConditionList-Request.

22.3.2 DeleteEventConditionList-Response

The abstract syntax of the **deleteECL** choice of the ConfirmedServiceResponse type shall be the DeleteEventConditionList-Response.

22.4 AddEventConditionListReference protocol

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

```
AddEventConditionListReference-Request ::= SEQUENCE {
    eventConditionListName          [0] ObjectName,
    listOfEventConditionName        [1] IMPLICIT SEQUENCE OF ObjectName,
IF ( recl )
    listOfEventConditionListName    [2] IMPLICIT SEQUENCE OF ObjectName OPTIONAL
        -- shall appear if an only if recl has been negotiated.
ENDIF
    }

AddEventConditionListReference-Response ::= NULL

AddEventConditionListReference-Error ::= ObjectName
```

22.4.1 AddEventConditionListReference-Request

The abstract syntax of the **addECLReference** choice of the ConfirmedServiceRequest type shall be the AddEventConditionListReference-Request.

22.4.2 AddEventConditionListReference-Response

The abstract syntax of the **addECLReference** choice of the ConfirmedServiceResponse type shall be the AddEventConditionListReference-Response.

22.4.3 AddEventConditionListReference-Error

The abstract syntax of the **addECLReference** choice of the AdditionalService-Error shall be the AddEventConditionListReference-Error, which shall be the Object in error parameter of the Result(-) parameter of the AddEventConditionListReference.response primitive, and shall appear as the Object in error parameter of the AddEventConditionListReference.confirm primitive, if issued.

22.5 RemoveEventConditionListReference protocol

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

```
RemoveEventConditionListReference-Request ::= SEQUENCE {
    eventConditionListName          [0] ObjectName,
    listOfEventConditionName        [1] IMPLICIT SEQUENCE OF ObjectName,
IF ( recl )
    listOfEventConditionListName    [2] IMPLICIT SEQUENCE OF ObjectName OPTIONAL
        -- shall appear if an only if recl has been negotiated.
ENDIF
    }

RemoveEventConditionListReference-Response ::= NULL

RemoveEventConditionListReference-Error ::= CHOICE {
    eventCondition                  [0] ObjectName,
    eventConditionList              [1] ObjectName    }
```

22.5.1 RemoveEventConditionListReference-Request

The abstract syntax of the **removeECLReference** choice of the ConfirmedServiceRequest type shall be the RemoveEventConditionListReference-Request.

22.5.2 RemoveEventConditionListReference-Response

The abstract syntax of the **removeECLReference** choice of the ConfirmedServiceResponse type shall be the RemoveEventConditionListReference-Response.

22.5.3 RemoveEventConditionListReference-Error

The abstract syntax of the **removeECLReference** choice of the AdditionalService-Error shall be the RemoveEventConditionListReference-Error, which shall be the Object in error parameter of the Result(-) parameter of the RemoveEventConditionListReference.response primitive, and shall appear as the Object in error parameter of the RemoveEventConditionListReference.confirm primitive, if issued.

22.6 GetEventConditionListAttributes protocol

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

```
GetEventConditionListAttributes-Request ::= ObjectName  -- eventConditionListName

GetEventConditionListAttributes-Response ::= SEQUENCE {
    listOfEventConditionName        [1] IMPLICIT SEQUENCE OF ObjectName,
IF ( recl )
    listOfEventConditionListName    [2] IMPLICIT SEQUENCE OF ObjectName OPTIONAL
        -- shall appear if an only if recl has been negotiated.
ENDIF
    }
```

22.6.1 GetEventConditionListAttributes-Request

The abstract syntax of the **getECLAttributes** choice of the ConfirmedServiceRequest type shall be the GetEventConditionListAttributes-Request.

22.6.2 GetEventConditionListAttributes-Response

The abstract syntax of the **getECLAttributes** choice of the ConfirmedServiceResponse type shall be the GetEventConditionListAttributes-Response.

22.7 ReportEventConditionListStatus protocol

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

```
ReportEventConditionListStatus-Request ::= SEQUENCE {
    eventConditionListName      [0] ObjectName -- Event Condition List Name,
    continueAfter               [1] IMPLICIT Identifier OPTIONAL     }

ReportEventConditionListStatus-Response ::= SEQUENCE {
    listOfEventConditionStatus  [1] IMPLICIT SEQUENCE OF EventConditionStatus,
    moreFollows                 [2] IMPLICIT BOOLEAN DEFAULT TRUE

EventConditionStatus ::= SEQUENCE {
    eventConditionName          [0] ObjectName,
    currentState                [1] IMPLICIT EC-State,
    numberOfEventEnrollments    [2] IMPLICIT Unsigned32,
    enabled                     [3] IMPLICIT BOOLEAN OPTIONAL,
    timeOfLastTransitionToActive [4] EventTime OPTIONAL,
    timeOfLastTransitionToIdle  [5] EventTime OPTIONAL       }
```

22.7.1 ReportEventConditionListStatus-Request

The abstract syntax of the **reportECLStatus** choice of the ConfirmedServiceRequest type shall be the ReportEventConditionListStatus-Request.

22.7.2 ReportEventConditionListStatus-Response

The abstract syntax of the **reportECLStatus** choice of the ConfirmedServiceResponse type shall be the ReportEventConditionListStatus-Response.

22.8 AlterEventConditionListMonitoring protocol

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

```
AlterEventConditionListMonitoring-Request ::= SEQUENCE {
    eventConditionListName      [0] ObjectName,
    enabled                     [1] IMPLICIT BOOLEAN,
    priorityChange              [2] CHOICE {
        priorityValue               [0] IMPLICIT INTEGER,
        priorityReset               [1] IMPLICIT NULL       } OPTIONAL
    }

AlterEventConditionListMonitoring-Response ::= NULL
```

22.8.1 AlterEventConditionListMonitoring-Request

The abstract syntax of the **alterECLMonitoring** choice of the ConfirmedServiceRequest type shall be the AlterEventConditionListMonitoring-Request.

22.8.2 AlterEventConditionListMonitoring-Response

The abstract syntax of the **alterECLMonitoring** choice of the ConfirmedServiceResponse type shall be the AlterEventConditionListMonitoring-Response.

23 Journal Management Protocol

23.1 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	ReportJournalStatus
WriteJournal	CreateJournal
InitializeJournal	DeleteJournal

23.2 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    }
```

23.2.1 ReadJournal-Request

The abstract syntax of the **readJournal** choice of the ConfirmedServiceRequest shall be the ReadJournal-Request.

23.2.2 ReadJournal-Response

The abstract syntax of the **readJournal** choice of the ConfirmedServiceResponse shall be the ReadJournal-Response.

23.3 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
```

23.3.1 WriteJournal-Request

The abstract syntax of the **writeJournal** choice of the ConfirmedServiceRequest shall be the WriteJournal-Request.

23.3.2 WriteJournal-Response

The abstract syntax of the **writeJournal** choice of the ConfirmedServiceResponse shall be the WriteJournal-Response.

23.4 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
```

23.4.1 InitializeJournal-Request

The abstract syntax of the **initializeJournal** choice of the ConfirmedServiceRequest shall be the InitializeJournal-Request.

23.4.2 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.

23.5 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,
IF ( aco )
      accessControlList           [2] IMPLICIT Identifier OPTIONAL
                -- Shall not appear in minor version one or two
ENDIF
      }
```

23.5.1 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.

23.5.2 ReportJournalStatus-Response

The abstract syntax of the **reportJournalStatus** choice of the ConfirmedService Response shall be the ReportJournalStatus-Response.

23.5.2.1 Access Control List

The accessControlList parameter shall appear if and only if the **aco** CBB has been negotiated.

23.6 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
```

23.6.1 CreateJournal-Request

The abstract syntax of the **createJournal** choice of the ConfirmedServiceRequest shall be the CreateJournal-Request.

23.6.2 CreateJournal-Response

The abstract syntax of the **createJournal** choice of the ConfirmedServiceResponse shall be the CreateJournal-Response.

23.7 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
```

23.7.1 DeleteJournal-Request

The abstract syntax of the **deleteJournal** choice of the ConfirmedServiceRequest shall be the DeleteJournal-Request.

23.7.2 DeleteJournal-Response

The abstract syntax of the **deleteJournal** choice of the ConfirmedServiceResponse shall be the DeleteJournal-Response.

23.8 Supporting Productions

23.8.1 EntryContent

```
EntryContent ::= SEQUENCE {
    occurrenceTime          [0] IMPLICIT TimeOfDay,
    entryForm               CHOICE {
        data                    [2] IMPLICIT SEQUENCE {
            event                   [0] IMPLICIT SEQUENCE {
                eventConditionName      [0] ObjectName,
                currentState            [1] IMPLICIT EC-State     } OPTIONAL,
            listOfVariables         [1] IMPLICIT SEQUENCE OF Journal-Variable OPTIONAL,
            },
        annotation              [3] IMPLICIT VisibleString      }
    }
```

23.8.1.1 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 DATA. 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 ANNOTATION.

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 DATA. 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 ANNOTATION.

The following END statement terminates the module opened in clause 7.

```
END
```

24 Mapping to Underlying Communication Services

This clause defines the way in which the services provided by the underlying communication system are used by the Manufacturing Messaging Protocol Machine (MMPM). Any use of these services 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), this International Standard makes use of and maps on to the services and service primitives of the underlying communication system. The MMS-user may be elements within the application process or another ASE.

24.1 Mapping of PDUs

All MMS PDUs shall be carried as user data on an underlying 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	Underlying Communication Service Primitive	
Confirmed-RequestPDU	**M-DATA request, indication**	
Confirmed-ResponsePDU	**M-DATA request, indication**	
Confirmed-ErrorPDU	**M-DAA request, indication**	
Unconfirmed-RequestPDU	**M-DATA request, indication**	
RejectPDU	**M-DATA request, indication**	
Cancel-RequestPDU	**M-DATA request, indication**	
Cancel-ResponsePDU	**M-DATA request, indication**	
Cancel-ErrorPDU	**M-DATA request, indication**	
Initiate-RequestPDU	**M-ASSOCIATE request, indication**	
Initiate-ResponsePDU	**M-ASSOCIATE response, confirm**	**(with Result parameter accepted)**
Initiate-ErrorPDU	**M-ASSOCIATE response, confirm**	**(with Result parameter rejected)**
Conclude-RequestPDU	**M-RELEASE request, indication**	
Conclude-ResponsePDU	**M-RELEASE response, confirm**	**(with Result parameter accepted)**
Conclude-ErrorPDU	**M-RELEASE response, confirm**	**(with Result parameter rejected)**

Any other mappings of MMS PDUs on to these services shall constitute a protocol error.

24.2 M-ASSOCIATE Data

Data from the M-ASSOCIATE request or response services shall be used for the initialization of attributes of the Application Association object that is created for the association. The Application Reference parameter is used to identify the peer MMS-user in this association. This value shall identify (1) the communicating node and (2) the user process within that node. This value is used to establish the value of the &client field of Application-Association object (see 8.2). For the calling MMS-user, the Responding Application Reference parameter is used as the value of the &client field; for the called MMS-user, the Calling Application Reference field is used as the value of the &client field. If the Authentication Value parameter is present is present in either the request or the response service primitive, the value of this parameter shall be used to initialize the field &authenticationValue of the Application-Association object.

24.3 Termination of Application Association

Upon receipt of a valid Conclude-RequestPDU, the MMPM shall issue
an M-RELEASE.request service primitive with the Conclude-RequestPDU contained as user data.

Upon receipt of a valid Conclude-ResponsePDU with a result parameter that 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 M-U-ABORT.request service primitive and shall deliver a conclude.confirm service primitive indicating Result(+) to the MMS-user.

24.4 Directly-Mapped Abort Service

The MMS abort service is directly mapped to the M-U-ABORT service, and hence this International Standard does not define an abort PDU.

Upon receiving an indication service primitive (either M-U-ABORT or M-P-ABORT) from the supporting communication system specifying an abort, the MMPM shall issue an abort indication service primitive to the MMS-user. If the M-ABORT request was generated by the system in which the MMS-user is located (i.e. M-P-ABORT), 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 M-U-ABORT.request service primitive.

The MMPM may, at any time, issue an abort.indication service primitive to the MMS-user and an M-P-ABORT.request service primitive as a local matter (due to locally detected conditions).

24.5 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 conclude.request service primitive, or the conclude.response 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 23 and annexes C and D), and

b) send the constructed PDU as user data on the M-DATA service primitive specified above in accordance with the requirements of ISO 9506-1 and ISO 9506-2.

24.6 Delivery of Service Primitives to an MMS-user

Upon receipt of an indication or confirm service primitive from the underlying communication system other than an M-ABORT.indication, an M-RELEASE indication, or an M-RELEASE.confirm, 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 service primitive (as defined above), and arrives in conformance with all sequencing rules defined in ISO 9506-1 and ISO 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 9506-1 and ISO 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 an M-DATA.request service primitive;

b) otherwise, the MMPM shall issue an abort.indication to the MMS-user, and issue an M-ABORT.request service primitive if an application association exists.

24.7 Right to Send Data

This International Standard requires that the underlying communication system support full duplex communication.

24.8 Reliable Underlying Service

This International Standard makes no provisions for handling of misordered messages, transmission errors, lost messages, or duplicated messages. This International Standard assumes that a reliable underlying service for communicating data between two application entities exists.

24.9 Flow Control

There is no peer flow control in MMS. The receiving MMPM may make use of flow control mechanism in the underlying communication system 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.

24.10 Use of Presentation Contexts

OSI defines a presentation context as the abstract syntax used by the application for communication with its peers and the transfer syntax that provides the encoding mechanism for transmitting this information. ISO 9506 defines only the abstract syntax of the messages exchanged between peer MMS-users. The selection of a transfer syntax is outside the scope of this standard.

Annex A provides guidance on the use of transfer syntaxes in an OSI environment. In other environments, OSI transfer syntaxes may also be employed, or other encoding schemes appropriate to those environments may be used. The only requirement on such transfer syntaxes is that it shall be possible to reconstruct the abstract syntax used in specifying MMS unambiguously from the encoding.

Agreement on the use of transfer syntaxes to be employed on a given instance of communication may be the subject of prior agreement between the parties, or it may be the subject of negotiation within the underlying services of the communication system.

24.11 Abstract Syntax Definition

This part of ISO 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 module defined in of this part of ISO 9506, clauses 7 to 16, and annexes B and C. The corresponding ASN.1 object descriptor value shall be

"mms-abstract-syntax-major-version1"

The major version number of this version of ISO 9506-1 and ISO 9506-2 shall be one. The minor version number of this version of ISO 9506-1 and ISO 9506-2 shall be four.

25 Configuration and Initialization Statement

25.1 Introduction

This clause describes the Configuration and Initialization Statement (CIS) for an implementation. It provides forms for the implementor to report the values of various fields in the implementation. In addition, it provides prescriptions for initialization of some fields at system start up. Every implementor shall complete the entire CIS. Implementor in this clause refers both to the vendor of the hardware and software necessary to realize a VMD and to the installer or owner who configures a product into a specific installation.

25.2 CIS Part One: Initialization of the VMD

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.

An implementation shall support one object of the object class VMD. For each of the fields of the VMD model, the implementor shall assign initial values. For each predefined subordinate object in the VMD, the implementor shall provide the value of the &name field and either (1) a reference to the complete definition of the object (e.g. to ISO 9506-1 for MMS defined objects) or (2) a complete definition in the form of values for all the fields of that object.

Table 1 - CIS Implementation Information

Implementation Serial Number: Date Issued:

Field of the VMD Model	CBB	Value
&executiveFunction		
&vendorName		
&modelName		
&revision		
&AbstractSyntaxes		
&accessControl		
&Capabilities		provide Table 2
&local-detail		
&AccessControlLists		provide Table 3
&Domains		provide Table 4
&ProgramInvocations		provide Table 5
&UnitControls		provide Table 6
&UnnamedVariables	vadr	provide Table 7
&NamedVariables	vnam	provide Table 8
&NamedVariableLists	vlis	provide Table 9
&NamedTypes	vnam	provide Table 10
&DataExchanges		provide Table 11
&Semaphores		provide Table 12
&OperatorStations		provide Table 13
&EventConditions		provide Table 14
&EventActions		provide Table 15
&EventEnrollments		provide Table 16
&EventConditionLists	cspi	provide Table 17
&Journals		provide Table 18
&selected-Program-Invocation	csr	

25.2.1 &executiveFunction

The implementor shall supply an Application Reference value that identifies this implementation within its network environment. The nature of this Application Reference depends on the network to which the implementation is attached. For an OSI network, this field shall be an Object Identifier.

25.2.2 &vendorName

The implementor shall supply a character string value identifying the vendor of the system.

25.2.3 &modelName

The implementor shall supply a character string value identifying the model of this implementation.

25.2.4 &revision

The implementor shall supply a character string value identifying the version of this implementation.

25.2.5 &AbstractSyntaxes

The implementor shall supply a set of abstract syntaxes that this implementation can recognize, either in the Domain upload/download operations, or in the execution argument of the Start and Resume operations, or both. This set may be empty.

25.2.6 &EATransactions

The implementator shall assign an empty set to this field.

25.2.7 &accessControl

The implementor shall supply the name of a predefined Access Control List object which describes the access control characteristics of this VMD. This object shall be included in 25.2.12.

25.2.8 &logicalStatus

At system start up, this field shall be initialised to reflect the condition of the hardware underlying the VMD. This value shall be one of the values **state-changes-allowed**, **no-state-changes-allowed**, **limited-services-permitted**, and **support-services-allowed**.

25.2.9 &Capabilities

The implementor shall fill out Table 2, describing each of the Capabilities of the VMD. The number of rows of the table shall be extended as required to accomodate all the Capabilities.

This table may be empty.

Table 2 - Capability Description

Capability (character string)	meaning	parse rule

25.2.10 &physicalStatus

At system start up, this field shall be initialised to reflect the condition of the hardware underlying the VMD. This value shall be one of the values **operational**, **partially-operational**, **inoperable**, and **needs-commissioning**.

25.2.11 &local-detail

The implementor shall supply the initial value of the &local-detail BIT STRING and the meaning of each of the bits. If there are no local-detail bits to be defined, this value shall be "B.

25.2.12 &AccessControlLists

The implementor shall supply a set of predefined Access Control List objects. For each such object, the implementor shall complete Table 3, providing either a reference to the definition or values for all the field elements of the object.

This set shall contain at least one object, the object referenced by the &accessControl field in Table 1.

Table 3 - Predefined Access Control object

Access Control List Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&readAccessCondition		
&storeAccessCondition		
&writeAccessCondition		
&loadAccessCondition		
&executeAccessCondition		
&deleteAccessCondition		
&editAccessCondition		
&AccessControlLists		
&Domains		
&ProgramInvocations		
&UnitControls		
&UnnamedVariables	vadr	
&NamedVariables	vnam	
&NamedVariableLists	vlis	
&NamedTypes	vnam	
&DataExchanges		
&Semaphores		
&OperatorStations		
&EventConditions		
&EventActions		
&EventEnrollments		
&Journals		
&EventConditionLists	cspi	

25.2.13 &Domains

The implementor shall supply a set of predefined Domain objects. For each such object, the implementor shall complete Table 4, providing either a reference to the definition or values for all the field elements of the object.

For the &Capabilities field, one or more copies of Table 2 shall be created, documenting the Capability alloted to this predefined Domain. The &state field shall be assigned either tha value **ready** or **in-use**, depending on whether the &ProgramInvocations field is empty or not. The &aAssociation field should be empty.

For each predefined object that is subordinate to the Domain, a table corresponding to the object type shall be created and referenced in the &content field of Table 4.

This set may be empty.

Table 4 - Predefined Domain object

Domain Object	CBB	value or reference
&name		
reference to Definition		
&Capabilities		
&state		
&aAssociation		empty
&accessControl		
&sharable		
&ProgramInvocations		
&uploadsInProgress		set to 0
&NamedVariables	vnam	provide Table 8
&NamedVariableLists	vlis	provide Table 9
&NamedTypes	vnam	provide Table 10
&EventConditions		provide Table 14
&EventActions		provide Table 15
&EventEnrollments		provide Table 16
&EventCondtionLists	cspi	provide Table 17

25.2.14 &ProgramInvocations

The implementor shall supply a set of predefined Program Invocation objects. For each such object, the implementor shall complete Table 5, providing either a reference to the definition or values for all the field elements of the object.

This set may be empty. If this set is not empty, Table 4 shall not be empty.

Table 5 - Predefined Program Invocation object

Program Invocation Object	CBB	value or reference
&name		
reference to Definition		
&programInvocationState		
&Domains		
&accessControl		
&reusable		
&monitor		
&eventCondition		
&eventAction		
&eventEnrollment		
&executionArgument		
&control	csr	
&controlling-Program-Invocation	csr	
&controlled-Program-Invocations	csr	

25.2.15 &UnitControls

The implementor shall supply a set of predefined Unit Control objects. For each such object, the implementor shall complete Table 6, providing either a reference to the definition or values for all the field elements of the object.

This set may be empty. If this set is not empty, Table 4 shall not be empty.

Table 6 - Predefined Unit Control object

Unit Control Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&Domains		
&ProgramInvocations		

25.2.16 &UnnamedVariables

If the implementation is capable of supporting the **vadr** conformance building block, the implementor shall complete Table 7, supplying information about the composition of Unnamed Variables including the format and range of addresses supported (whether it is **numeric**, **symbolic**, or **unconstrained**) and the algorithm for associating a Type Description choice with an address.

Table 7 - Unnamed Variable objects

Unnamed Variables	Description
Address	
Type Description	

25.2.17 &NamedVariables

If the implementation is capable of supporting the **vnam** conformance buildling block, the implementor shall supply a set of predefined Named Variable objects. For each such object, the implementor shall complete Table 8, providing either a reference to the definition or values for all the field elements of the object.

This set may be empty.

Table 8 - Predefined Named Variable object

Named Variable Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&typeDescription		
&accessmethod		
&address	vadr	
&meaning	sem	

25.2.18 &NamedVariableLists

If the implementation is capable of supporting both the **vnam** and the **vlis** conformance buildling blocks, the implementor shall supply a set of predefined Named Variable List objects. For each such object, the implementor shall complete Table 9, providing either a reference to the definition or values for all the field elements of the object.

For each item in the &listOfVariables field, the implementor shall provide an identification of the Named or Unnamed Variable referenced, and the alternate access specification, if applicable.

This set may be empty.

Table 9 - Predefined Named Variable List object

Named Variable List Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&listOfVariables		
unnamedItem	vadr	
namedItem	vnam	
alternaeAccess	valt	

25.2.19 &NamedTypes

If the implementation is capable of supporting the **vnam** conformance buildling block, the implementor shall supply a set of predefined Named Type objects. For each such object, the implementor shall complete Table 10, providing either a reference to the definition or values for all the field elements of the object.

This set may be empty.

Table 10 - Predefined Named Type object

Named Type Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&typeDescription		
&meaning	sem	

25.2.20 &DataExchanges

The implementor shall supply a set of predefined Data Exchange objects. For each such object, the implementor shall complete Table 11, providing either a reference to the definition or values for all the field elements of the object.

For each such object, the &inUse field shall be initialised to false.

This set may be empty.

Table 11 - Predefined Data Exchange object

Data Exchange Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&request		
&response		
&linked		
&programInvocation		

25.2.21 &Semaphores

The implementor shall supply a set of predefined Semaphore objects. For each such object, the implementor shall complete Table 12, providing either a reference to the definition or values for all the field elements of the object. Depending on the value of &class, either &numberOfTokens or &Named-Tokens shall be filled out.

The fields &numberOfOwnedTokens, if present, shall be initialised to zero. The fields &Owners and &Requesters shall be initialised to an empty set.

This set may be empty. If this set is not empty, &EventConditions shall not be empty.

Table 12 - Predefined Semaphore object

Semaphore Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&class		
&numberOfTokens		
&Named-Tokens		
&eventCondition		

25.2.22 &OperatorStations

The implementor shall supply a set of predefined Operator Station objects. For each such object, the implementor shall complete Table 13, providing either a reference to the definition or values for all the field elements of the object.

The &inputBuffer field, if present, shall be initialised to an empty string. The &outputBuffers field shall be initialised to empty strings. The &state field shall be initialised to **idle**.

This set may be empty.

Table 13 - Predefined Operator Station object

Operator Station Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&stationType		

25.2.23 &EventConditions

The implementor shall supply a set of predefined Event Condition objects. For each such object, the implementor shall complete Table 14, providing either a reference to the definition or values for all the field elements of the object.

The &timeToActive field and the &timeToIdle field shall be initialised to **undefined**.

This set may be empty.

Table 14 - Predefined Event Condition object

Event Condition Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&ecClass		
&ecState		
&priority		
&severity		
&EventEnrollments		
&enabled		
&alarmSummaryReports		
&monitoredVariable		
&evaluationInterval		
&displayEnhancement	cspi	
&group-Priority-Override	cspi	
&ReferencingEventConditionLists	cspi	

25.2.24 &EventActions

The implementor shall supply a set of predefined Event Action objects. For each such object, the implementor shall complete Table 15, providing either a reference to the definition or values for all the field elements of the object.

The &timeToActive field and the &timeToIdle field shall be initialised to **undefined**.

This set may be empty.

Table 15 - Predefined Event Action object

Event Action Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&confirmedServiceRequest		
&Modifiers		
&EventEnrollments		

25.2.25 &EventEnrollments

The implementor shall supply a set of predefined Event Enrollments objects. For each such object, the implementor shall complete Table 16, providing either a reference to the definition or values for all the field elements of the object.

The &aAssociation field shall be initialised to empty. The &invokeID field, if present, shall be initialised to zero. The &notificationLost field, if present, shall be initialised to false. The &timeActiveAck field and the &timeIdleAck fields, if present, shall be initialised to **undefined**. The &ackState field, if present, shall be initialised to **acked**.

This set may be empty. If this set is not empty, the &EventConditions set shall not be empty.

Table 16 - Predefined Event Enrollment object

Event Enrollment Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&eeClass		
&eventCondition		
&ecTransitions		
&remainingDelay		
&eventAction		
&duration		
&clientApplication		
&aaRule		
&displayEnhancement	cspi	

25.2.26 &EventConditionLists

The implementor shall supply a set of predefined Event Condition List objects. For each such object, the implementor shall complete Table 17, providing either a reference to the definition or values for all the field elements of the object.

This set may be empty. If this set is not empty, the &EventConditions set shall not be empty.

Table 17 - Predefined Event Condition List object

Event Condition List Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&EventConditions		
&EventConditionLists	recl	
&ReferencingEventConditionLists	recl	

25.2.27 &Journals

The implementor shall supply a set of predefined Journal objects. For each such object, the implementor shall complete Table 18, providing either a reference to the definition or values for all the field elements of the object.

This set may be empty.

Table 18 - Predefined Journal object

Journal Object	CBB	value or reference
&name		
reference to Definition		
&accessControl		
&Entries		

For each entry in the &Entries field of this Journal object, the implementor shall complete Table 19, providing either a reference to the definition or values for all the field elements of the object.

The values assigned to the &entry field shall be unique within a given Journal.

This set may be empty.

Table 19 - Predefined Journal Entry object

Journal Entry Object	CBB	value or reference
&journal		
&entry		
&clientApplication		
&timeStamp		
&orderOfReceipt		
&informationType		
&textComment		
&eventTransitionRecord		
&JournalVariables		

25.2.28 &operation-State

If the implementation is capable of supporting the **csr** conformance building block, the implementor shall initialise this field to reflect the condition of the hardware underlying the VMD. The possible values are: **idle**, **loaded**, **ready**, **execution**, **motion-paused**, and **manualInterventionRequired**.

25.2.29 &safety-Interlocks-Violated

If the implementation is capable of supporting the **csr** conformance building block, the implementor shall initialise this field to reflect the condition of the hardware underlying the VMD.

25.2.30 &any-Resource-Power-On

If the implementation is capable of supporting the **csr** conformance building block, the implementor shall initialise this field to reflect the condition of the hardware underlying the VMD.

25.2.31 &local-Control

If the implementation is capable of supporting the **csr** conformance building block, the implementor shall initialise this field to reflect the condition of the hardware underlying the VMD.

25.2.32 &selected-Program-Invocation

If the implementation is capable of supporting the **csr** conformance building block, the implementor shall supply the initial value of the selected-Program-Invocation. This value may be null.

25.3 CIS Part Two: Service and Parameter CBBs

The implementor shall provide the information in the following tables about the service and parameter CBBs supported by the implementation, indicating whether the implementation fulfils the server requirements, the client requirements, or both when operating in the abstract syntax defined in this part of ISO 9506. The terms client, server, requesting, and responding are defined in clause 5 of ISO 9506-1. The server and client requirements for each CBB are described in ISO 9506-1, clause 25.

25.3.1 Environment and General Management Services

For this subclause only, support shall be indicated in Table 20 in terms of whether an implementation can support the requestor role, the responder role or both.

Table 20 - Environment & General Management services

Environment & General Management	Requester	Responder
Initiate		
Conclude		
Cancel		

In Table 21, the implementor shall indicate:

1) Whether the **csr**, **csnc**, **csplc**, and **cspi** parameter CBBs are supported.

2) Local Detail Calling/Called. The implementor shall state the detail used within both the Local Detail Calling and Local Detail Called parameters used in the Initiate service, if they are provided as part of an implementation. The semantics of any values used shall be provided.

3) level of support for time: The implementor shall state whether date and time of day can be supported in the presence of a system clock. The implementor shall also state whether or not the Time Sequence Identifier is supported.

4) granularity of time in milliseconds: The implementor shall state the smallest unit of time, in milliseconds, that can be used to time resolution. This value will only have meaning if date and time of day are supported.

Table 21 - Environment & General Management parameters

General Management parameters	Value
csr	
csnc	
csplc	
cspi	
Local Detail	
Support for time	
Granularity of time (ms)	

25.3.2 Access Control Services

The implementor shall indicate in Table 22 whether the implementation supports the server role, the client role, or both for the following services.

Table 22 - Access Control services

Access Control	Server	Client
Define Access Control List		
Get Access Control List Attributes		
Report Access Controlled Objects		
Delete Access Control List		
Change Access Control		

The implementor shall indicate in Table 23 whether the **aco** CBB is supported.

Table 23 - Access Control parameter

Access Control parameter	
aco	

25.3.3 VMD Support Services

The implementor shall indicate in Table 24 whether the implementation supports the server role, the client role, or both for the following services.

Table 24 - VMD Support services

VMD Support	Server	Client
Status		
Unsolicited Status		
Get Name List		
Identify		
Rename		
Get Capability List		
VMD Stop		
VMD Reset		

An implementation that indicates responder support for the Initiate service is required to support the server role for the Identify service.

In Table 25 the implementator shall indicate:

1) Local Detail (parameter in the Status and UnsolicitedStatus services): The implementor shall state what the local detail is, and how this parameter can be parsed within a bit string (both the syntax and semantics of the character string shall be provided).

2) Method for Extended Derivation of Status Information: The implementor shall state the method used for extended derivation of status information, if any, as described in 10.3.

Table 25 - VMD Support parameters

VMD Support parameters	Value
Local Detail	
Extended Derivation	

25.3.4 Domain Management Services

The implementor shall indicate in Table 26 whether the implementation supports the server role, the client role, or both for the following services.

Table 26 - Domain Management services

Domain Management Support	Server	Client
Initiate Download Sequence		
Download Segment		
Terminate Download Sequence		
Initiate Upload Sequence		
Upload Segment		
Terminate Download Sequence		
Request Domain Download		
Request Domain Upload		
Load Domain Content		
Store Domain Content		
Delete Domain		
Get Domain Attributes		

In Table 27, the implementor shall indicate:

1) whether the implementation can support the **tpy** parameter CBB.

2) Load Data Format. The implementor shall state semantics and further syntax definitions for the octet string in the Load Data parameter of the DownloadSegment and UploadSegment services.

3) If the EXTERNAL or EMBEDDED PDV choice of these parameters is supported, the implementor shall state the abstract syntax names which are supported.

4) Maximum Number of Upload State Machines. The implementor shall supply the maximum number of Upload State Machines which may be simultaneously invoked on a single Domain.

Table 27 - Domain Management parameters

Domain parameters	Value
tpy	
Load Data - octet strings	
Load Data - abstract syntax	
Max Upload State Machines	

25.3.5 Program Invocation Management Services

The implementor shall indicate in Table 28 whether the implementation supports the server role, the client role, or both for the following services.

Table 28 - Program Invocation Management services

Program Invocation Management Support	Server	Client
Create Program Invocation		
Delete Program Invocation		
Start		
Stop		
Resume		
Reset		
Kill		
Get Program Invocation Attributes		
Select		
Alter Program Invocation Attributes		
Reconfigure Program Invocation		

In Table 29, the implementor shall indicate:

1) Execution Argument (parameter in the Start and Reset services): The implementor shall state the maximum number of characters supported for the character string in the Execution Argument parameter.

2) The parse rules of the Execution Argument character string.

3) If the EXTERNAL or EMBEDDED PDV choice for this parameter is supported, the implementor shall state the abstract syntax names which are supported.

4) The format of the notation for the &programLocation field, if present

5) Whether the Step Mode of operation is supported

Table 29 - Program Invocation Management parameters

Program Invocation parameters	Value
Execution Argument max size	
Execution Argument parse rules	
Execution Argument abstract syntax	
&programLocation	
Step Mode	

25.3.6 Unit Control Services

The implementor shall indicate in Table 30 whether the implementation supports the server role, the client role, or both for the following services.

Table 30 - Unit Control services

Unit Control Management Support	Server	Client
Initiate Unit Control Load		
Unit Control Load Segment		
Unit Control Upload		
Start Unit Control		
Stop Unit Control		
Create Unit Control		
Add To Unit Control		
Remove From Unit Control		
Get Unit Control Attributes		
Load Unit Control From File		
Store Unit Control To File		
Delete Unit Control		

25.3.7 Variable Access Services

The implementor shall indicate in Table 31 whether the implementation supports the server role, the client role, or both for the following services.