

INTERNATIONAL STANDARD

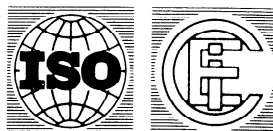
ISO/IEC
9506-1

First edition
1990-10-15

Industrial automation systems — Manufacturing Message Specification —

Part 1: Service definition

*Systèmes d'automatisation industrielle — Spécification de messagerie industrielle —
Partie 1: Définition de service*



Reference number
ISO 9506-1 : 1990 (E)

AVA Documents
Document Sharing Hub

Contents

Page

1	Scope	1
2	Normative references	1
3	Definitions	2
3.1	Reference Model definitions	2
3.2	Service Convention definitions	3
3.3	Abstract Syntax Notation definitions	3
3.4	Other definitions	4
4	Abbreviations	7
5	Conventions	8
5.1	Base of Numeric Values	8
5.2	Service Parameter Description	8
5.3	Invocation Identifier on Service Primitives	10
5.4	List Of Modifier on Service Primitives	10
5.5	Addressing in MMS	10
5.6	Service Conventions	10
5.7	Calling and Called MMS-user	11
5.8	Sending and Receiving MMS-user and MPPM	11
5.9	Requesting and Responding MMS-user	11
5.10	Client and Server of a Service	11
5.11	Object modelling	12
5.12	References to Objects	13
5.13	Parameter Types	14
6	MMS in the OSI Environment	14
6.1	Information Processing Tasks and Real Systems	14
6.2	Application Processes	15
6.3	Interaction of Application Processes	15
6.4	Interaction of Application Processes in OSI	15

© ISO/IEC 1990

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office ● Case postale 56 ● CH-1211 Genève 20 ● Switzerland
Printed in Switzerland

6.5	Structure of Application Entities	16
6.6	Addressing of Application Entities	16
6.7	Application Context	16
6.8	Presentation Context, Abstract Syntaxes, and Transfer Syntaxes	17
7	The Virtual Manufacturing Device	17
7.1	Introduction	17
7.2	The Structure of a VMD	19
7.3	Specification of Named Objects	24
7.4	Object Name Structure	27
7.5	Services on the VMD	28
8	Environment And General Management Services	28
8.1	Introduction	28
8.2	Initiate Service	30
8.3	Conclude Service	35
8.4	Abort Service	37
8.5	Cancel Service	38
8.6	Reject Service	40
9	VMD Support Services	45
9.1	Introduction	45
9.2	Status Service	45
9.3	UnsolicitedStatus Service	46
9.4	GetNameList Service	47
9.5	Identify Service	50
9.6	Rename Service	51
9.7	GetCapabilityList Service	53
10	Domain Management Services	55
10.1	The Domain Object	55
10.2	InitiateDownloadSequence Service	61
10.3	DownloadSegment Service	62
10.4	TerminateDownloadSequence Service	64
10.5	InitiateUploadSequence Service	65
10.6	UploadSegment Service	66
10.7	TerminateUploadSequence Service	68
10.8	RequestDomainDownload Service	69
10.9	RequestDomainUpload Service	70
10.10	LoadDomainContent Service	71
10.11	StoreDomainContent Service	74
10.12	DeleteDomain Service	77
10.13	GctDomainAttributes Service	78
11	Program Invocation Management Services	79
11.1	The Program Invocation Object	80
11.2	CreateProgramInvocation Service	86
11.3	DeleteProgramInvocation Service	89

11.4	Start Service	90
11.5	Stop Service	92
11.6	Resume Service	93
11.7	Reset Service	94
11.8	Kill Service	96
11.9	GetProgramInvocationAttributes Service	97
12	Variable Access Services	98
12.1	The MMS Variable Access Model	99
12.2	Specification of Types	108
12.3	Specification of Alternate Access	112
12.4	Specification of Data Values	116
12.5	Specification of Access to Variables	119
12.6	Read Service	124
12.7	Write Service	125
12.8	InformationReport Service	127
12.9	GetVariableAccessAttributes Service	128
12.10	DefineNamedVariable Service	129
12.11	DefineScatteredAccess Service	131
12.12	GetScatteredAccessAttributes Service	133
12.13	DeleteVariableAccess Service	134
12.14	DefineNamedVariableList Service	136
12.15	GetNamedVariableListAttributes Service	139
12.16	DeleteNamedVariableList Service	141
12.17	DefineNamedType Service	143
12.18	GetNamedAttributes Service	144
12.19	DeleteNamedType Service	145
12.20	Conformance	147
12.21	Guidance To Implementors	149
13	Semaphore Management Services	150
13.1	The Semaphore Management Model	151
13.2	TakeControl Service	158
13.3	RelinquishControl Service	161
13.4	DefineSemaphore Service	162
13.5	DeleteSemaphore Service	164
13.6	ReportSemaphoreStatus Service	165
13.7	ReportPoolSemaphoreStatus Service	166
13.8	ReportSemaphoreEntryStatus Service	168
13.9	AttachToSemaphore Modifier	170
13.10	Conformance	173
14	Operator Communication Services	173
14.1	The Operator Communications Model	174
14.2	Input Service	176
14.3	Output Service	178

15	Event Management Services	179
15.1	The Event Management Model	180
15.2	DefineEventCondition Service	198
15.3	DeleteEventCondition Service	200
15.4	GetEventConditionAttributes Service	202
15.5	ReportEventConditionStatus Service	204
15.6	AlterEventConditionMonitoring Service	205
15.7	TriggerEvent Service	207
15.8	DefineEventAction Service	209
15.9	DeleteEventAction Service	210
15.10	GetEventActionAttributes Service	212
15.11	ReportEventActionStatus Service	213
15.12	DefineEventEnrollment Service	214
15.13	DeleteEventEnrollment Service	218
15.14	GetEventEnrollmentAttributes Service	221
15.15	ReportEventEnrollmentStatus Service	226
15.16	AlterEventEnrollment Service	227
15.17	EventNotification Service	230
15.18	AcknowledgeEventNotification Service	232
15.19	GetAlarmSummary Service	234
15.20	GetAlarmEnrollmentSummary Service	237
15.21	Attach To Event Condition Modifier	241
15.22	Event Management State Diagrams	244
15.23	Conformance Requirements Unique to Event Management	249
16	Journal Management Services	250
16.1	The Journal Management Model	250
16.2	ReadJournal Service	253
16.3	WriteJournal Service	260
16.4	InitializeJournal Service	263
16.5	ReportJournalStatus Service	265
16.6	CreateJournal Service	266
16.7	DeleteJournal Service	267
16.8	Conformance Requirements Unique to Journals	268
17	Errors	268
17.1	Error Type	269
17.2	Description of structure of generic error type:	269
17.3	Additional Code	276
17.4	Additional Detail	276
17.5	Modifier Position	276
18	MMS Standardized Names	276
18.1	Introduction	276

ISO/IEC 9506-1 : 1990 (E)

18.2	Unique Name Assignment Mechanism	277
18.3	MMS Standardized Names	277
19	Conformance	278
19.1	Introduction	278
19.2	Conformance Building Blocks (CBBs)	278
19.3	Static Conformance Requirements	281
19.4	Calling MMS-user Conformance Requirements	281
19.5	Called MMS-User Conformance Requirements	282
19.6	Server Conformance Requirements	282
19.7	Client Conformance Requirements	283
19.8	Parameter CBB Conformance Requirements	284
19.9	Dynamic Conformance	284
Annexes		
A	Requirements for Companion Standards (normative)	285
B	File Access Service (normative)	297
C	File Management Services (informative)	299
Index		312

Foreword

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

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

International Standard ISO/IEC 9506-1 was prepared by Technical Committee ISO/IEC TC 184, *Industrial automation systems*, Sub-Committee SC 5, *System integration and communication*.

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

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

Introduction

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

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

- a) applicability of the service to the device;
- b) the complexity of services and requirements;
- c) the complexity of provision of a particular class of service via the network versus the complexity of the device.

Security considerations

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

- a) access control;
- b) authentication;
- c) non-repudiation.

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

Complexity of services and requirements

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

Keywords

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

General

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

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

- a) from different manufacturers;
- b) under different managements;
- c) of different levels of complexity;
- d) of different ages.

Purpose

The purpose of this part of ISO/IEC 9506 is to define the service provided by the Manufacturing Message Specification. The MMS Service is provided by the Manufacturing Message Specification Protocol making use of services available from the Association Control Service Element (ASCE) and the Presentation layer, as defined in ISO 8649 and ISO 8822, respectively.

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

ISO/IEC 9506-2 specifies the protocol that supports the Manufacturing Message Specification.

This page intentionally left blank

Industrial automation systems - Manufacturing Message Specification - Part 1: Service definition

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. This environment is referred to in ISO/IEC 9506 as the manufacturing environment. This part of ISO/IEC 9506 does not specify a complete set of services for remote programming of devices, although provision of such a set of services may be the subject of future standardization efforts.

This part of ISO/IEC 9506 defines the Manufacturing Message Specification within the OSI application layer in terms of

- a) an abstract model defining the interaction between users of the service;
- b) the externally visible functionality of implementations conforming to ISO/IEC 9506, in the form of procedural requirements associated with the execution of service requests;
- c) the primitive actions and events of the service;
- d) the parameter data associated with each primitive action and event;
- e) the relationship between, and the valid sequences of, these actions and events.

The service defined in this part of ISO/IEC 9506 is that which is provided by the Manufacturing Message Specification protocol. The service may be used by other application layer service elements or by other elements of the application process.

This part of ISO/IEC 9506 does not specify individual implementations or products, nor does it constrain the implementation of entities and interfaces within a computer system. This part of ISO/IEC 9506 specifies the externally visible functionality of implementations together with conformance requirements for such functionality.

2 Normative References

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

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

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

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

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

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

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

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

ISO/IEC 9506-1: 1990(E)

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

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

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

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

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

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

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

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

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

ISO/IEC 9506-2: 1990, *Industrial automation systems — Manufacturing Message Specification — Part 2: Protocol specification.*

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

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

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

3 Definitions

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

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

3.1 Reference Model definitions

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

- a) application-entity;
- b) application-process;
- c) application service element;
- d) open system;
- e) (N)-protocol;
- f) (N)-protocol-data-unit;
- g) (N)-service-access-point;
- h) (N)-layer;
- i) system;

1) To be published.

- j) (N)-user-data.

3.2 Service Convention definitions

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

- a) confirm;
- b) indication;
- c) primitive;
- d) request;
- e) response;
- f) service primitive;
- g) service provider;
- h) service user.

3.3 Abstract Syntax Notation definitions

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

- 1) value;
- 2) type;
- 3) simple type;
- 4) structure type;
- 5) component type;
- 6) tag;
- 7) tagging;
- 8) type (or value) reference name;
- 9) character string type;
- 10) boolean type;
- 11) true;
- 12) false;
- 13) integer type;
- 14) bitstring type;
- 15) octetstring type;
- 16) null type;
- 17) sequence type;
- 18) sequence-of type;
- 19) tagged type;

ISO/IEC 9506-1: 1990(E)

- 20) choice type;
- 21) selection type;
- 22) real type;
- 23) object identifier type;
- 24) module;
- 25) production;
- 26) ASN.1 encoding rules;
- 27) ASN.1 character set;
- 28) external type.

3.4 Other definitions

This part of ISO/IEC 9506 makes use of the following terms:

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 with respect to 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 conformance building block (CBB):

An atomic unit used to describe MMS conformance requirements.

3.4.4 Called MMS-user:

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

3.4.5 Calling MMS-user:

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

3.4.6 Client:

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

3.4.7 data:

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

3.4.8 domain:

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

3.4.9 Domain-specific:

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

3.4.10 download:

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

3.4.11 event management:

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

3.4.12 file:

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

3.4.13 file operation:

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

3.4.14 filestore:

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

3.4.15 information:

The combination of data and the meaning that it conveys.

3.4.16 journal:

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

3.4.17 local matter:

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

3.4.18 Manufacturing Message Protocol Machine (MMPM):

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

3.4.19 MMS-environment:

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

3.4.20 MMS-provider:

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

3.4.21 MMS-user:

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

3.4.22 monitored event:

A detected change in the state of an event condition.

3.4.23 network-triggered event:

An event which occurs due to an explicit solicitation by a client.

3.4.24 operator station:

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

3.4.25 predefined object:

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

3.4.26 program invocation:

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

3.4.27 Receiving MMPM:

The MMPM that receives an MMS PDU.

3.4.28 Receiving MMS-user:

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

3.4.29 remote device control and monitoring:

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

3.4.30 semaphore:

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

3.4.31 semaphore management:

The control of semaphores.

3.4.32 Server:

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

3.4.33 Sending MMPM:

The MMPM that sends an MMS PDU.

3.4.34 Sending MMS-user:

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

3.4.35 standardized object:

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

3.4.36 type:

An abstract description of a set of values which may be conveyed by the value of a variable.

3.4.37 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.

3.4.38 variable:

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

3.4.39 variable access:

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

3.4.40 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

ISO/IEC 9506-1: 1990(E)

AA	: application association
ACSE	: Association Control Service Element
AE	: application entity
AP	: application process
ASE	: application service element
ASN.1	: Abstract Syntax Notation One
CBB	: conformance building block
FRSM	: file read state machine
FTAM	: File Transfer, Access and Management
MMPM	: Manufacturing Message Protocol Machine
MMS	: Manufacturing Message Specification
NC	: Numerical Control
OSI	: Open Systems Interconnection
PC	: Programmable Controller
PDU	: protocol data unit
PSAP	: presentation service access point
SAP	: service access point
SDU	: service data unit
ULSM	: upload state machine
VMD	: Virtual Manufacturing Device
VT	: Virtual Terminal

5 Conventions

5.1 Base of Numeric Values

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

5.2 Service Parameter Description

This part of ISO/IEC 9506 uses a tabular format to describe the component parameters of the MMS service primitives. Each table consists of up to six columns, containing the name of the service parameter, a column each for the request ("Req"), indication ("Ind"), response ("Rsp"), and confirm ("Cnf") primitives, and a column for conformance building block specification ("CBB"). The "Rsp" and "Cnf" columns are absent when the service is not a confirmed service.

5.2.1 Companion Standard Service Parameters

Some MMS service procedures which allow extensions to be supplied by Companion Standards. Such service procedures allow the specification of extra parameters in either the request primitive or in the response (+) primitive or both. This is indicated in the tables by the appearance of "COMP" following the "Argument" or the "Result(+)" entry of the service table respectively. Some service procedures allow the specification of additional parameters by the Companion Standards directly in an Argument list. Such occurrences will be indicated by the "COMP" in the columns describing the service primitives.

5.2.2 Service Table Structure

For those tables that require support of particular parameter conformance building blocks, the required conformance building blocks are enumerated on the first line in the table. In the remainder of each table, one parameter (or part of it) is listed on each horizontal line. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive specified in the vertical column:

- M - parameter is mandatory for the primitive
- U - parameter is a user option, and may or may not be provided depending on dynamic usage by the MMS-user
- C - parameter is conditional upon other parameters or the environment of the MMS-user
- (blank) parameter is never present
- COMP - parameter is for definition in MMS Companion Standards. Such parameters shall not be used other than as defined in a Companion Standard (and shall be omitted in the abstract syntax defined in Clause 19 of 9506-2). Annex C provides additional detail on the use of Companion Standards.
- S - parameter is a selection from a collection of two or more possible parameters. The parameters that make up this collection are indicated in the table as follows:
 - a) each parameter in the collection is specified with the code "S";
 - b) the name of each parameter in the collection is at the same indentation from the beginning of the parameter column in the table;
 - c) Either
 - 1) each parameter is at the leftmost (outer) indentation in the table; or
 - 2) each parameter is part of the same parameter group. A parameter group is a collection of parameters where each group member has a common parent parameter. The parent parameter for any group member is the first parameter above the member that is not indented as far as that member. In the example below, ParameterA and ParameterB form a parameter group:

```

ParameterX
  ParameterA
  ParameterB
ParameterY
  ParameterC

```

Informally, for parameters involved in a selection, the indentation in the services tables signifies which parameters are involved in a selection. All parameters at the same level of indentation that are under a common "higher level" parameter are a part of the same selection.

The code "(=)" following one of the codes M, U, C or S indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table. (For instance, an "M(=)" code in the indication service primitive column and an "M" in the request service primitive column means that the parameter in the indication primitive is semantically equivalent to that in the request primitive.)

Some parameters may contain subparameters. Subparameters are indicated by labelling of the parameter as M, U or C, and indenting all subparameters under the parameter. Presence of subparameters is always dependent on presence of the parameter that they appear under (for example, an optional parameter may have subparameters; if the parameter is not supplied, then no subparameters may be supplied).

The CBB column is used to indicate that usage of the parameter is dependent on support of conformance building blocks, other than that containing the service. If no entry exists in the CBB column, then there is no dependency on other conformance building blocks. If an entry does exist, then the parameter is available (and permitted for use by this Standard) if and only if the named conformance building block is supported and negotiated for use.

Some service parameters are named using a "List Of . . ." convention. Unless otherwise noted, all parameters whose names begin with "List Of . . ." specify a list of zero or more of the item specified after the "List Of" keyword phrase. (This type of parameter corresponds with the sequence-of ASN.1 type in ISO/IEC 9506-2.)

The descriptions of parameters in this part of ISO/IEC 9506 make reference to types, in order to describe the allowable values for such parameters. The types referenced may either be types defined in ISO 8824 (Abstract Syntax Notation One), or may be defined in ISO/IEC 9506-2.

5.3 Invocation Identifier on Service Primitives

For services identified in the ConfirmedServiceRequest production in 7.5.2 of ISO/IEC 9506-2, each MMS service primitive contains an "Invoke ID" parameter, which is mandatory in the request, indication, response, and confirm primitives. The value in the indication, response, and confirm primitives is semantically equivalent to that in the request primitive. This parameter serves to identify unambiguously the service invocation from an MMS-user on an application association. This parameter is not explicitly shown in the service primitive tables, nor is it explained separately for each service.

5.4 List Of Modifier on Service Primitives

Every confirmed MMS service contains a "List Of Modifier" parameter, which is a user option in the request and indication primitives. The value in the indication primitive is semantically equivalent to that in the request primitive. This parameter serves to specify a list of one or more service state machine modifiers which add a condition which must be satisfied for the execution of the service request to begin. This parameter is not explicitly shown in the service primitive tables, nor is it explained separately for each service.

MMS defines two modifiers: the AttachToSemaphore modifier and the AttachToEventCondition modifier, which are described in clauses 13 and 15, respectively.

The effect of the modifier on the state machine for execution of a confirmed MMS service is described in clause 7.

5.5 Addressing in MMS

The MMS Standard does not provide the means for naming and addressing of a peer MMS-user or peer Manufacturing Message Protocol Machine (MMPM). Within OSI, the identification and addressing of peer application entities is carried out through the use of ACSE services defined in ISO 8649. After such association of peers has succeeded, all of the MMS PDUs flow between these peers over the established presentation connection. It is therefore not necessary for MMS to carry addressing information. Additional information on naming and addressing may be found in 6.6.

5.6 Service Conventions

This part of ISO/IEC 9506 uses the descriptive conventions contained in the OSI Service Conventions (ISO/TR 8509). The OSI Service Conventions define the interactions between the MMS-user and the MMS-provider. Information is passed between the MMS-user and the MMS-provider by service primitives, which may convey parameters. The following apply to the use of this model:

- a) ISO/TR 8509 defines a model for the service provided by a layer of the OSI Reference Model. The MMS service does not correspond to such a layer (it describes a part of the application layer) but the model used is identical in all other respects;

- b) at any instant in time, an application entity has multiple service requests outstanding, each proceeding independently of the others.

NOTE – It should be noted that the MMS-user/MMS-provider distinction is an abstraction, and may not necessarily correspond to the realization of MMS in any particular system. Clauses 6 and 7 provide further details on the usage of abstract models.

5.7 Calling and Called MMS-user

This part of ISO/IEC 9506 makes use of the terms Calling and Called MMS-users. 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.

5.8 Sending and Receiving MMS-user and MMPM

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

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

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

5.9 Requesting and Responding MMS-user

This part of ISO/IEC 9506 makes use of the terms Requesting and Responding MMS-users. 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.

5.10 Client and Server of a Service

This part of ISO/IEC 9506 makes use of the terms Client and Server in order to describe the model of the MMS VMD (The VMD is described in clause 7). The Server is defined as the peer communicating entity which behaves as a VMD for a particular service request instance. The Client is the peer communicating entity which makes use of the VMD for some particular purpose via a service request instance. The VMD model is primarily useful in describing the actions of the Server, and thus in describing the commands and responses that a Client may use. A real end system may adopt the Client role, or the Server role, or both during the lifetime of an application association. Use of MMS in the OSI environment is further described in clause 6.

Figure 1 depicts the relationships of the client and server of a service, the requesting and responding MMS-user, and the sending and receiving MMS-user and MMPM.

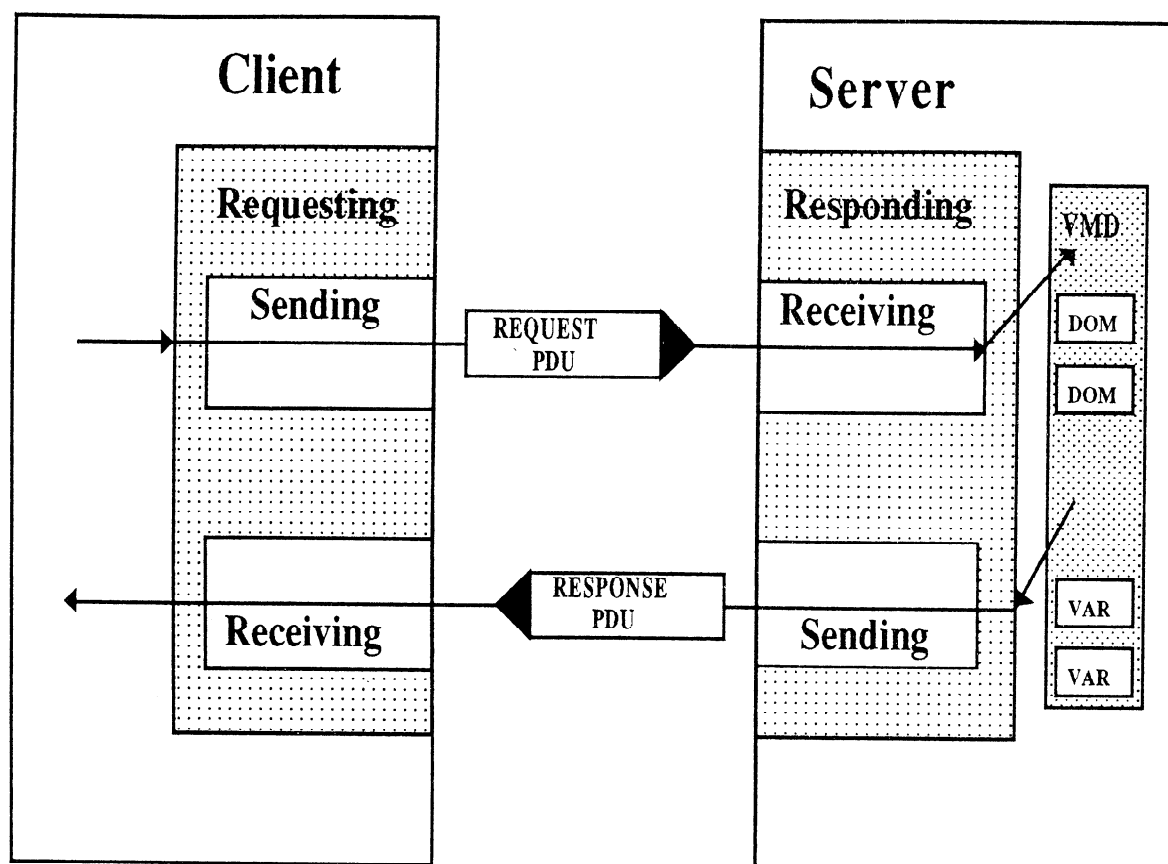


Figure 1 – Relationships of Client and Server, Requesting and Responding MMS-user, and Sending and Receiving MMPM

5.11 Object modelling

This part of ISO/IEC 9506 makes use of a technique of abstract object modelling in order to fully describe the MMS device model and the MMS service procedures. In this modelling technique, abstract objects, the characteristics of such objects, and operations on those objects are described. The objects defined are abstract and aid in the understanding of the intent of MMS service procedures and their effects. In implementing MMS, a real system maps the concepts described in the model to the real device. Hence, as viewed externally, a device that conforms to this part of ISO/IEC 9506 exhibits the characteristics described in the object modelling technique, but the mechanisms for realization of this view are not defined by this part of ISO/IEC 9506.

MMS defines a number of classes of objects. Each object is an instance of a class, and constitutes an abstract entity which exhibits certain characteristics and may be affected by certain MMS services and operations. Each class is given a name, by which it may be referenced.

Each class is characterized by a number of attribute types, which serve to describe some externally visible feature(s) of all objects of this class. Each instance of a class (object) has the same set of attribute types, but has its own set of attribute values. The values of these attributes are defined by this part of ISO/IEC 9506 or may be established by MMS services, and hence their effect on the device may be modeled by a change in one or more attribute values of an object (or objects).

Each object must be uniquely identified among all instances of the same class. For this purpose, one or more of the object's attribute values, as a combination, must be unique. (For example, many objects have an attribute type called "object name", which is different for each object of the same class.) In MMS, each attribute which is a part of this combination of attributes which make the object unique is identified as a "key attribute".

Finally, some objects contain attributes which are conditional, in the sense that they are relevant to the object if and only if certain conditions hold true. MMS expresses such attributes through the use of a "constraint", which specifies a condition. Attributes that are subject to a constraint are considered to be object attributes for an object if and only if the corresponding constraint is satisfied for that object.

In MMS, classes are syntactically defined as a set of objects as follows:

```
Object: (name of class)
  Key Attribute: (name of attribute type (values))
  .
  .
  .
  Key Attribute: (name of attribute type (values))
  Attribute: (name of attribute type (values))
  .
  .
  .
  Attribute: (name of attribute type (values))
  Constraint: (constraint expression)
    Attribute: (name of attribute type (values))
    Attribute: (name of attribute type (values))
    Attribute: (name of attribute type (values))
```

By convention, each object definition begins with an object declaration and the name of the object. Immediately following, and indented, one or more Key Attributes are named. Next, zero or more Attributes are named. Note that constraints may be expressed anywhere within the attributes, with the convention that all attributes subject to the constraint are indented underneath it. The first attribute definition that is not indented ends the list of attributes that are subject to the constraint.

NOTE — For convenience, attributes may be indented under other attributes in order to show a hierarchy of nesting of such attributes.

5.12 References to Objects

Some objects contain attributes which make reference to other objects. Such attributes, called reference attributes, provide a mechanism to create a linkage from one object to another. The values used to represent such attributes in a real system is a local matter, and such attributes may not be directly modified or examined. Many MMS services provide the capability to determine the identity of an object referenced by such a linkage, however, through the use of such an indirect reference.

NOTE — Reference attributes are similar to "pointer" types available in many programming languages. As an example of the operation of reference attributes, the MMS Rename service may be used to change the Object Name attribute of a referenced object, while having no effect on the value of the reference attribute.

Reference attributes may take on the value UNDEFINED when the object referenced by the link is deleted. In this case, the consequences of this value and its use on the behaviour on other objects is specified in the relevant object and service descriptions. Once a reference attribute becomes UNDEFINED, creation of a object of the same type as the deleted (and referenced object) with the same attribute values as that object which was deleted does not change the reference attribute (it retains the value UNDEFINED).

5.13 Parameter Types

Types for parameters defined in this part ISO/IEC 9506 make use of types defined in ISO 8824, clause 3. Additionally, complex types are used, which are constructed from the ISO 8824 primitive types and subsequently named to allow them to be referenced.

6 MMS in the OSI Environment

This clause serves to describe the relationship between MMS and the OSI environment, and relates OSI terms and concepts to the manufacturing environment. Clause 7 describes the specific model of the MMS device within this environment.

The development of standards for the interconnection of manufacturing monitoring and control devices is assisted by the use of abstract models. To specify the externally perceived behaviour of interconnected manufacturing devices, each manufacturing device is represented by a functionally equivalent abstract model of the device called a Virtual Manufacturing Device (VMD) (see clause 7). The VMD model, along with the extensions to this model which are provided by clauses 8 to 16, describes the externally visible aspects of these devices.

In order to accomplish this, however, it is necessary to describe both the internal and external behaviour of these manufacturing devices. Only the external behaviour of the devices is retained as the standard of behaviour of a VMD. The description of the internal behaviour of such devices is provided in the model only to support the definition of the externally perceived aspects. Any manufacturing device which behaves externally as a VMD can be considered to be in conformance with ISO/IEC 9506.

NOTE 1 – The reader not familiar with the technique of abstract modelling is cautioned that the concepts introduced in the description of the VMD constitute an abstraction despite a similar appearance to concepts commonly found in real devices. Therefore, the VMD model is not a specification for an implementation.

Additional information to supplement this clause may be found in the OSI Reference Model (ISO 7498-1 and ISO 7498-3), the OSI Application Layer Structure (ISO 9545), and the Service Definition for the Association Control Service Element (ISO 8649). References may be found in this part of ISO/IEC 9506, clause 2. This information may be particularly useful in understanding the relationship between AEs, APs, and their respective invocations.

NOTE 2 – This part of ISO/IEC 9506 specifies MMS operation in the OSI environment only. Operation in other environments and architectures is beyond the scope of this part of ISO/IEC 9506, but may be standardized elsewhere.

6.1 Information Processing Tasks and Real Systems

The control and monitoring of a manufacturing process is a distributed information processing task. In order to carry out successfully this task, inter-operation of a number of real open systems is required. In OSI, a real open system is defined as a set of computers and associated software (including peripherals, terminals, human operators, physical processes, etc.) that complies with the requirements of OSI standards in its communications with other real systems.

In OSI, the inter-operation of real open systems is modelled in terms of the interactions between Application Processes (APs) in these systems. The distributed task of control and monitoring of a manufacturing process requires the co-operation of two or more Application Processes.

6.2 Application Processes

An Application Process (AP) is an element within a real open system which takes part in the execution of one or more information processing tasks. It is an abstract representation of those aspects of a real open system which are specific to the performance of those tasks. APs generally have requirements above and beyond any requirements to communicate with other APs. In the manufacturing environment, an AP represents a real system participating in the overall control/monitoring distributed task.

6.3 Interaction of Application Processes

Several requirements must be met in order to allow co-operative operation of APs. First, they must share sufficient information to enable them to interact and carry out processing functions in a compatible and co-operative manner. In the manufacturing environment, the term "universe of discourse" is used to name a model of those aspects of the "world" which are pertinent to the objectives of a manufacturing control and monitoring task of an AP. In order to allow successful interworking between APs, they must share a universe of discourse (i.e. they must have a common understanding of the manufacturing environment).

It is convenient to describe the common model of the manufacturing environment, or universe of discourse, in terms of a set of abstract "objects". Examples of objects in the manufacturing environment include variables, programs, and semaphores. The model describes the characteristics of objects and relationships between them. The characteristics may include the properties of those objects (either static or dynamic), and these properties are expressed as a set of rules and constraints about the behaviour of these objects within the model of the manufacturing environment. An example of a property of an object is the characteristic of a program that it is either running or stopped.

A universe of discourse can be described formally by a "conceptual schema". The conceptual schema for the manufacturing universe of discourse is described by the models provided by MMS. The second requirement for successful interworking of two APs is that they share a common model of the objects (such as variables, programs, and semaphores) in the universe of discourse. In OSI, this is called a "shared conceptual schema". This requirement is met in a manufacturing control and monitoring application through the sharing of the models in MMS for the manufacturing universe of discourse.

When two APs co-operate, their behaviour is determined partly by these models (the shared conceptual schema) and partly by their past interactions. The past interactions are modelled by the state of the objects in the universe of discourse. As an example, the co-operative behaviour of two APs may be partly determined by the Program Invocation model of MMS and partly by the state of Program Invocation objects in the APs. The shared information about the state of objects is called an information base.

6.4 Interaction of Application Processes in OSI

The activity of a given AP in a specific information processing task is supported by an application-process-invocation. At any time, an Application Process may have zero, one or more application-process-invocations. While the Application Process describes a specific set of information processing functions in a particular real system, the application-process-invocation (AP-invocation) describes an instance of the Application Process in a real system for a particular occasion of information processing. (Hence, an AP may describe control of a robot arm in a specific real system, while an AP-invocation describes the control of a robot arm in a specific real system on some occasion of control for assembly of some particular part).

Thus, co-operation between APs for the performance of a given information processing task takes place through interacting AP-invocations. When APs interact using OSI communication capabilities, these communication capabilities are in turn modelled by one or more Application Entities (AE). An AE is an active element in the Application Layer of an open system, and it represents those parts of an Application Process involved in OSI communications. Each AE represents exactly one AP.

Like APs, the activity of a given AE is supported by an application-entity-invocation. An AE-invocation performs the functions of an AE for a particular occasion of communication. At any time, an AP-invocation may be represented by zero or more AE-invocations for each AE associated with the AP. Hence, the interactions of AP-invocations for a particular occasion of communication is represented in OSI by a set of corresponding AE-invocations.

Because an AE describes only part of the operation of an AP (namely, that involving OSI communications), resources in AP-invocations may exist beyond the lifetime of (and may be independent of) an AE-invocation.

NOTE – The relationship between an AE and the VMD is defined in clause 7.

6.5 Structure of Application Entities

The AE represents a set of communication capabilities of an AP. These capabilities are defined by a set of Application Service Elements (ASEs). An ASE is a coherent set of integrated functions that provides a capability for communications for some particular purpose. MMS is modelled as an ASE for the purpose of communications with manufacturing devices. Support for the Association Control Service Element (ACSE) ASE, as defined in ISO 8649, is required in every AE that supports the MMS ASE.

6.6 Addressing of Application Entities

ISO 7498-3 describes naming and addressing concepts in the OSI environment. This subclause describes how some of these concepts apply to MMS.

A Presentation-address is associated with a set of PSAPs in a single real system. An AE is attached to one or more Presentation Service Access Points (PSAPs) in order to make it addressable in the OSI environment.

Each AE is identified by one or more application-entity-titles (AE-titles), which are unambiguous throughout the OSI environment (OSIE). At any instant of time, each AE-title is bound to a single Presentation-address that identifies the set of PSAPs to which the AE is attached. This binding is recorded in the Application Title Directory Service, which contains information about AEs.

A system may access the Application Title Directory Service via the OSI Directory Service (ISO 9594). Conceptually, each system contains a directory function, from which AEs may obtain addressing information. This directory function may operate via the use of a local cache, or may utilize the Application Title Directory Service via the OSI Directory Service, in order to satisfy requests from AEs.

NOTE – This is only a conceptual model, and that there are many valid implementation styles.

In order for an application-entity to establish an application association (AA) with another AE, it must know the Presentation-address or obtain it by using the called AE-title to get the Presentation-address of the AE from the directory function. It then uses this Presentation-address to establish a presentation connection with the called AE. The mechanisms for control of associations are provided in ISO 8649 and ISO 8650 by the Association Control Service Element (ACSE). Since an AE is the representation of an AP within OSI, the establishment of communication to support an association with the called AE is the establishment of communication with the related AP.

MMS makes use of an "Application Reference" to identify an AE in another system, by making use of its title as defined in ISO 8649. The syntax of an Application Reference is defined in ISO/IEC 9506-2.

6.7 Application Context

The application context, which is established via the Association Control Service Element (ACSE) (ISO 8649), identifies the set of application service elements, their options, rules for use of the service elements, and their effects that are available on an application association. In the case of MMS, the application context associated with MMS indicates that the rules and requirements associated with ISO/IEC 9506 are in effect when the MMS application context is negotiated.

6.8 Presentation Context, Abstract Syntaxes, and Transfer Syntaxes

In OSI, there is an important distinction between the generic requirements of an application for the transfer of data and how those requirements are met in terms of a specific representation of data values. The former aspect of data description is referred to by the term "abstract syntax", while the latter aspect is referred to by the term "transfer syntax".

Abstract syntaxes are intimately associated with application protocol standards. ISO/IEC 9506-2 defines an abstract syntax matching its data transfer requirements. An abstract syntax can be viewed informally as describing the generic structure of data. In MMS, the set of type definitions provided in ISO/IEC 9506-2 constitutes an abstract syntax. The MMS abstract syntax may be supported by many different transfer syntaxes.

Transfer syntaxes are concerned with the way in which data is actually represented in terms of bit patterns during transmission. A transfer syntax may have attributes that are not related to the abstract syntaxes which it can support, but such attributes may be significant. For example, a transfer syntax may provide data compression or encryption. Such attributes (and how well they meet the requirements of the device) may influence the choice of syntaxes offered or selected for an instance of communication.

For the purpose of transferring data between MMS entities, it is necessary to identify the abstract syntax being used (MMS) and a transfer syntax that is capable of representing data values that are generated using this abstract syntax. The presentation service (ISO 8822) provides the mechanism by which appropriate abstract and transfer syntaxes are negotiated. A specific combination of an abstract syntax and a transfer syntax that is used for transfer of data using the presentation service is called a presentation context.

Abstract Syntax Notation One (ASN.1) is an example of a tool for specification of syntaxes and associated encoding rules. (The application of encoding rules to a particular abstract syntax may generate a transfer syntax.) The ASN.1 Specification (ISO 8824) has been chosen to describe the MMS abstract syntax. One possible transfer syntax for MMS is defined by the application of the ASN.1 Encoding Rules (ISO 8825) to the MMS abstract syntax. Other transfer syntaxes may be negotiated via the presentation layer. ISO/IEC 9506 requires that any system claiming conformance to ISO/IEC 9506 shall support the transfer syntax that results from the application of the ASN.1 Basic Encoding Rules (ISO 8825) to the MMS abstract syntax. Support of other transfer syntaxes is a local matter. See ISO/IEC 9506-2, clause 18.

7 The Virtual Manufacturing Device

7.1 Introduction

The MMS services define the externally visible behaviour of an MMS server application process. This behaviour is modelled by describing an entity called a Virtual Manufacturing Device (VMD). This clause describes the model for a VMD. It explains the VMD's relationship to the OSI application process and the OSI application entity (AE). It defines the structural elements of the VMD and introduces the abstract objects which exist at, and are manipulated by, a VMD on behalf of a client MMS-user.

An implementation of an MMS server must provide a mapping of the VMD model on to the functionality of a real manufacturing device. Guidance in the selection of a particular mapping may be found in various Companion Standards. These Companion Standards address the specific needs of discrete parts manufacturing systems - numerical controllers, programmable controllers, robotic controllers, and vision systems - and of batch and continuous process control systems.

NOTE – The MMS services do not constrain the behaviour of a client MMS application process, except with respect to valid sequences of primitives. Therefore a model of the MMS client application process is not given.

7.1.1 Relationship of the VMD to the OSI Model

A VMD exists within the MMS server application process. It constitutes that portion of an information processing task which makes available - for control, or monitoring, or both - a set of resources and functionality associated with a real manufacturing device. An application process may contain zero or more VMDs. If it does not define a VMD it may not act as an MMS-server.

Each VMD represents a virtual manufacturing device within that AP, and each VMD is logically separate from all other VMDs.

EXAMPLE — An MMS system which is connected to a non-MMS environment containing multiple attached manufacturing devices could be modelled as a single application process containing one VMD for each attached device or as several application processes, each containing a single VMD for a single, distinct, attached device. The clients of the VMDs in either case will see a particular attached device as a single VMD. Relative to MMS services, this VMD will appear to be independent of all other VMDs.

Each AP may contain zero or more AEs, such that an AE represents a set of communication capabilities of the AP (an AP that contains no AEs may not communicate in the OSI environment).

A VMD may wholly contain zero or more AEs. Each AE in a VMD represents a set of communication capabilities used by the aspects of the AP represented by a VMD. Each AE is related to one and only one VMD. When a VMD contains more than one AE, then it contains more than one set of communication capabilities.

At any instant of time, each AE-title is bound to a single presentation-address that identifies a set of PSAPs to which the AE-title is bound and therefore the AE-title is addressable in OSI. Communication with a particular AE is used to model communication with a VMD, and hence an AP. Application associations are modelled as taking place between AEs (and hence VMDs and APs).

An AE within an AP may have zero or more AE-invocations. For the purposes of MMS, each AE-invocation models an instance of usage of that AE in an application association. Hence, several AE-invocations of an AE may be used to model several application associations with a VMD that contains that AE. By generalization, there may simultaneously exist several AEs of a VMD, with several associations each (modelled by multiple AE-invocations). Note that in OSI a more generalized view allows an AE-invocation to have more than one application association.

An OSI Presentation Address which identifies one or more PSAPs addresses a single VMD. This binding of a Presentation Address to a VMD is of a relatively long duration. Figure 2 illustrates the relationship between an OSI application process acting as an MMS server, its VMDs, and the PSAPs used to access them.

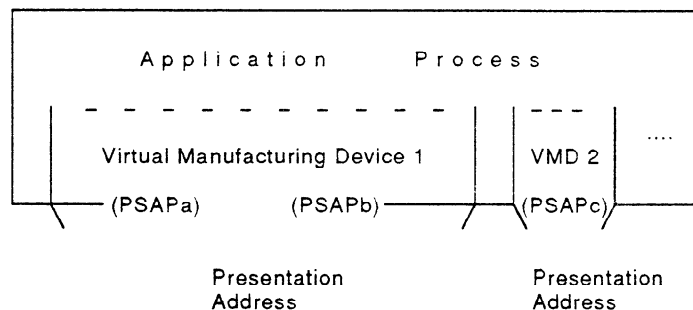


Figure 2 – The MMS Server Application Process

7.1.2 Relationship of the VMD to the Real Manufacturing Device

A VMD is an abstract representation of a specific set of resources and functionality at the real manufacturing device and a mapping of this abstract representation to the physical and functional aspects of the real manufacturing device. This mapping of a virtual resource to the underlying actual resource is of relatively long duration.

Generally, the resources of a given VMD are distinct from, and independent of, the resources of all other VMDs. When a virtual resource of two (or more) VMDs is mapped to the same underlying physical resource, a mechanism must be provided by the application process(es), and made available through the VMDs, so that clients of the various VMDs may coordinate their access to the single real resource. This coordination may be modelled by requiring each VMD to obtain control of a "virtual" semaphore whenever disruptive access to the virtual resource is being requested by an MMS service.

This virtual semaphore would then be mapped on to a real semaphore which controls access to the real resource. With this approach, the virtual semaphores of the various VMDs are independent when viewed using MMS services. In other words, users of the virtual semaphore of one VMD appear as though part of the resident system when viewed by users of the semaphores of the other VMDs. (Semaphores are described in clause 13.)

MMS describes the operation of a VMD by describing the abstract objects which are manipulated by it, and by describing the set of operations which may be performed on these objects through use of the MMS services.

7.2 The Structure of a VMD

Each VMD contains exactly one Executive Function and zero or more Program Invocations, each of which depend on one or more Domains. The Domain represents a specific use of a set of capabilities of the VMD. The state of the VMD, in the complete sense, is determined by the values of all the attributes of the VMD, including all the attributes of its Domains and their subordinate objects. The elements of the VMD are described below.

The VMD Object Model

Object: VMD

- Key Attribute: Executive Function
- Attribute: Vendor Name
- Attribute: Model Name
- Attribute: Revision
- Attribute: List Of Abstract Syntaxes Supported
- Attribute: Logical Status (STATE-CHANGES-ALLOWED,
NO-STATE-CHANGES-ALLOWED, LIMITED-SERVICES-PERMITTED,
SUPPORT-SERVICES-ALLOWED)
- Attribute: List Of Capabilities
- Attribute: Physical Status (OPERATIONAL, PARTIALLY-OPERATIONAL,
INOPERABLE, NEEDS-COMMISSIONING)
- Attribute: List Of Program Invocations
- Attribute: List Of Domains
- Attribute: List Of Transaction Objects
- Attribute: List Of Upload State Machines (ULSM)
- Attribute: Lists of Other VMD-specific Objects
- Attribute: Additional Detail

7.2.1 The Executive Function

The existence of a fully functional Executive Function exactly corresponds with the existence of the VMD.

7.2.2 Vendor Name

This attribute is a character string which identifies the vendor of the system which supports this VMD.

7.2.3 Model Name

This attribute is a character string which identifies the model of the system which supports this VMD. The value of this string is normally assigned by the vendor.

7.2.4 Revision

This attribute is a character string which identifies the revision level of the system which supports this VMD. The value of this string is normally assigned by the vendor.

7.2.5 List Of Abstract Syntaxes Supported

This attribute identifies the set of Abstract Syntaxes which this VMD is able to support in the MMS Application Context. This set includes any abstract syntaxes defined by a Companion Standard as well as any abstract syntax which can be recognized as an encoding for Load Data (see 10.3) and Execution Argument (see 11.4). The abstract syntax defined in ISO/IEC 9506-2, clause 19, shall not be included in the list.

7.2.6 Logical Status

There are four separate levels of functionality available through MMS which are described by the Logical Status attribute of the Executive Function.

7.2.6.1 STATE-CHANGES-ALLOWED

In this case, all MMS services which are supported by this VMD may be performed.

7.2.6.2 NO-STATE-CHANGES-ALLOWED

In this case, the only MMS services which may be performed (if the VMD supports the service) are:

- Abort
- Conclude
- Cancel
- GetAlarmEnrollmentSummary
- GetAlarmSummary
- GetCapabilityList
- GetDomainAttributes
- GetEventActionAttributes
- GetEventConditionAttributes
- GetEventEnrollmentAttributes
- GetNamedTypeAttributes
- GetNamedVariableListAttributes
- GetNameList
- GetProgramInvocationAttributes
- GetScatteredAccessAttributes
- GetVariableAccessAttributes
- Identify
- Initiate
- Read
- ReadJournal
- ReportEventActionStatus
- ReportEventConditionStatus
- ReportEventEnrollmentStatus
- ReportJournalStatus
- ReportPoolSemaphoreStatus
- ReportSemaphoreEntryStatus
- ReportSemaphoreStatus
- Status

7.2.6.3 LIMITED-SERVICES-PERMITTED

In this case, the only MMS services which may be performed are: Abort, Conclude, Status and Identify.

7.2.6.4 SUPPORT-SERVICES-ALLOWED

In this case, all MMS services supported by the VMD except the Start, Stop, Reset, Resume, and Kill services may be performed.

7.2.7 List Of Capabilities

A VMD represents the real device to the client by providing a set of capabilities which may be used by the client in order to effect some control or monitoring (or both) activity through the VMD.

A capability is a locally defined resource (physical or logical) or a locally defined set of resources which can be identified by a character string. The definition and management of capabilities is outside the scope of this part of ISO/IEC 9506. However, it is assumed that the existence and status of capabilities is known by the Executive Function, and that the Executive Function contains sufficient knowledge in order to "allocate" capabilities to the various Domains which may be created. No object model is provided for a capability, since it is considered primitive from the MMS point of view.

A capability may be sharable (or not) depending on local criteria. Additionally, a capability may be distinct, or it may encompass (or be encompassed by) one or more other capabilities.

NOTE – An implementation of an MMS server should provide a mapping of the VMD model on to the functionality of a real manufacturing device. Guidance in the selection of a particular mapping may be found in various Companion Standards. In particular, the Companion Standards will aid in the identification of Domains and Program Invocations with features of real systems. The attribute "capability" is intended to allow expression of features of the implementation needed for relating any real implementation to the VMD model and to the extended model provided by the Companion Standard. The use of capabilities in any MMS implementation requires prior agreement between implementations of MMS client and server.

EXAMPLE 1 – In a programmable controller application, a capability "BlockMem1" may be assigned to represent a collection of memory address ranges which are available for the purpose of customizing a standard control program for the manufacture of a specific part.

EXAMPLE 2 – In a robotics application, the capability "Arm1" may represent the set of sensors, actuators, and logic associated with a specific arm of a multi-arm robot.

7.2.8 Physical Status

Associated with each capability is one or more attributes which describe the state of the capability. It is outside the scope of part of ISO/IEC 9506 to provide a uniform representation of these attributes. However, it is useful to provide a standard representation of the gross aspects of all the capabilities, taken together, in order to characterize the operational state of the hardware. The Physical Status attribute (of the entire collection of capabilities) represents this attribute of the real device.

NOTE – This status refers to the hardware associated with the device's operation. It is unrelated to the ability of the device to communicate in an OSI environment.

7.2.8.1 OPERATIONAL

The real device associated with this VMD has no known deficiencies and is able to perform its intended tasks.

7.2.8.2 PARTIALLY-OPERATIONAL

One or more functions of the real device cannot be performed due to hardware malfunctions or limitations.

7.2.8.3 INOPERABLE

One or more significant problems exist at the real device which prevent it from doing any useful task.

7.2.8.4 NEEDS-COMMISSIONING

The device is in a state such that a local commissioning process needs to be performed before useful tasks can be accomplished.

7.2.9 List Of Program Invocations

A Program Invocation consists of a set of procedural and data elements contained within Domains together with execution control information. These elements may be predefined within the VMD, or they may be dynamically defined (for example by creation of a Domain through use of the MMS load services), or both. The Program Invocation itself may be predefined within the VMD, or it may be dynamically created and deleted either by local means or through the use of MMS services. An object model of the Program Invocation is given in clause 11.

7.2.10 List Of Domains

A Domain represents a specific instance of use of a set of capabilities of the VMD. A Domain includes those aspects of a VMD which are associated with a specific element (possibly all) of a coordinated control or monitoring (or both) strategy. The allocation of the capabilities of a VMD to the Domains may be static or it may be dynamic. If the allocation is static, then the Domain is predefined within the MMS server and its name is considered to be known. If the allocation is dynamic, the Domain comes into existence and is removed from the VMD either through the action of MMS services or through local actions. Within a given VMD, either or both types of Domains may exist. Domains are further described and an object model provided in clause 10.

A Domain may be empty or it may contain "information". The "information" may be program instructions for some processor or tables of values or other classes of data or all of the above. For dynamic Domains which come into existence through MMS services, the Domains are implicitly created by the process of loading their contents. Dynamic Domains may also be created through actions of the Program Invocation when it is in execution or through other local means.

The Domain content is often closely associated with a file. The concept of file, although it appears in some MMS services, is not defined within MMS. The file may be part of a virtual Filestore (ISO 8571) which is part of the Application Process which contains the VMD, or it may be locally defined. If the "information" within a file is a part of the VMD and can be used directly by the VMD (as a component of a Program Invocation), it can be considered to be a Domain. The file, however, must be considered a separate object, outside the scope of the VMD.

EXAMPLE — A numerical control application may require several Domains. One Domain may consist of the resources associated with part program execution and other Domains may be associated with one or more material handling functions. A third static Domain may be associated with a predefined calibration procedure. Thus Domains not only subset the physical resources - such as memory and I/O devices - of the numerical controller, but they also partition the functionality of the controller.

Most MMS objects are defined subordinate to (within the name space of) a Domain. Thus, a Domain represents a single name space in which MMS objects must be uniquely identifiable.

7.2.11 Transaction Object

Most MMS services are confirmed services (see ISO/IEC 9506-2, clause 6). When the VMD receives an indication primitive for one of the confirmed services, a Transaction Object is created which governs the processing of this service. The description of that object follows:

Object: Transaction

- Key Attribute: Invoke ID
- Key Attribute: Application Association Identifier
- Attribute: List Of Pre-execution Modifiers
- Attribute: Current Modifier Reference
- Attribute: Confirmed Service Request
- Attribute: List Of Post-execution Modifiers
- Attribute: Cancelable (TRUE, FALSE)

Invoke ID

This attribute is an integer which serves to identify the transaction within the Application Association.

Application Association Identifier

This attribute identifies the application association over which this transaction object is created.

List Of Pre-execution Modifiers

An ordered list which identifies modifiers which must be satisfied before execution of the Confirmed Service Request attribute can begin.

Current Modifier Reference

References the Semaphore Entry object or the Event Enrollment object controlling the current modifier's execution.

Confirmed Service Request

Service identifier and Argument of the pending service.

List Of Post-execution Modifiers

References the Semaphore Entry objects which this service invocation owns due to a processed Attach to Semaphore modifier.

Cancelable

Initially true, the service may set this attribute false (local issues). When true, cancel works; when false, cancel does not work.

7.2.11.1 Initialization of Transaction Objects

A transaction object shall be created upon receipt of an indication service primitive for an MMS confirmed service, and deleted after the MMS-user issues a response service primitive for that service instance. The number of transaction objects which may exist at any time is governed by the negotiated maximum number of services outstanding (see 8.2).

When an indication service primitive is received, the List Of Pre-execution Modifiers attribute of the newly created transaction object shall be set based on the List Of Modifier parameter on the indication service primitive (see 5.4). The Current Modifier Reference is set to one, indicating the first Modifier on the List Of Pre-execution Modifiers, or to zero if there are no Pre-execution Modifiers. The List Of Post-execution Modifiers is set to empty (no modifiers are on this list).

The Confirmed Service Request attribute of the transaction object shall be set based on the Argument parameter received on the indication service primitive.

The Cancelable attribute shall initially be set to true. The Cancelable attribute of the transaction object shall not be set to false during the processing of the pre-execution modifiers. After all pre-execution modifiers have been processed, the cancelable attribute may be set to false by the MMS-user at any time as a local matter, subject to the requirements for the Cancel service and the service named in the Confirmed Service Request. The cancelable attribute shall have been set to false by the time that post-execution modifier processing takes place.

7.2.11.2 Processing of Transaction Objects

After a transaction object has been initialized by the MMS-user, processing may begin on that object. Processing begins with the List Of Pre-execution Modifiers, followed by execution of the Confirmed Service Request, and terminates with processing of the List Of Post-execution modifiers.

NOTE – Although this part of ISO/IEC 9506 requires that Transaction Objects be initialized in the order of receipt, there is no such requirement for the order of processing of these objects. To enforce serialization of actions, the MMS client should wait for a service response before issuing subsequent service requests.

Each of the Pre-execution modifiers shall be processed in order. Each modifier shall complete successfully before the next modifier can be processed. If a modifier fails (see definition of particular modifiers), the MMS-user shall process the Post-execution modifier list (as specified below) and then issues a response(-) service primitive, specifying the appropriate error class and code, and the transaction object shall be deleted. The Current Modifier Reference shall be set to indicate the modifier currently being processed in the list. The Current Modifier Reference shall be identified by an integer, where 1 indicates the first element in the list. For each AttachToSemaphore modifier which is successfully processed, an entry shall be made in the List Of Post-execution Modifier attribute (in order to allow relinquishing of control of the semaphores after service execution). The List Of Post-execution Modifier shall be constructed in reverse order, such that the first pre-execution modifier becomes the last post-execution modifier.

After all pre-execution modifiers have been processed successfully (if any are specified on the indication service primitive), the Confirmed Service Request shall be executed in accordance with the service procedure specified in this part of ISO/IEC 9506 for the named service.

Following completion of the service request, the MMS-user shall process the post-modifiers in the List Of Post-execution Modifiers attribute in the order that they are specified in the list (note that this is the opposite of the order in which the pre-modifiers were executed).

After all post-execution modifiers have been processed, the response service primitive shall be issued by the MMS-user. The Current Modifier Reference attribute is used to indicate the current modifier during the processing of this list.

7.2.12 Upload State Machines

The Upload State Machine object (ULSM) is created through the InitiateUploadSequence service which is described in clause 10.

7.2.13 Other VMD-specific Named Objects

There are many other named objects which may have VMD-specific scope. The attributes of such objects shall be considered part of the VMD. General considerations for all named objects are described in 7.3, and the objects themselves are described in separate clauses of this part of ISO/IEC 9506.

7.2.14 Additional Detail

This attribute contains zero or more attributes whose meaning and syntax are defined by appropriate Companion Standards.

7.3 Specification of Named Objects

MMS objects are normally referenced by name. The name of an object shall be unique within its scope of definition and within the class of object it identifies. In MMS, the names of objects are defined within one of three scopes: within a VMD instance, within a Domain, or within a single application association.

7.3.1 Scope of Names

In general, MMS names can have one of three scopes, VMD-specific Domain-specific, and Application Association-specific.

7.3.1.1 VMD-specific Scope

A name which has VMD-specific scope shall be unique among all VMD-specific objects of the same object class. Such a name may be referenced by all clients of the VMD instance on any application association with it. VMD-specific objects are not automatically deleted when the MMS application association ceases to exist.

7.3.1.2 Domain-specific Scope

A Domain represents a single flat name space. A name which is Domain-specific shall be unique for its class of object within the Domain in which it is defined. The unique identification of such an object requires the specification of the name of the Domain and the name of the object, thus implying a two level hierarchy.

7.3.1.3 Application Association-specific Scope

A name having "Application Association-specific" (AA-specific) scope may only be referenced by the client MMS-user for which the name was defined, and only on the specific application association over which the name's definition is valid. The definition of an object bearing an application association-specific name, if not explicitly deleted previously, shall be deleted whenever the defining application association ceases to exist.

7.3.2 Classes of Objects

The specific instances of MMS objects which can be named are listed in Table 1. Not all objects can have all possible Name Scopes. The allowed combinations are indicated by Xs in Table 1.

Table 1 – Name Class and Scope

	VMD	Domain	AA	clause
Named Variable Objects	X	X	X	12
ScatteredAccess Objects	X	X	X	12
Named Variable List Objects	X	X	X	12
Named Type Objects	X	X	X	12
Semaphore Objects	X	X		13
Event Condition Objects	X	X	X	15
Event Action Objects	X	X	X	15
Event Enrollment Objects	X	X	X	15
Journal Objects	X		X	16
Domain Objects	X			10
Program Invocation Objects	X			11
Operator Station Objects	X			14

In addition to the requirement that the name of each object be unique within its Name Scope and for the class of the object, there is an additional requirement on the first two objects listed. Named Variable Objects and Scattered Access Objects share a common name space. Therefore the name of any such object shall be unique within this common name space for its Name Scope.

NOTE – There are other limitations on the names of Event Conditions, Event Actions, and Event Enrollments. Creation of a semaphore automatically creates an Event Condition of the same name (see 13.4). Therefore, a semaphore cannot be proposed for creation which has the same name as an existing Event Condition. If the monitor parameter is selected, the creation of a Program Invocation automatically creates an Event Condition, and Event Action, and an Event Enrollment, all having the same name as the Program Invocation (see 11.2). Therefore, a Program Invocation cannot be proposed for creation whose name is the same as an existing Event Condition, Event Action, or Event Enrollment.

7.3.3 Object Lifetime

Each of the objects listed above has a lifetime within the VMD which can be inferred from the scope of its name.

VMD-specific Scope

Such objects exist as long as the VMD exists (unless explicitly deleted).

Domain-specific Scope

Such objects exist as long as the Domain on which they depend exists (unless explicitly deleted).

AA-specific Scope

Such objects exist only as long as the Application Association over which they were defined continues to exist (unless explicitly deleted).

7.3.4 Object Visibility

The Name Scope of a named object also determines the visibility of the object. An object whose name is VMD-specific or Domain-specific may be referenced by any association to the VMD; an object whose name is AA-specific may only be referenced by the association over which the object was defined.

7.3.5 Creation of MMS objects

Each of the MMS objects can either be static, that is predefined within the implementation, or dynamic, coming into existence during the course of operation of the VMD. Static objects are usually not MMS deletable, while dynamic objects are usually MMS deletable, but there may be exceptions to either rule. Static objects are predefined by the implementation and have names assigned to them either by the implementor, or in conformance to this part of ISO/IEC 9506 or to one of the Companion Standards.

Dynamic objects can come into existence either (1) through explicit MMS service procedures, (2) through local action of the system or of the system operators, or (3) through the execution of some Program Invocation. The means of local creation is outside the scope of this part of ISO/IEC 9506. However, if a locally created object is to be visible to MMS services, its external behaviour, including all its visible attributes, shall be consistent with this service definition.

7.3.6 Deletion of MMS Objects

All MMS objects which have the attribute MMS Deletable equal to true may be removed from the VMD through appropriate MMS service requests. In addition to this explicit method of deletion, MMS objects may also be deleted through local action of the system or of the system operators or through the execution of some Program Invocation. The means of local deletion is outside the scope of this part of ISO/IEC 9506. Objects which are subordinate to a Domain are automatically deleted when the Domain is deleted. This is true even if the subordinate object has its MMS-Deletable attribute set to FALSE.

7.3.7 Alteration of MMS Objects

All MMS objects have as part of their description a list of externally visible attributes. In addition to the MMS services which alter these attributes, these attributes may also be changed through the local action of the system or of the system operators, or through the execution of a Program Invocation. In particular, the values of variables associated with Domains or with the VMD directly may change due to the execution of a Program Invocation. The means of local alteration of Object Attributes is outside the scope of this part of ISO/IEC 9506.

A VMD should, if possible, guarantee that access to any MMS object required by an MMS service is not interruptible. In other words, for the entire duration of the MMS service, the object should have a set of attributes that represent the state of the VMD at a single instant of time. Since such guarantees may not be possible, or if possible for some objects may not be possible for all objects, the static conformance statement for the VMD shall state whether uninterruptible access is supported and, if supported, under what constraints it is guaranteed.

7.4 Object Name Structure

The parameter Object Name occurs frequently in the specification of MMS services. The structure of the Object Name is shown in Table 2.

Table 2 – Object Name

Object Name	Req Rsp	Ind Cnf
Name Scope	M	M(=)
AA-specific	S	S(=)
Item Identifier	M	M(=)
Domain-specific	S	S(=)
Domain Identifier	M	M(=)
Item Identifier	M	M(=)
VMD-specific	S	S(=)
Item Identifier	M	M(=)

7.4.1 Name Scope

This parameter specifies which of the possible scopes is to be used for this Object Name.

7.4.2 AA-specific

Some named objects may have an AA-specific name. Such names must be defined over the association on which they exist.

7.4.2.1 Item Identifier

This parameter, of type Identifier, is the name of the object. This parameter shall be unique within this application association for the class of object named.

7.4.3 Domain-specific

Named objects which depend on a Domain may have names which are Domain-specific.

7.4.3.1 Domain Identifier

This parameter, of type Identifier, is itself a VMD-specific name of the Domain which contains the named object.

7.4.3.2 Item Identifier

This parameter, of type Identifier, is the name of the object within the named Domain. This parameter shall be unique within the named Domain for the class of object named.

7.4.4 VMD-specific

Any named object may have a VMD-specific name.

7.4.4.1 Item Identifier

This parameter, of type Identifier, is the name of the object. This parameter shall be unique within the VMD for the class of object named.

7.5 Services on the VMD

ISO/IEC 9506, clause 9, describes the MMS services which operate on the VMD in its entirety. In addition to these services, provision is made for groups defining Companion Standards to define additional services which operate on the VMD.

8 Environment And General Management Services

8.1 Introduction

The environment and general management services contain the Initiate, Conclude, Abort, Cancel and Reject services. These services allow the MMS-user

- a) to initiate communication with another MMS-user in the MMS environment, and to establish the requirements and capabilities that support that communication;
- b) to conclude communication with another MMS-user in the MMS environment in a graceful manner;
- c) to abort communications with another MMS-user in the MMS environment in an abrupt manner;
- d) to cancel pending service requests;
- e) to receive notification of protocol errors that occur.

8.1.1 Environment Management State Diagram

This clause defines the state diagram for entering and leaving the MMS environment. The initial state for both the calling and called MMS users shall be the state "No MMS Environment". The state diagram is depicted from the point of view of an MMS-user. The state of the MMS environment is within the context of a specific application association. Restrictions on the use of MMS services within this environment shall not apply to other environments established by the MMS-user on different application associations.

8.1.2 Restriction on Use of MMS Services

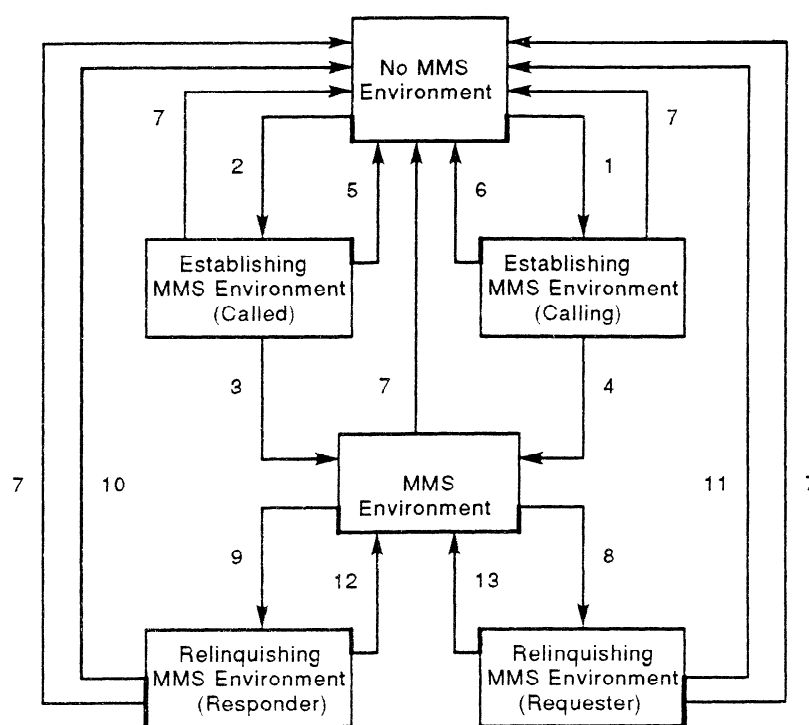
This clause places restrictions on the use of MMS services for certain states. The restrictions in this clause are supplementary to other restrictions placed by other clauses in this part of ISO/IEC 9506.

8.1.2.1 The "No MMS Environment" State

While in the state "No MMS Environment", an MMS-user shall only issue the initiate.request service primitive.

8.1.2.2 The "Establishing MMS Environment (Calling) State

While in the state "Establishing MMS Environment (Calling)", an MMS-user shall only issue the abort.request service primitive.



Transitions:

- | | |
|---------------------------------------|--------------------------|
| 1 - initiate.request | 8 - conclude.request |
| 2 - initiate.indication | 9 - conclude.indication |
| 3 - initiate.response + | 10 - conclude.response + |
| 4 - initiate.confirm+ | 11 - conclude.confirm+ |
| 5 - initiate.response- | 12 - conclude.response- |
| 6 - initiate.confirm- | 13 - conclude.confirm- |
| 7 - abort.request or abort.indication | |

Figure 3 - Environment Management State Diagram

8.1.2.3 The "Establishing MMS Environment (Called) State

While in the state "Establishing MMS Environment (Called)", an MMS-user shall only issue the initiate.response or abort.request service primitives.

8.1.2.4 The "MMS Environment" State

Clauses 8 to 16 place requirements on the use of service primitives in the state "MMS Environment". This state is divided into a series of substates, which are described in those clauses. All MMS states except the "No MMS Environment", "Establishing MMS Environment (Calling)", "Establishing MMS Environment (Called)", "Relinquishing MMS Environment (Requester)" and "Relinquishing MMS Environment (Responder)" are substates of the "MMS Environment" state.

The only events which cause an exit from the "MMS Environment" state are the issuance of an abort.request, the issuance of a conclude.request service primitive, the receipt of an abort.indication, or the receipt of a conclude.indication service primitive.

While in the state "MMS Environment", an MMS-user may issue any request or response service primitive, subject to the following restrictions:

- other clauses of this part of ISO/IEC 9506 restrict the usage of services and service primitives via sequencing requirements;

- b) a response primitive shall not be issued unless a request primitive for a corresponding service indication has been received (see ISO/IEC 9506, clause 7);
- c) the initiate.request service primitive shall not be issued.

8.1.2.5 The "Relinquishing MMS Environment (Requester)" State

While in the state "Relinquishing MMS Environment (Requester)", the only request primitive that the MMS-user may issue is the abort.request service primitive. Note that responses, either (+) or (-) may continue to be issued.

8.1.2.6 The "Relinquishing MMS Environment (Responder)" State

While in the state "Relinquishing MMS Environment (Responder)", the MMS-user may only issue the abort.request and conclude.response service primitives.

8.2 Initiate Service

The Initiate service shall be used to establish the MMS environment and to allow the communicating MMS-users to exchange information regarding their capabilities and requirements. The Initiate service must complete successfully before any other services may be carried out between a pair of MMS-users in the MMS environment.

8.2.1 Structure

The structure of the component service primitives is shown in Table 3.

Table 3 – Initiate Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument (COMP)	M	M			
Local Detail Calling	U	U(=)			
Proposed Max Serv Outstanding Calling	M	M			
Proposed Max Serv Outstanding Called	M	M			
Proposed Data Structure Nesting Level	U	U			
Init Request Detail	M	M			
Result(+) (COMP)			S	S(=)	
Local Detail Called			U	U(=)	
Negotiated Max Serv Outstanding Calling			M	M(=)	
Negotiated Max Serv Outstanding Called			M	M(=)	
Negotiated Data Structure Nesting Level			U	U(=)	
Init Response Detail			M	M	
Result(-)			S	S(=)	
Error Type			M	M(=)	

NOTE – The collection of these parameters is contained in the user data portion of the A-ASSOCIATE request of ACSE. Additional parameters are required to complete the A-ASSOCIATE request. The mapping of the Initiate Service on to the A-ASSOCIATE is discussed in ISO/IEC 9506-2, clause 17. Owing to the mapping of Initiate on to A-Associate, resolution of collisions in the Initiate service is handled by ACSE mechanisms for resolving collisions in A-Associate.

8.2.1.1 Argument

This parameter shall convey the service specific parameters of the Initiate service request.

For those parameters in the request primitive, the MMS-provider in the calling system may reduce the values supplied by the calling MMS-user. For those parameters in the indication primitive, the MMS-provider in the called system may reduce the values from those in the InitiateRequestPDU, except for the Services Supported Calling parameter. Reductions of the values must follow the rules for reduction as specified in the parameter descriptions. No other modifications are permitted to these values.

NOTE — The purpose of allowing the service providers to reduce values as specified by the user is to provide a mechanism which may be used to ensure that the values specified by the user do not exceed the capabilities of the service provider. Implementation of this feature is a local matter.

8.2.1.1.1 Local Detail Calling

When present, this parameter, of type integer, shall represent information about the Calling MMS-User's implementation. The content and interpretation of this field is a local matter and not a subject for further standardization.

8.2.1.1.2 Proposed Max Serv Outstanding Calling

This parameter, of type integer, shall identify the proposed maximum number of Transaction Object instances whose Application Association Identifier attribute references this association that may be created at the Calling MMS-user.

The value of this parameter may be reduced by the MMS-provider. The value in the indication primitive shall be less than or equal to the value in the request primitive. The value in the request or indication primitives shall not be less than zero.

8.2.1.1.3 Proposed Max Serv Outstanding Called

This parameter, of type integer, shall identify the proposed maximum number of Transaction Object instances that may be created at the called MMS-user whose Application Association Identifier attribute references this association.

The value of this parameter may be reduced by the MMS-provider. The value in the indication primitive shall be less than or equal to the value in the request primitive. The value in the request or indication primitives shall not be less than zero.

8.2.1.1.4 Proposed Data Structure Nesting Level

This parameter, of type integer, shall indicate the proposed maximum number of levels of nesting supported by both of the MMS-users that may occur within any data element used in the association. Absence of this parameter shall indicate an unlimited number of nesting levels.

The request and indication primitives shall specify the maximum nesting level of Type Specification (explicit or derived) that the calling MMS-user desires to use in the negotiated MMS environment. A value of zero shall indicate that only simple types are allowed (see 12.2).

The value of this parameter may be reduced by the MMS-provider. The value in the indication primitive shall be less than or equal to the value in the request primitive.

8.2.1.1.5 Init Request Detail

This parameter shall specify additional information necessary for the establishment of an instance of MMS communication. For the abstract syntax defined in ISO/IEC 9506-1 and ISO/IEC 9506-2, this parameter is described in 8.2.3.

8.2.1.2 Result(+)

This parameter shall indicate that the requested service has succeeded. When success is indicated, the following additional parameters shall also apply.

8.2.1.2.1 Local Detail Called

When present, this parameter, of type integer, shall represent information about the Called MMS-User's implementation. The content of this field is a local matter and not a subject for further standardization.

8.2.1.2.2 Negotiated Max Serv Outstanding Calling

This parameter, of type integer, shall identify the maximum number of Transaction Object instances that may be created at the calling MMS-user whose Application Association Identifier attribute references this association.

The negotiated maximum number of services outstanding in the response primitive shall be less than or equal to the Proposed Max Serv Outstanding Calling parameter in the indication primitive, but shall not be less than zero.

8.2.1.2.3 Negotiated Max Serv Outstanding Called

This parameter, of type integer, shall identify the maximum number of Transaction Object instances that may be created at the called MMS-user whose Application Association Identifier attribute references this association.

The negotiated maximum number of services outstanding in the response primitive shall be less than or equal to the Proposed Max Serv Outstanding Called parameter in the indication primitive, but shall not be less than zero.

8.2.1.2.4 Negotiated Data Structure Nesting Level

This parameter, of type integer, shall indicate the maximum number of levels of nesting supported by both of the MMS-users that may occur within any data element used in the association. Absence of this parameter shall indicate an unlimited nesting level.

This parameter shall be equal to or less than the Proposed Data Structure Nesting level parameter in the indication primitive, but shall not be less than zero.

The response and confirmation primitives shall specify the maximum nesting level of Type Specification (explicit or derived) that shall be used in the negotiated MMS environment. A value of zero shall indicate only simple types are allowed (see 12.2).

8.2.1.2.5 Init Response Detail

This parameter shall specify additional information necessary for the establishment of an instance of MMS communication. For the abstract syntax defined in ISO/IEC 9506-1 and ISO/IEC 9506-2 this parameter is described in 8.2.4.

8.2.1.3 Result(-)

This parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

8.2.2 Service Procedure

The called MMS-user shall issue a response primitive indicating success in the Result parameter if that MMS-user is willing to accept communications in the MMS environment under the constraints identified in the indication primitive, or if alternate values can be proposed (according to the negotiation rules), with the requesting MMS-user. Otherwise, a response primitive indicating the Result(-) parameter shall be issued.

Successful execution of the Initiate service shall result in the establishment of the MMS environment. The MMS environment shall only be established through the use of the Initiate service. The Initiate service shall not be used within an established MMS environment. If an Initiate Request PDU is received on an association where an established MMS environment exists, a Reject PDU shall be issued with a Reject PDU Type of PDU-ERROR and a Reject Code of ILLEGAL-ACSE-MAPPING.

8.2.3 Init Request Detail Parameter

The structure of the Init Request Detail Parameter is shown in Table 4.

Table 4 – Init Request Detail Parameter

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Proposed Version Number	M	M			
Proposed Parameter CBB	M	M			
Services Supported Calling	M	M			

8.2.3.1 Proposed Version Number

This parameter, of type integer, shall contain a number which represents a minor version number of ISO/IEC 9506-1 and ISO/IEC 9506-2. The minor version number of ISO/IEC 9506-1 and ISO/IEC 9506-2 shall be that specified in ISO/IEC 9506-2, clause 17. This parameter, the Proposed Version Number, is the proposed minor version number which will be used in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, for this instance of communication. Proposal of a number greater than one indicates support, in this presentation context, for all minor versions between one and the number proposed, inclusive.

NOTE – Major versions of ISO/IEC 9506-1 and ISO/IEC 9506-2 are reflected through the definition and registration of distinct abstract syntaxes. (see ISO/IEC 9506-2, clauses 7 and 19). Minor versions are reflected in the minor version number parameter. Minor versions of ISO/IEC 9506-1 and ISO/IEC 9506-2 at the same major version level are compatible with major versions of ISO/IEC 9506-1 and ISO/IEC 9506-2 with smaller minor version numbers.

The value of this parameter may be reduced by the MMS-provider if it cannot support the requested value. The value in the indication primitive shall be less than or equal to the value in the request primitive, but not less than one.

8.2.3.2 Proposed Parameter CBB

This parameter, of type bitstring, shall specify the set of parameter conformance building blocks (CBB) which are proposed to be supported in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, on this application association.

The value of this parameter in the request primitive shall specify the set of Parameter CBBs supported by the Calling MMS-user.

The value of this parameter in the indication primitive shall specify the intersection of the set of Parameter CBBs supported by the Calling MMS-user and the set of Parameter CBBs supported by the MMS-provider.

The possible parameter conformance building blocks are specified in ISO/IEC 9506-2, clause 18. The assignment of a parameter CBB to an individual bit of a CBB bitstring type is specified in ISO/IEC 9506-2, clause 8. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

8.2.3.3 Services Supported Calling

This parameter, of type bitstring, shall specify support by the Calling MMS-user of a set of services for use in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, on the application association.

The value of the parameter in the indication primitive shall specify the intersection of the set of services supported by the Calling MMS-user and the set of services supported by the MMS-provider.

The assignment of an service to an individual bit of the bitstring type is specified in ISO/IEC 9506-2, clause 19. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

Support for confirmed services shall be defined as the ability to receive a request indication and execute the service procedure defined for the responder role. Support of unconfirmed services shall be defined as the ability to accept an indication primitive and to pass the parameters to the service interface. Support of a modifier shall be defined as the ability to accept an indication primitive that contains the Modifier and to execute properly the service procedure defined for the Modifier.

If a confirmed service, an unconfirmed service, or Modifier is supported, then a Reject PDU shall not be issued on receipt of that service or modified service, except in the case of a protocol error. If a confirmed service, an unconfirmed service, or modifier is not supported, then a Reject PDU shall be issued on receipt of that service or modified service with a reject code of "UNRECOGNIZED SERVICE".

NOTE – This parameter only represents the services defined in this part of ISO/IEC 9506. Additional services defined in Companion Standards are represented in a separate parameter (see annex A).

8.2.4 Init Response Detail Parameter

The structure of the Init Response Detail Parameter is shown in Table 5.

Table 5 – Init Response Detail Parameter

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Negotiated Version Number			M	M(=)	
Negotiated Parameter CBB			M	M(=)	
Services Supported Called			M	M	

8.2.4.1 Negotiated Version Number

This parameter, of type integer, shall contain a number which represents a minor version number of ISO/IEC 9506-1 and ISO/IEC 9506-2. The minor version number of ISO/IEC 9506-1 and ISO/IEC 9506-2 shall be that specified in ISO/IEC 9506-2, clause 17. This parameter, the Negotiated Version Number, shall be the minor version number which will be used in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, for this instance of communication. This number shall be less than or equal to the Proposed Version Number parameter in the request primitive. It shall not be reduced to less than one.

NOTE – Major versions of ISO/IEC 9506 are reflected through the definition and registration of distinct abstract syntaxes (see ISO/IEC 9506-2, clauses 7 and 19). Minor versions are reflected in the minor version number parameter. Minor versions of ISO/IEC 9506-1 and ISO/IEC 9506-2 at the same major version level are compatible with versions of ISO/IEC 9506-1 and ISO/IEC 9506-2 with smaller minor version numbers.

8.2.4.2 Negotiated Parameter CBB

This parameter, of type bitstring, shall specify the negotiated set of parameter conformance building blocks (CBB) which are to be supported in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, on this application association.

The value of this parameter in the response primitive shall specify the intersection of the set of Parameter CBBs supported by the Called MMS-user and the set of Parameter CBBs specified by the Proposed Parameter CBB parameter in the indication primitive.

The possible parameter conformance building blocks are specified in ISO/IEC 9506-2, clause 18. The assignment of a Parameter CBB to an individual bit of a CBB bitstring type is specified in of ISO/IEC 9506-2, clause 8. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

8.2.4.3 Services Supported Called

This parameter, of type bitstring, shall specify support by the Called MMS-user of a set of services for use in the presentation context derived from the abstract syntax defined in ISO/IEC 9506-2, clause 19, on the application association.

The value of the parameter in the confirmation primitive shall specify the intersection of the set of services supported by the Called MMS-user and the set of services supported by the MMS-provider.

The assignment of a service to an individual bit of the bitstring type is specified in ISO/IEC 9506-2, clause 8. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

Support for confirmed services shall be defined as the ability to receive an indication primitive and properly execute the service procedure defined for the responder role. Support of unconfirmed services shall be defined as the ability to accept an indication primitive and to pass the parameters to the service interface. Support of a modifier shall be defined as the ability to accept an indication primitive that contains the Modifier and to properly execute the service procedure defined for the Modifier.

If a confirmed service, an unconfirmed service, or Modifier is supported, then a Reject PDU shall not be issued on receipt of that service or modified service, except in the case of a protocol error. If a confirmed service, an unconfirmed service, or modifier is not supported, then a Reject PDU shall be issued on receipt of that service or modified service with a reject code of "UNRECOGNIZED SERVICE".

NOTE – This parameter only represents the services defined in this part of ISO/IEC 9506. Additional services defined in Companion Standards are represented in a separate parameter (see annex A).

8.3 Conclude Service

The Conclude service may be used to cause the orderly relinquishing of the MMS environment. An MMS-user requests the Conclude service to indicate that it has completed requests which it had planned and that no further requests will be issued.

8.3.1 Structure

The structure of the component service primitives is shown in Table 6.

Table 6 – Conclude Service

Table 6 (Cont.) – Conclude Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Result(+)			S	S(=)	
Result(-) Error Type			S M	S(=) M(=)	

8.3.1.1 Argument

There are no service specific parameters for the Conclude service request.

8.3.1.2 Result(+)

This parameter shall indicate that the requested service has succeeded.

8.3.1.3 Result(-)

This parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

8.3.2 Service Procedure

The requesting MMS-user shall issue a Conclude request primitive, which begins the Conclude service. Once this primitive has been issued, the requesting MMS-user shall not issue any further request primitives until a Conclude confirm primitive is received from the MMS-provider (except the Abort request primitive, which may be issued at any time). The requesting MMS-user may continue to issue response primitives in order to complete service requests from the peer MMS-user.

The requesting MMS-user shall not issue a Conclude request primitive to a peer on an association if a Domain with a State attribute value equal to LOADING, COMPLETE, INCOMPLETE, D1, D2, D3, or D9 exists at the requesting MMS-user as a result of a request from the peer MMS-user on the association.

In the peer open system, the responding MMS-user shall issue a Conclude response primitive indicating whether or not it accepts the conclusion (via the Result parameter) before any other service request primitives may be issued by that MMS-user (except the Abort request service primitive, which may be issued at any time). The MMS-user shall not accept conclusion on an association if it is awaiting any response from the peer MMS-user for a confirmed service or the Conclude Service.

A responding MMS-user shall not accept a Conclude attempt on an association if any of the following conditions exist:

- The responding MMS-user has received an indication on the association for a confirmed service request from the peer MMS-user for which it has not issued a response;
- An Upload State Machine exists on the responding MMS-user as a result of a request from the peer MMS-user on the association;
- A Domain with a State attribute value equal to LOADING, COMPLETE, or INCOMPLETE exists on the responding MMS-user as a result of a request from the peer MMS-user on the association;
- The responding MMS-user has control of a semaphore on the requesting MMS-user on the association;

- e) The requesting MMS-user has control of a semaphore on the responding MMS-user on the association.

Upon issuing a Conclude response primitive with a Result(+) parameter, the MMS Environment shall be considered terminated. No further request or response primitives shall be issued by the responding MMS-user in that MMS environment on that association.

Upon issuing a Conclude response primitive with a Result(-), the MMS environment shall not be affected and the responding MMS-user may continue to issue request and/or response service primitives.

Upon receiving a Conclude confirm primitive with a Result(+), the requesting MMS-user shall consider the MMS environment to be terminated, and no further request or response primitives shall be issued by the requesting MMS-user in the MMS environment on that association.

Upon receiving a Conclude confirmation primitive with a Result(-), the requesting MMS-user shall consider the MMS Environment to be unaffected, and may continue to issue request or response service primitives in the MMS environment.

NOTE 1 – The effect of a failed Conclude service confirmation is as if no Conclude request had ever been issued.

Upon successful completion of the Conclude service, all application association specific objects associated with the terminated MMS environment are released unless other specifications for that object are explicitly stated in this part of ISO/IEC 9506.

NOTE 2 – It is possible that both of the peers will initiate a Conclude attempt at or near the same time resulting in failure of both attempts. If both peers retry the Conclude, on result of failure, then this situation could potentially (although very unlikely) continue indefinitely. Through use of application design, this situation can be avoided. One possible method is to designate the Calling MMS-user as the peer that will initiate the Conclude attempt should this situation arise.

8.4 Abort Service

The Abort service shall be used to relinquish the MMS environment abruptly and without negotiation. The MMS-user shall issue the Abort request primitive to indicate that it wishes immediately, and without negotiation, to discontinue communications in the MMS Environment on the application association. The effect of the Abort service may be to destroy previously issued requests and/or responses issued by either of the MMS-users. The MMS Abort service is derived from and makes use of the A-ABORT service of ACSE (see 17.2 in ISO/IEC 9506-2).

NOTE – The abort indication service primitive may also be generated by the MMS-provider.

8.4.1 Structure

The structure of the component service primitives is shown in Table 7.

Table 7 – Abort Service

Parameter Name	Req	Ind	CBB
Argument Locally Generated	M	M M	

8.4.1.1 Argument

This parameter shall convey the service specific parameters of the Abort service. The abort service primitives additionally shall contain the parameters specified for the ACSE abort service to which the MMS Abort is mapped. These parameters are specified in ISO 8649.

8.4.1.1.1 Locally Generated

This parameter, of type boolean, shall indicate whether the abort request was generated by the system in which the MMS-user receiving the indication is located (indicated by the true value), or whether the abort request was received by that system (indicated by the false value). This parameter shall be provided by the MMS-provider.

8.4.2 Service Procedure

In the case of an Abort initiated by the MMS-user, the peer MMS-user shall be notified of the user requested abort by receipt of an Abort indication service primitive, and the MMS Environment shall be terminated.

In the case of an Abort initiated by the MMS-provider, both MMS-users shall be notified of the provider abort by receipt of Abort indication primitives (if this is possible), and the MMS Environment shall be terminated.

Upon completion of the Abort service, all application association specific objects associated with the terminated MMS environment are released unless other specifications for that object are explicitly stated in ISO/IEC 9506.

8.5 Cancel Service

The Cancel service is used by an MMS-user to cancel a request that has previously been issued, but has not yet completed. Only confirmed services may be cancelled (see clause 5).

8.5.1 Structure

The structure of the component service primitives is shown in Table 8.

Table 8 – Cancel Service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument Original Invoke ID	M M	M(=) M(=)			
Result(+) Original Invoke ID			S M	S(=) M(=)	
Result(-) Original Invoke ID Error Type			S M M	S(=) M(=) M(=)	

8.5.1.1 Argument

This parameter shall convey the service specific parameters for the Cancel service request.

8.5.1.1.1 Original Invoke ID

This parameter, of type integer, shall specify the invoke ID of the service whose cancellation is desired. (An invoke ID is provided for every confirmed service request primitive, see clause 5.)

8.5.1.2 Result(+)

The Result(+) parameter shall indicate that the Cancel service request succeeded. When success is indicated service specific parameters shall also be returned.

8.5.1.2.1 Original Invoke ID

This parameter, of type integer, shall specify the invoke ID of the service that was cancelled. (An invoke ID is provided for every confirmed service request primitive, see clause 5.)

8.5.1.3 Result(-)

The Result(-) parameter shall indicate that the Cancel service request failed. The Error Type parameter, which is defined in detail in clause 17, provides the reason for failure.

8.5.1.3.1 Original Invoke ID

This parameter, of type integer, shall specify the invoke ID of the service whose cancellation was desired. (An invoke ID is provided for every confirmed service request primitive, see clause 5.)

8.5.1.3.2 Error Type

The Error Type parameter, which is defined in detail in clause 17, provides the reason for the failure.

8.5.2 Service Procedure

Upon receiving the Cancel indication service primitive, the responding MMS-user shall attempt to cancel the designated service, assuming that it can do this non-destructively and can return to a state that is logically as if the service indication targeted for cancellation had not been received. Destructive cancellation is permitted for the Start, Stop, Resume, and Reset Services provided it follows the service procedures defined for these services. If this is not the case, or if the service request invocation identified has not been requested, or if a response primitive has already been issued, or if the cancelable attribute of the Transaction object associated with the designated service has a value of false, then the Cancel request shall fail. In this case, the responding MMS-user shall issue a Cancel response service primitive with the Result(-) parameter with the appropriate Cancel error codes for the Error Type parameter. The responding MMS-user shall continue with the original service and eventually return a response, either positive or negative, as appropriate.

Otherwise, the service invocation identified shall be successfully cancelled by the responding MMS-user, and a Cancel response primitive with the Result(+) parameter shall be issued. The responding MMS-user shall also issue a response primitive with a Result(-) parameter for the cancelled service indicating the SERVICE-PREEMPT Error Class with an Error Code of CANCEL. The state of the responder shall then appear logically as if the service targetted for cancellation had not been requested, unless otherwise stated in the service procedure for the cancelled service. Additional limitations on requesting the cancel service are specified in ISO/IEC 9506-2, clause 6.

8.6 Reject Service

The Reject service is a provider-initiated service that is used to inform the MMS-users of the occurrence of a protocol error. A definition of a protocol error may be found in ISO/IEC 9506-2, clause 3.

8.6.1 Structure

The structure of the component service primitives is shown in Table 9.

Table 9 – Reject Service

Parameter Name	Ind	CBB
Detected Here	M	
Original Invoke ID	C	
Reject PDU Type	M	
Reject Code	M	

8.6.1.1 Detected Here

This parameter, of type boolean, shall indicate whether the protocol error that results in the Reject service was detected at the local end or remote end of the underlying presentation connection. If true, the protocol error was detected at the local end, otherwise the error was detected at the remote end.

NOTE – Handling of local error conditions between an MMS-user and MMS-provider is a local matter.

8.6.1.2 Original Invoke ID

This parameter, of type integer, shall indicate the original invoke ID of the PDU which was found to be in error. Its presence is conditional on whether the invoke ID could be determined. If there is no invoke ID specified in the PDU being rejected, this parameter shall not be present.

8.6.1.3 Reject PDU Type

This parameter, of type integer, shall indicate the type of the PDU that caused the protocol error. The value PDU-ERROR shall be used when the PDU being rejected is not a syntactically valid MMS PDU. The possible values are as follows:

CONFIRMED-REQUESTPDU
CONFIRMED-RESPONSEPDU
CONFIRMED-ERRORPDU
UNCONFIRMEDPDU
PDU-ERROR
CANCEL-REQUESTPDU
CANCEL-RESPONSEPDU
CANCEL-ERRORPDU
CONCLUDE-REQUESTPDU
CONCLUDE-RESPONSEPDU
CONCLUDE-ERRORPDU